Department of Computer Science & Engineering

Object Oriented Programming through JAVA Laboratory (AK20)

ANNAMACHARYA INTITUTE OF TECHNOLOGY & SCIENCES:: TIRUPATHI

(Autonomous Institution - UGC, Govt. of India)

(Recognized under 2(f) and 12 (B) of UGC ACT 1956)

(Affiliated to JNTUA, Anantapuramu, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade, Accredited by Institute of Engineers, Kolkata, A-Grade Awarded by AP Knowledge Mission)

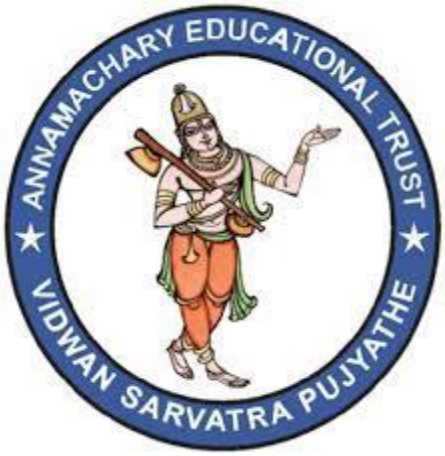
Tirupathi, Andhrapradesh - 517 520

Laboratory Manual

for

Object Oriented Programming through JAVA Laboratory

Course Code	:	**CSE(20APC0514) / CIC(20APC3610)**
Regulations	:	**AK20**
Class	:	**II Year II Semester**
Branch	:	**CSE/CIC**



Department of Computer Science & Engineering

ANNAMACHARYA INTITUTE OF TECHNOLOGY & SCIENCES:: TIRUPATHI

(Autonomous Institution - UGC, Govt. of India)

(Recognized under 2(f) and 12 (B) of UGC ACT 1956)

(Affiliated to JNTUA, Anantapuramu, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade, Accredited by Institute of Engineers, Kolkata, A-Grade Awarded by AP Knowledge Mission)

Tirupathi, Andhrapradesh - 517 520

TABLE OF CONTENTS

SNo	Content	Page Number
1	Institute Vision & Mission	1
2	Department Vision & Mission	1
3	PEOs	2
4	POs	3
5	PSOs	4
6	Course Outcomes	5
7	Laboratory Instructions	6

SNO	Week	Problem Statement	Page Number
		Syllabus	7-9
		Introduction to Object Oriented Programming through JAVA laboratory	10-15
1	Week-1	How to Download and Install Eclipse to Run Java	16-22
2		To write a JAVA program to display default value of all primitive data type of JAVA	22-23
3		Write a java program to find prime numbers between 1 to n.	23-24
4		Write a Java program that prints all real solutions to the quadratic equation ax2+bx+c=0. Read in a, b, c and use the quadratic formula.	24-25
5	Week-2	To write a JAVA program to implement class mechanism. – Create a class, methods and invoke them inside main method	25-25
6		Write a java program that reads a line of integer, and then displays and then displayseach integer and sum of all integers(Using StringTokenizer class of java.util).	25-26
7		Write a Java program to multiply two given matrices	26-28
8	Week-3	Write a java program to illustrate the concept of Single level and multi level inheritance	28-29
9		To write a JAVA program to implement constructor overloading	29
10		Develop a Java application to generate Electricity bill. Create a class with the following members: Consumer no., consumer name, previous month reading, current month reading, type of EB connection (i.e. domestic or commercial). Commute the bill amount using	29-33

		the following tariff. If the type of the EB connection is domestic, calculate the amount to be paid as follows: First 100 units - Rs. 1 per unit; 101-200 units - Rs. 2.50 per unit; 201 -500 units - Rs. 4 per unit; 501 units - Rs. 6 per unit If the type of the EB connection is commercial, calculate the amount to be paid as follows: First 100 units - Rs. 2 per unit; 101-200 units - Rs. 4.50 per unit; 201 -500 units - Rs. 6 per unit; 501 units - Rs. 7 per unit.	
11	Week-4	Write a JAVA program give example for “super” keyword	33-34
12		Create a base class Fruit which has name ,taste and size as its attributes. A method called eat() is created which describes the name of the fruit and its taste. Inherit the same in 2 other class Apple and Orange and override the eat() method to represent each fruit taste.	35-36
13		Write a program to create a class named shape. It should contain 2 methods, draw() and erase() that prints —Drawing Shape‖ and —Erasing Shape‖ respectively. For this class, create three sub classes, Circle, Triangle and Square and each class should override the parent class functions - draw () and erase (). The draw() method should print —Drawing Circle‖, —Drawing Triangle‖ and —Drawing Square‖ respectively. The erase() method should print —Erasing Circle‖, —Erasing Triangle‖ and —Erasing Square‖ respectively. Create objects of Circle, Triangle and Square in the following way and observe the polymorphic nature of the class by calling draw() and erase() method using each object. Shape c=new Circle(); Shape t=new Triangle(); Shape s=new Square();	37-38
14	Week-5	Write a java program to illustrate the concept of class with method overloading.	39
15		Write Java program(s) on use of inheritance, preventing inheritance using final, abstract classes.	39-41
16		Write a java program to demonstrate interfaces and abstract classes.	41-42
17		Write Java program(s) on dynamic binding, differentiating method overloading and overriding.	42-44
18		To write a JAVA program that implements Runtime polymorphism	45-46

<table>
<tr><td>19</td><td rowspan="2">Week-6</td><td>Develop a java application to implement currency converter (Dollar to INR, EURO to INR, Yen) using Interfaces.</td><td>46-50</td></tr>
<tr><td>20</td><td>To write a JAVA program to implement Interface.</td><td>50-53</td></tr>
<tr><td>21</td><td rowspan="4">Week-7</td><td>To write a JAVA program that describes exception handling mechanism</td><td>53-54</td></tr>
<tr><td>22</td><td>To write a JAVA program Illustrating Multiple catch clauses</td><td>54</td></tr>
<tr><td>23</td><td>To write a JAVA program for creation of Illustrating throw</td><td>55</td></tr>
<tr><td>24</td><td>To write a JAVA program for creation of Java Built-in Exceptions</td><td>55-57</td></tr>
<tr><td>25</td><td rowspan="2">Week-8</td><td>To write a JAVA program for creation of User Defined Exception Program</td><td>58</td></tr>
<tr><td>26</td><td>a) Write a Java Program to demonstrate the following String Handlings.
i. String Length& Concatenation.
ii. Character Extraction.
iii. String Comparison.
iv. Searching and modifying String.</td><td>58-65</td></tr>
<tr><td rowspan="2">27</td><td rowspan="2">Week-9</td><td>b) Write a Java Program to demonstrate String Buffer Class and String Builder Class</td><td>65-67</td></tr>
<tr><td>a) Write a Java program for multi-thread implementation.</td><td>67-70</td></tr>
<tr><td rowspan="2">28</td><td rowspan="2">Week-10</td><td>b) Write a Java program to implement producer consumer problem using inter-thread communication mechanism</td><td>70-74</td></tr>
<tr><td>a) Practice any two Programs on Collections.</td><td>75-84</td></tr>
<tr><td rowspan="4">29</td><td rowspan="4">Week-11</td><td>b) Practice any two Programs on String Tokenizer & Scanner.</td><td>85-87</td></tr>
<tr><td>a) Write a Java Program to develop an applet that displays a simple message.</td><td>87-88</td></tr>
<tr><td>b) Develop an applet that receives an integer in one text field, and computes its factorial Value and returns it in another text field, when the button named —Compute‖ is clicked.</td><td>89-91</td></tr>
<tr><td>c) Write a java program to handle keyboard events.</td><td>91-96</td></tr>
<tr><td rowspan="3">30</td><td rowspan="3">Week-12</td><td>d) Write a java program to handle Mouse events</td><td>96-102</td></tr>
<tr><td>a) Write a Java Program to demonstrate AWT Label & Button.</td><td>102-104</td></tr>
<tr><td>b) Write a Java Program to demonstrate JLabel, JTextField & JButton.</td><td>104-108</td></tr>
<tr><td>c) Write a program to design a calculator using event driven programming paradigm of java</td><td>108-123</td></tr>
</table>

INSTITUTE VISION & MISSION

Vision

"To Promote Excellence in Technical and Management Education."

Mission

- **Strengthen the Learning-Teaching Process for Holistic Development.**
- **Upgrade Physical Infrastructure to meet the Curriculum needs.**
- **Enhance Industry-Institute Interactions to acquire Professional Competency.**
- **Promote Innovation and Research to address Challenges of Society.**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision

- To achieve excellence in the field of Computer Science and Engineering with professional competency.

Mission

M1: Provide quality education to achieve excellence.

M2: Upgrade infrastructure and Technologies to meet the learner's needs.

M3: Establish linkages with Government and Industry to enhance technical skills, entrepreneurship and innovations.

M4: Support research to serve the needs of the society.

Program Educational Objectives

The department of CSE has developed and adopted Program Educational Objectives (PEO's) for guiding UG programs towards the mission and vision which reflects three aspects of student learning: Cognitive, Affective and Behavioral. PEOs are expected to attain by the students few years after their graduation.

PEO's (Program Educational Objectives)	
PEO1	Graduates will be able to become competent professionals rendering service to IT and ITES industry
PEO2	Graduates will be able to become lifelong learners by adapting new technologies to sustain in their career.
PEO3	Graduates will be able to become technocrats to serve the needs of the society with ethical values

Program Outcomes

PO's (Program Outcomes)	
PO No.	**Program Outcome Description**
PO 1	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO 2	**Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO 3	**Design / Development of solution:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO 4	**Conduct investigation of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO 5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO 6	**The engineer & society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO 7	**Environment & sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO 8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO 9	**Individual & team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO 10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO 11	**Project management & finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to

	manage projects and in multidisciplinary environments.
PO 12	**Life long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes:

PSO's (Program Specific Outcomes)	
PSO1	Demonstrate the working principles of the hardware and software aspects of computer systems.
PSO2	Design products based on the professional engineering practices with effective strategies

Course Outcomes:

CO's (Course Outcomes)	
CO1	Demonstrate java compiler and eclipse platform and learn how to use net beans IDE to create java application
CO2	Ability to create user friendly interfaces
CO3	Ability to solve the problem using object oriented approach and design solutions which are robust
CO4	Implement exception handling and Templates

List of CO's	PO no. and keyword	Competency Indicator	Performance Indicator
CO1	PO1: Apply the knowledge of mathematics	1.1	1.1.1
CO2	PO1:Apply the knowledge of mathematics	1.1	1.1.1
CO3	PO2:Analyse complex engineering problems	2.1	2.1.3
CO4	PO2:Analyse complex engineering problems	2.4	2.4.1

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:

   a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.

   b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.

   c. Proper Dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out ; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.

10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly. Head of the Department Principal

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

Course Code	Object Oriented Programming through Java Lab		L	T	P	C
20APC0514			0	0	4	2
Pre-requisite	NIL	Semester	II-II			

Course Objectives:

- To experiment with the syntax and semantics of java language and gain experience with java programming
- Learn to use object orientation to solve problems and use java language to implement them.

Course Outcomes (CO):

- Demonstrate java compiler and eclipse platform and learn how to use net beans IDE to create java application
- Ability to create user friendly interfaces
- Ability to solve the problem using object oriented approach and design solutions which are robust
- Implement exception handling and Templates

List of Experiments:

Week-1: (Unit-1)

Installation of Java software, study of any integrated development environment, Use Eclipse or Net bean platform and acquaint with the various menus. Create a test project, add a test class and run it.

Practice Java Basic Programs on Classes and Objects.

Week-2: (Unit-1)

Develop a Java application to generate Electricity bill. Create a class with the following members: Consumer no., consumer name, previous month reading, current month reading, type of EB connection (i.e domestic or commercial). Commute the bill amount using the following tariff.
If the type of the EB connection is domestic, calculate the amount to be paid as follows:

First 100 units - Rs. 1 per unit; 101-200 units - Rs. 2.50 per unit; 201 -500 units - Rs. 4 per unit;

501 units - Rs. 6 per unit. If the type of the EB connection is commercial, calculate the amount to be paid as follows: First 100 units - Rs. 2 per unit; 101-200 units - Rs. 4.50 per unit; 201 -500 units - Rs. 6 per unit; > 501 units - Rs. 7 per unit.

Write a java program to illustrate the concept of class with method overloading. C) Write a java program to illustrate the concept of class with Constructors overloading.

Week-3:(Unit-2)

a) Write a program to create a class named shape. It should contain 2 methods, draw() and erase() that prints “Drawing Shape” and “Erasing Shape” respectively. For this class, create three sub classes, Circle, Triangle and Square and each class should override the parent class functions - draw () and erase (). The draw() method should print “Drawing Circle”, “Drawing Triangle” and “Drawing Square” respectively. The erase() method should print “Erasing Circle”, “Erasing Triangle” and “Erasing Square” respectively. Create objects of Circle, Triangle and Square in the following way and observe the polymorphic nature of the class by calling draw() and erase() method using each object. Shape c=new Circle(); Shape t=new Triangle(); Shape s=new Square();

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem

JAVA Laboratory Record Programs

b) Write a Java Program to demonstrate inheritance &usage of super

Week-4:(Unit-2)

Write a Java Program to implement multilevel inheritance.
Write a Java program to implement the method overriding
Write a Java program to implement dynamic method dispatch.

Week-5:(Unit-2)

Write a Java program to implement abstract class.
Write a Java Program to implement Packages.
Write a Java Program to implement Access Protection in Packages.

Week-6:(Unit-2)

Write a Java program to demonstrate interfaces.
Write a Java program to implement the multiple inheritance using interfaces.

Week-7:(Unit-3)

Write a Java program to implement the exception handling mechanism.
Write a Java program to implement the nested try statement.
Write a Java program to implement your own exception class.

Week-8:(Unit-3)

Write a Java Program to demonstrate the following String Handlings.
String Length& Concatenation.
Character Extraction.
String Comparison.
Searching and modifying String.
Write a Java Program to demonstrate String Buffer Class.

Week-9:(Unit-4)

Write a Java program for multi-thread implementation.
Write a Java program to implement producer consumer problem using inter-thread communication mechanism.

Week-10:(Unit-4)

Practice any two Programs on Collections.
Practice any two Programs on String Tokenizer & Scanner.

Week-11:(Unit-5)

Write a Java Program to develop an applet that displays a simple message.
Develop an applet that receives an integer in one text field, and computes its factorial Value and returns it in another text field, when the button named ―Compute‖ is clicked.
Write a java program to handle keyboard events.
Write a java program to handle Mouse events

Week-12:(Unit-5)

Write a Java Program to demonstrate AWT Label & Button.
Write a Java Program to demonstrate JLabel, JTextField & JButton.
Write a program to design a calculator using event driven programming paradigm of java

References:

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

1. Herbert Schildt.Java. The complete reference, TMH. 9thEdition.
2. H.M.Dietel and P.J.Dietel, Java How to Program 6thEdition,PearsonEducation/PHI
3. Y.Daniel Liang, Introduction to Java programming, Pearson Education, 6thEdition.
4. Cay Horstmann, Big Java, 2ndedition, Wiley Student Edition, Wiley India Private Limited.

Online Learning Resources/Virtual Labs:

http://www.javatpoint.com

List of CO's	PO no. and keyword	Competency Indicator	Performance Indicator
CO1	PO1: Apply the knowledge of mathematics	1.1	1.1.1
CO2	PO1:Apply the knowledge of mathematics	1.1	1.1.1
CO3	PO2:Analyse complex engineering problems	2.1	2.1.3
CO4	PO2:Analyse complex engineering problems	2.4	2.4.1

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

Introduction TO JAVA Programming Laboratory

JAVA: Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. James Gosling initiated the Java language project in June 1991 for use in one of his many set-top box projects. The language, initially called Oak after an oak tree that stood outside Gosling's office, also went by the name Green and ended up later being renamed as Java, from a list of random words.

Laboratory Objective

Upon successful completion of this Lab the student will be able to:

1. Understand the concept of OOP as well as the purpose and usage principles of inheritance, polymorphism, encapsulation and method overloading.
2. Understand fundamentals of programming such as variables, conditional and iterative execution, methods, etc.
3. Identify classes, objects, members of a class and the relationships among them needed for a specific problem.
4. Understand fundamentals of object-oriented programming in Java, including defining classes, invoking methods, using class libraries, etc.
5. Create Java application programs using sound OOP practices (e.g., interfaces and APIs) and proper program structuring (e.g., by using access control identifies, automatic documentation through comments, error exception handling)
6. Have the ability to write a computer program to solve specified problems.
7. Develop programs using the Java Collection API as well as the Java standard class library.
8. Use the Java SDK environment to create, debug and run simple Java programs

Overview of Java

Java Is Important to the Internet, The Internet helped catapult Java to the forefront of programming, and Java, in turn, has had a profound effect on the Internet. The reason for this is quite simple: Java expands the universe of objects that can move about freely in cyberspace. In a network, two very broad categories of objects are transmitted between the server and our personal computer: passive information and

dynamic, active programs.

Java can be used to create two types of programs: applications and applets. An application is a program that runs on your computer, under the operating system of that computer. An applet is an application designed to be transmitted over the Internet and executed by a Java- compatible Web browser.

Features of JAVA

- Simple
- Secure
- Portable
- Object-oriented
- Robust
- Multithreaded
- Architecture-neutral
- Interpreted
- High performance
- Distributed
- Dynamic

JDK

The Java Development Kit (JDK) is an implementation of either one of the Java SE, Java EE or Java ME platforms. The JDK includes a private JVM and a few other resources to finish the development of a Java Application.

The JDK has as its primary components a collection of programming tools, including:

- appletviewer – this tool can be used to run and debug Java applets without a web browser
- apt – the annotation-processing tool.
- extcheck – a utility that detects JAR file conflicts
- idlj – the IDL-to-Java compiler. This utility generates Java bindings from a given Java IDL file.
- jabswitch – the Java Access Bridge. Exposes assistive technologies on Microsoft Windows systems.
- java – the loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, no longer comes with Sun JDK, and instead it has been

replaced by this new java loader.

- javac – the Java compiler, which converts source code into Java bytecode
- javadoc – the documentation generator, which automatically generates documentation from source code comments
- jar – the archiver, which packages related class libraries into a single JAR file. This tool also helps manage JAR files.
- javafxpackager – tool to package and sign JavaFX applications
- jarsigner – the jar signing and verification tool
- javah – the C header and stub generator, used to write native methods
- javap – the class file disassembler
- javaws – the Java Web Start launcher for JNLP applications
- JConsole – Java Monitoring and Management Console
- jdb – the debugger
- jhat – Java Heap Analysis Tool (experimental)
- jinfo – This utility gets configuration information from a running Java process or crash dump. (experimental)
- jmap – This utility outputs the memory map for Java and can print shared object memory maps or heap memory details of a given process or core dump. (experimental)
- jmc – Java Mission Control
- jps – Java Virtual Machine Process Status Tool lists the instrumented HotSpot Java Virtual Machines (JVMs) on the target system. (experimental)
- jrunscript – Java command-line script shell.
- jstack – utility that prints Java stack traces of Java threads (experimental)
- jstat – Java Virtual Machine statistics monitoring tool (experimental)
- jstatd – jstat daemon (experimental)
- keytool – tool for manipulating the keystore
- pack200 – JAR compression tool
- policytool – the policy creation and management tool, which can determine policy for a Java runtime, specifying which permissions are available for code from various sources
- VisualVM – visual tool integrating several command-line JDK tools and lightweight performance and memory profiling capabilities
- wsimport – generates portable JAX-WS artifacts for invoking a web service.
- xjc – Part of the Java API for XML Binding (JAXB) API. It accepts an XML schema and generates Java classes.

1. Data Types used in JDK

Each row contains the data type and size and range of the data type. The list of available data types in Java is shown in table below

Name	Width	Range
byte	8	–128 to 127
short	16	–32,768 to 32,767
int	32	–2,147,483,648 to 2,147,483,647
long	64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	32	1.4e–045 to 3.4e+038
double	64	4.9e–324 to 1.8e+308
Char	2	0 to 65,536
Boolean	1	True or false

2. Security

As we are likely aware, every time that we download a ―normal‖ program, we are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Even so, most users still worried about the possibility of infecting their systems with a virus. In addition to viruses, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords, by searching the contents of your computer‘s local file system. Java answers both of these concerns by providing a ―firewall‖ between a networked application and our computer.
When we use a Java-compatible Web browser, we can safely download Java applets without fear of viral infection or malicious intent. Java achieves this protection by confining a Java program to the Java execution environment and not allowing it access to other parts of the computer.
The ability to download applets with confidence that no harm will be done and that no security will be breached is considered by many to be the single most important aspect of Java.

The key that allows Java to solve both the security and the portability problems just described is that the output of a Java compiler is not executable code. Rather, it is bytecode. Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). That

is, in its standard form, the JVM is an interpreter for bytecode. This may come as a bit of a surprise. Translating a Java program into bytecode helps makes it much easier to run a program in a wide variety of environments. The reason is straightforward: only the

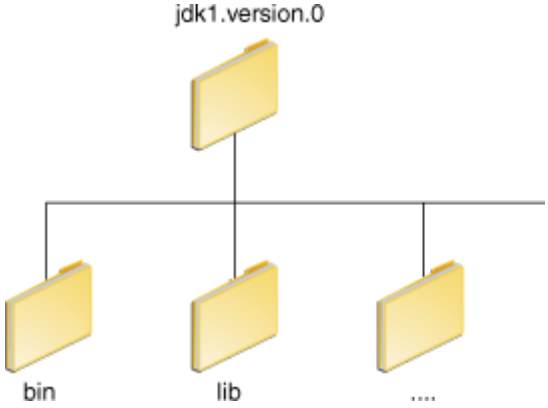
JVM needs to be implemented for each platform. Once the run-time package exists for a given system, any Java program can run on it. Because the execution of every Java program is under the control of the JVM, the JVM can contain the program and prevent it from generating side effects outside of the system. When a program is interpreted, it generally runs substantially slower than it would run if compiled to executable code.

3. PATH and CLASSPATH

Setting PATH and CLASSPATH environment variables on Microsoft Windows, Solaris, and Linux are as follows.

Install the Java Development Kit (JDK) software.

After installing the software, the JDK directory will have the structure shown below.



The bin directory contains both the compiler and the launcher.

Update the PATH Environment Variable (Microsoft Windows)

We can run Java applications just fine without setting the PATH environment variable. Or, we can optionally set it as a convenience.

Set the PATH environment variable if we want to be able to conveniently run the executables (javac.exe, java.exe, javadoc.exe, and so on) from any directory without having to type the full path of the command. If we do not set the PATH variable, we need to specify the full path to the executable every time we run it, such as:

C:\Java\jdk1.7.0\bin\javac MyClass.java

The PATH environment variable is a series of directories separated by semicolons (;). Microsoft Windows looks for programs in the PATH directories in order, from left to right. We should have only one bin directory for the JDK in the path at a time (those following the first are ignored), so if one is already present, we can update that

particular entry.

The following is an example of a PATH environment variable: C:\Java\jdk1.7.0\bin;C:\Windows\System32\;C:\Windows\;C:\Windows\System32\Wbem

It is useful to set the PATH environment variable permanently so it will persist after rebooting. To make a permanent change to the PATH variable, use the **System** icon in the Control Panel. The precise procedure varies depending on the version of Windows:

Windows 7/8/10

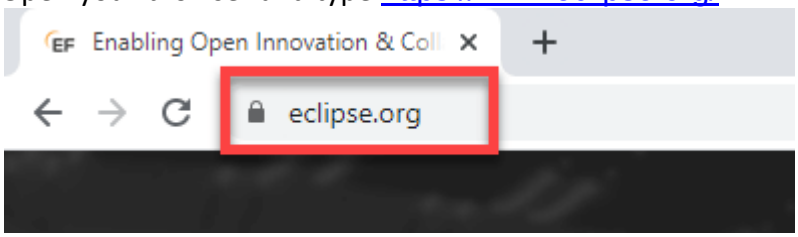
1. Select **Start**, select **Control Panel**. double click **System**, and select the **Advanced** tab.
2. Click **Environment Variables**. In the section **System Variables**, find the PATH environment variable and select it. Click **Edit**. If the PATH environment variable does not exist, click New.
3. In the **Edit System Variable** (or **New System Variable**) window, specify the value of the PATH environment variable. Click **OK**. Close all remaining windows by clicking **OK**.

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

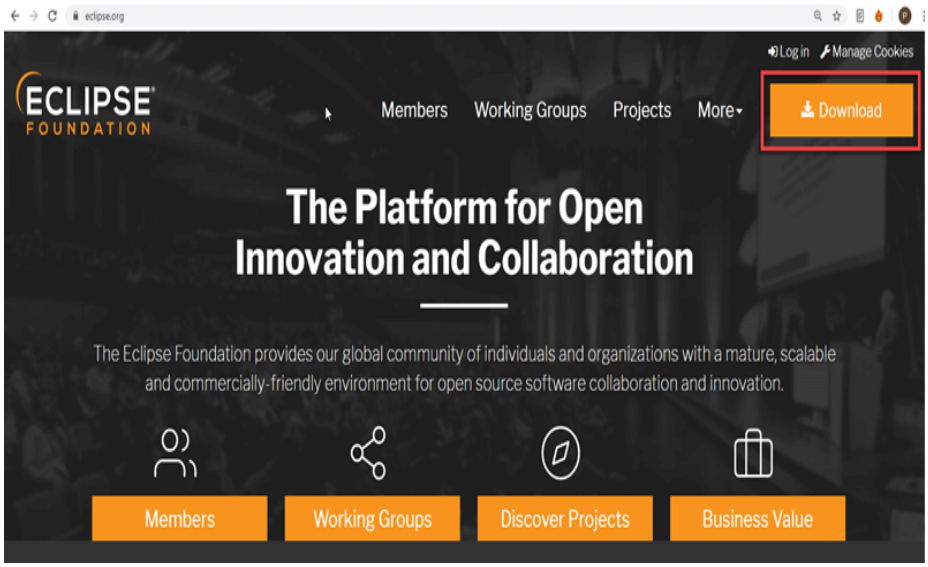
How to Download and Install Eclipse to Run Java

Step 1) Installing Eclipse

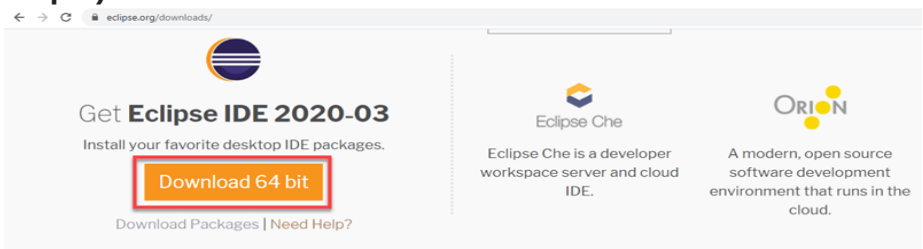
Open your browser and type https://www.eclipse.org/



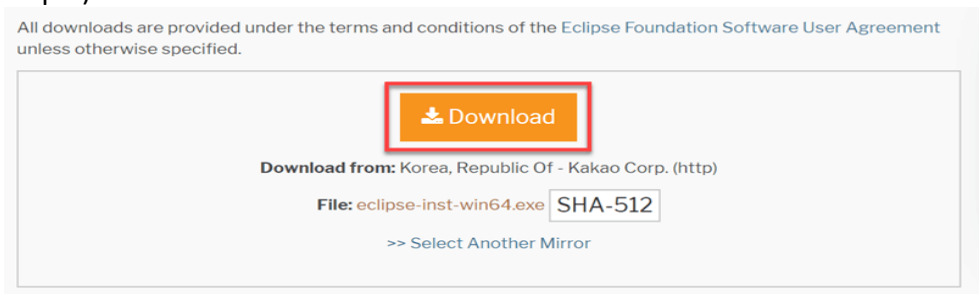
Step 2) Click on "Download" button.



Step 3) Click on "Download 64 bit" button



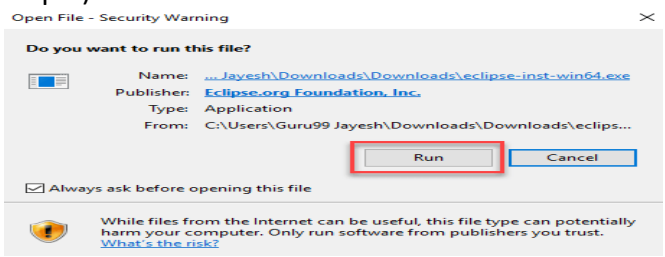
Step 4) Click on "Download" button



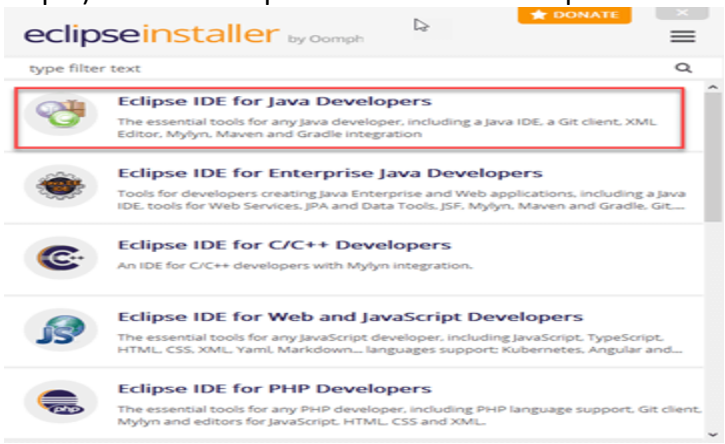
Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

Step 4) Install Eclipse.
Click on “downloads” in Windows file explorer.
Click on “eclipse-inst-win64.exe” file.

Step 5) Click on Run button

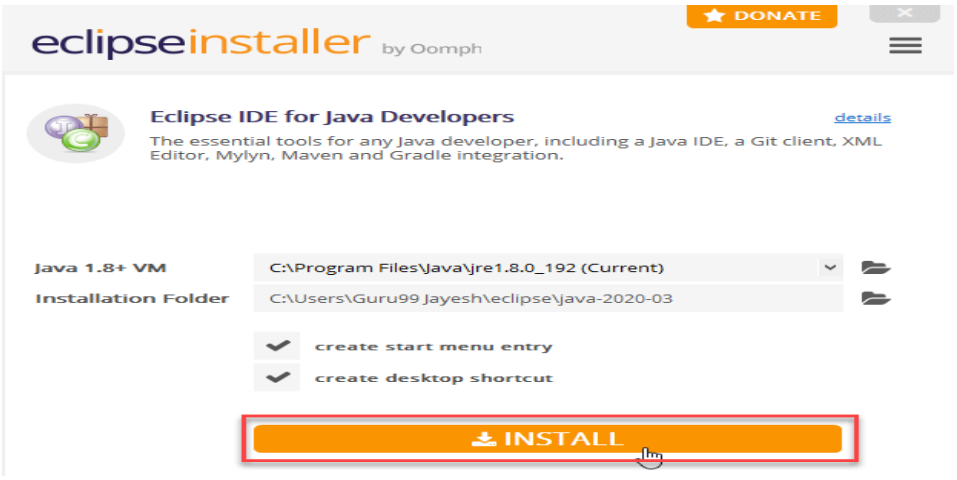


Step 6) Click on “Eclipse IDE for Java Developers”



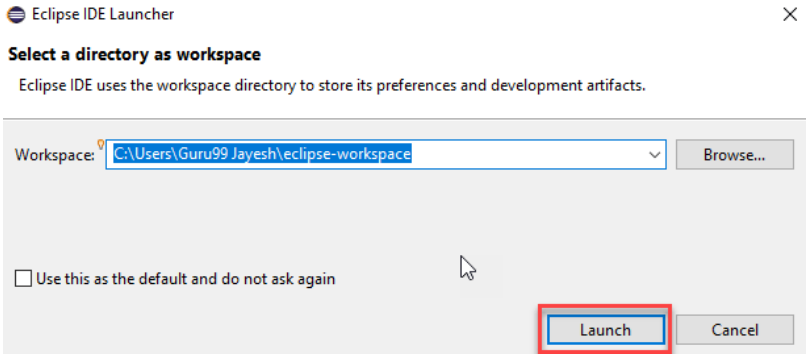
Step 7) Click on “INSTALL” button

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			



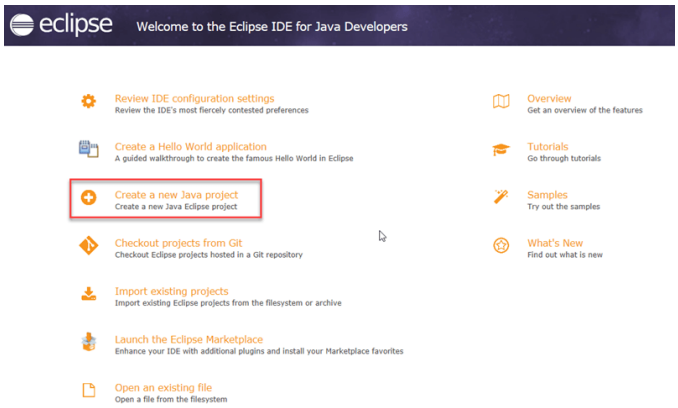
Step 8) Click on "LAUNCH" button.

Step 9) Click on "Launch" button.

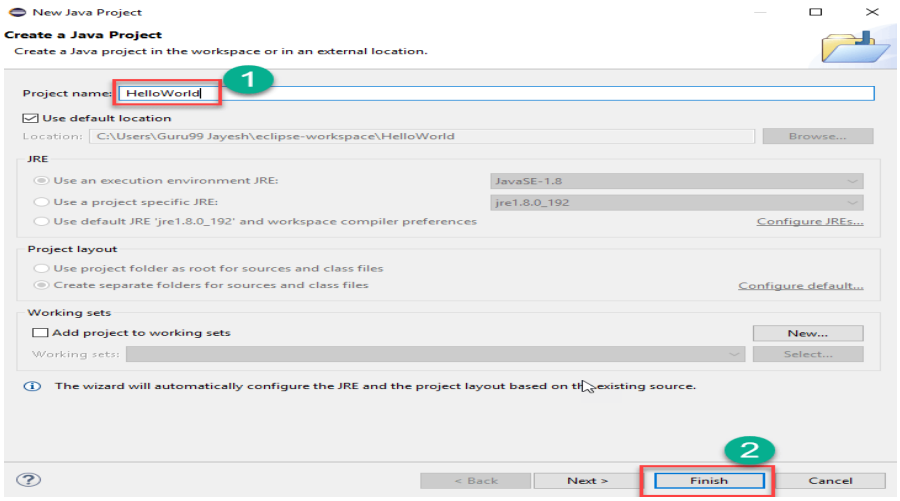


Step 10) Click on "Create a new Java project" link.

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

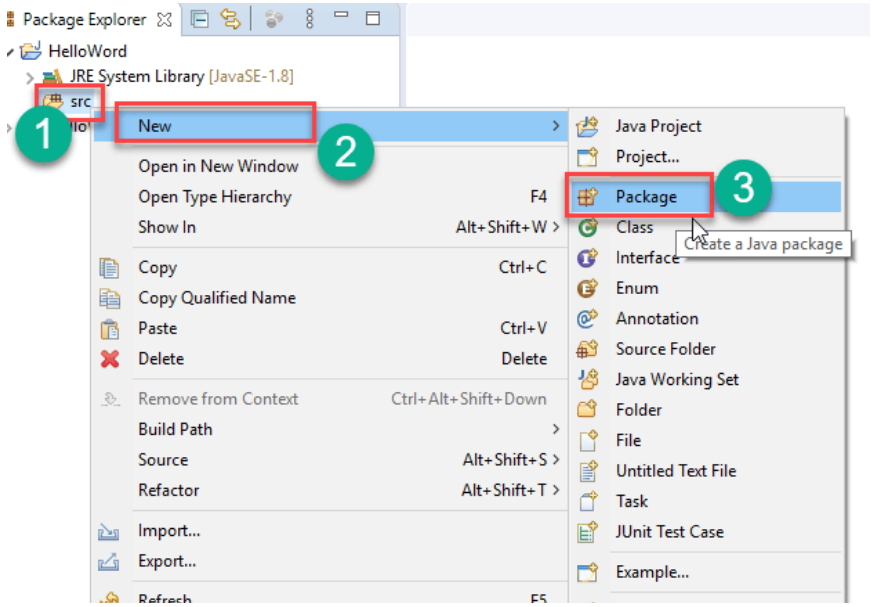


Step 11) Create a new Java Project
Write project name.
Click on “Finish button”.

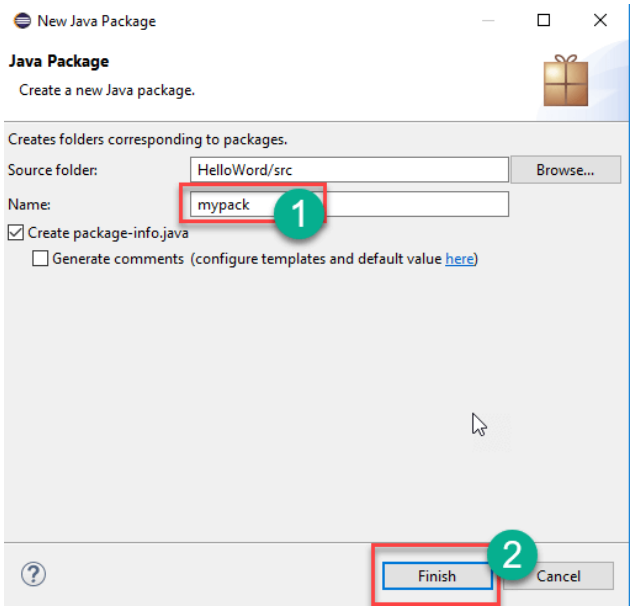


Step 12) Create Java Package.
Goto “src”.
Click on “New”.
Click on “Package”.

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

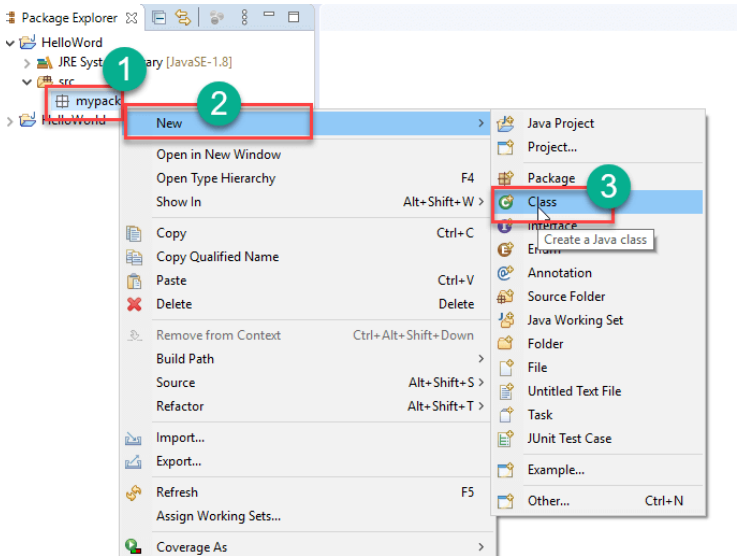


Step 13) Writing package name.
Write name of the package
Click on Finish button.



Step 14) Creating Java Class
Click on package you have created.
Click on “New”.
Click on “Class”.

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

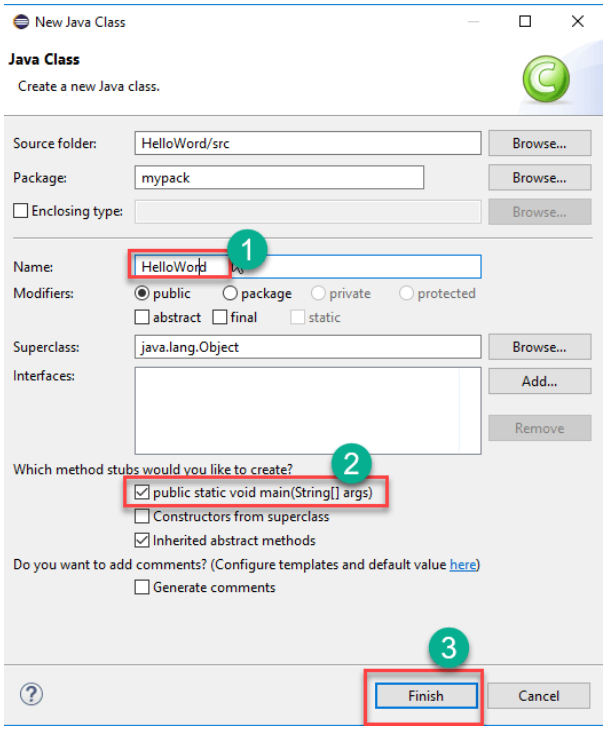


Step 15) Defining Java Class.

Write class name

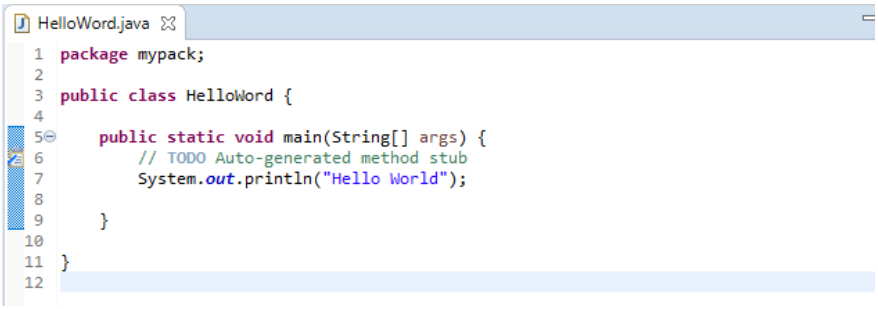
Click on “public static void main (String[] args)” checkbox.

Click on “Finish” button.

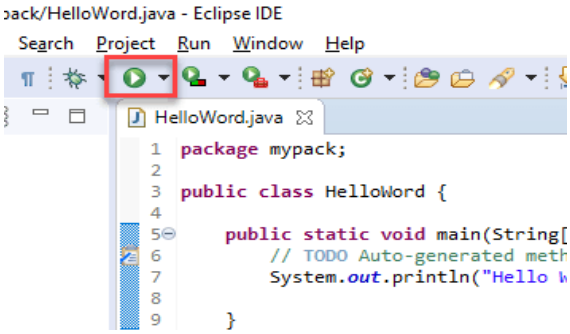


Helloword.java file will be created as shown below:

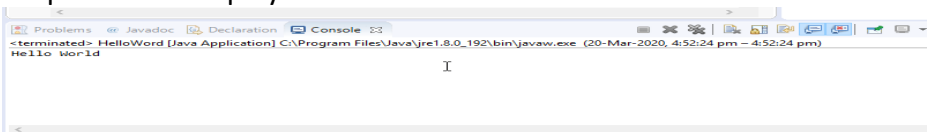
Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			



Step 16) Click on "Run" button.



Output will be displayed as shown below.



2.Aim: To write a JAVA program to display default value of all primitive data type of JAVA

Program:

```
class DefaultDemo
{
static byte b;
static short s;
static int i;
static long l;
static float f;
static double d;
static char c;
static boolean bl;
public static void main(String[] args)
{
System.out.println("The default values of primitive data types are:");
System.out.println("Byte :"+b);
```

```
System.out.println("Short :"+s);
System.out.println("Int :"+i);
System.out.println("Long :"+l);
System.out.println("Float :"+f);
System.out.println("Double :"+d);
System.out.println("Char :"+c);
System.out.println("Boolean :"+bl);
}
}
```

Output:

```
The default values of primitive data types are:
Byte :0
Short :0
Int :0
Long :0
Float :0.0
Double :0.0
Char :
String :null
Boolean :false
```

3.Write a java program to find prime numbers between 1 to n.

```
import java.util.*;
class PrimeExample{
        public static void main(String arg[]){
        int i,count;
        System.out.print("Enter n value : ");
        Scanner sc=new Scanner(System.in);
         int n=sc.nextInt();
        System.out.println("Prime numbers between 1 to "+n+" are ");
                for(int j=2;j<=n;j++){
                count=0;
                        for(i=1;i<=j;i++){
                                if(j%i==0){
                                count++;
                                }
                        }
```

```
                if(count==2)
                System.out.print(j+" ");
                }

        }
}
```

Sample Input/output:

```
E:\JP>javac PrimeExample.java
E:\JP>java  PrimeExample
Enter n value: 100
Prime numbers between 1 to 100 are

3  5  7  11  13  17  19  23  29  31  37  41  43  47 53  59  61  67  71  73  79  83  89  97
```

4.Write a Java program that prints all real solutions to the quadratic equation ax2+bx+c=0. Read in a, b, c and use the quadratic formula.

```
import java.util.*;
class Roots{
        public static void main(String args[]){
        int a,b,c,d,f=0;
        Scanner scr=new Scanner(System.in);
        System.out.println("\nEnter the values of a ,b ,c : ");
        a=scr.nextInt();
        b=scr.nextInt();
        c=scr.nextInt();
        d=(b*b)-(4*a*c);
                if(d==0){
                        System.out.println("Roots are real and Equal");
                        f=1;
                }
                        else if(d>0){
                                System.out.println("Roots are real and UnEqual");
                                f=1;
                        }
        else
                System.out.println("Roots are imaginary");
                if(f==1){
                        float r1=(float)(-b+Math.sqrt(d))/(2*a);
```

```
                float r2=(float)(-b-Math.sqrt(d))/(2*a);
                System.out.println("Roots are : "+r1+" ,"+r2);
            }

        }
}
```

Sample Input/output:

E:\JP>javac Roots.javaE:\JP>java Roots
Enter the values of a ,b ,c : 1 2 -3

Roots are real and Unequal   Roots are: 1.0 ,-3.0

5. Aim: To write a JAVA program to implement class mechanism. – Create a class, methods and invoke them inside main method

Programs:

```
class A
{
int l=10,b=20;
        void display()
        {
        System.out.println(l);
        System.out.println(b);
        }
}
class methoddemo{
        public static void main(String args[]){
                A a1=new A();
                a1.display();
 }
}
```

Output:
10
20

6. Write a java program that reads a line of integer, and then displays and then displays each integer and sum of all integers(Using StringTokenizer class of java.util).

Solution:

```
import java.util.Scanner;
```

```
import java.util.StringTokenizer;
public class StrTokenDemo {
public static void main(String[] args) {
Scanner sin = new Scanner(System.in);
System.out.print("Enterthe integer string(space as delimiter) : ");
String IntegerString = sin.nextLine();
int sum = 0;
StringTokenizer st = new StringTokenizer(IntegerString, " ");
while (st.hasMoreTokens()){
int val = Integer.parseInt(st.nextToken());
sum = sum + val;
System.out.println(val);
}
System.out.println("Sum : "+sum);
}
}
```

Output:

```
Enter the integer string(space as delimiter) : 1 2 3 4 5
1
2
3
4
5
Sum : 15
```

7. Write a Java program to multiply two given matrices

Aim: Write a Java program to multiply two Matrices with an example.
Or write a Java program to perform multiplication of two multidimensional arrays.

Description:

In this Java multiply two Matrices program, we declared two integer matrixes. Next, we used the For Loop to iterate those matrix values. We performed matrix multiplication on i and j matrixes within that loop and assigned it to another matrix called multi. Later, we used another for loop to print the final matrix.

```
import java.util.Scanner;
public class MultiplyTwoMatrix {
```

```
private static Scanner sc;
public static void main(String[] args) {
int i, j, rows, columns;
Scanner sc= new Scanner(System.in);

System.out.println("\n Enter Multiplication Matrix Rows & Columns : ");
rows = sc.nextInt();
columns = sc.nextInt();

int[][] arr1 = new int[rows][columns];
int[][] arr2 = new int[rows][columns];
System.out.println("\n Enter the First Multiplication Matrix Items : ");

for(i = 0; i < rows; i++) {
for(j = 0; j < columns; j++) {
arr1[i][j] = sc.nextInt();
}
}

System.out.println("\n Enter the Second Multiplication Matrix Items : ");
for(i = 0; i < rows; i++) {
for(j = 0; j < columns; j++) {
arr2[i][j] = sc.nextInt();
}
}

System.out.println("\n-----The Multiplication of two Matrix");
for(i = 0; i < rows; i++) {
for(j = 0; j < columns; j++) {
System.out.format("%d \t", (arr1[i][j] * arr2[i][j]));
}
System.out.println("");
}
}
}
```

Sample Input/output:

```
E:\>javac MultiplyTwoMatrix.java
```

```
E:\JP>java MultiplyTwoMatrix
Enter Multiplication Matrix Rows & Columns : 3 3
Enter the First Multiplication Matrix Items : 1 2 3
4 5 6
7 8 9
Enter the Second Multiplication Matrix Items : 10 20 30
40 50 60
70 80 90
-----The Multiplication of two Matrixes-----
10      40      90
160     250     360
490     640     810
```

8. Write a java program to illustrate the concept of Single level and multi level inheritance

```
class One{
String one_str = "class One";
void displayOne(){
System.out.println("parent " + one_str);
}
}
class Two extends One{
String two_str = "class Two";
void displayTwo(){
super.displayOne();
System.out.printf("child %s inherited from %s displaying single level
inheritance\n",two_str,one_str);
}
}
class Three extends Two{
String three_str = "class Three";
void displayThree(){
super.displayTwo();
System.out.printf("child %s inherited from %s and %s displaying
        multilevelinheritancen\n",three_str,two_str,one_str);
}
}
public class InheritanceDemo {
public static void main(String[] args) {
```

```
Three threeobj = new Three();
 threeobj.displayThree();
}
}
```

Output:

parent class One
child class Two inherited from class One displayingsingle level inheritance
child class Three inherited from class Two andclass One displaying multilevel inheritance

9. Aim: To write a JAVA program to implement constructor overloading

Program:

```
class A{
int l,b;
        A()     {
        l=10; b=20;
        }
                A(int u,int v){
                l=u; b=v;
                }
        int area(){
        return l*b;
        }
}
class OverConstructDemo
{
        public static void main(String args[])
        {
        A a1=new A();
        int r1=a1.area();
        System.out.println("The area is: "+r1);
        A a2=new A(30,40);
        int r2=a2.area();
        System.out.println("The area is: "+r2);
        }
}
```

Output:
The area is: 200 The area is: 1200

10. Develop a Java application to generate Electricity bill. Create a class with the following members:

Consumer no., consumer name, previous month reading, current month reading, type of EB connection (i.e. domestic or commercial). Commute the bill amount using the following tariff.

If the type of the EB connection is domestic, calculate the amount to be paid as follows:

First 100 units - Rs. 1 per unit; 101-200 units - Rs. 2.50 per unit; 201 -500 units - Rs. 4 per unit;

501 units - Rs. 6 per unit

If the type of the EB connection is commercial, calculate the amount to be paid as follows: First 100 units - Rs. 2 per unit; 101-200 units - Rs. 4.50 per unit; 201 -500 units - Rs. 6 per unit; 501 units - Rs. 7 per unit.

Aim: Develop a Java application to generate Electricity bill.

Description:

Create a class with the following members

Consumer no., consumer name, previous month reading, current month reading, type of EB Connection (i.e. domestic or commercial)

Compute the bill amount using the following tariff.
program:

```
import java.util.Scanner;class ElectBill
{

int  ConsumerNo;
String ConsumerName;
int PrevReading;
int CurrReading;
String EBConn;
```

```
double Bill;
void input_data()

{

Scanner sc = new Scanner(System.in);
 System.out.println("\n Enter Consumer Number: ");
ConsumerNo = sc.nextInt();
System.out.println("\n Enter Consumer Name: ");
ConsumerName = sc.next();
 System.out.println("\n Enter Previous Units: ");
PrevReading = sc.nextInt();
System.out.println("Enter Current Units consumed:");
CurrReading = sc.nextInt();
System.out.println("Enter the types of EB Connection(domestic or commercial)");
EBConn = sc.next();
}

double calculate_bill()
{

int choice;
if(EBConn=="domenstic")
choice=1;
else choice=2;
switch(choice)

{

case 1:

if(CurrReading>=0 && CurrReading<=100)
Bill=CurrReading*1;
else if(CurrReading>100 && CurrReading <= 200)
Bill=(100*1)+((CurrReading-100)*2.50);
else if(CurrReading>200 && CurrReading <= 500)
Bill=(100*1)+(100*2.50)+((CurrReading-200)*4);
 else
Bill=(100*1)+(100*2.50)+(300*4)+((CurrReading-500)*6);
```

```
break;
case 2:
if(CurrReading>=0 && CurrReading<=100)
Bill=CurrReading*2;
else if(CurrReading>100 && CurrReading <= 200)
Bill=(100*1)+((CurrReading-100)*4.50);
else if(CurrReading>200 && CurrReading <= 500)
Bill=(100*1)+(100*2.50)+((CurrReading-200)*6);
else
Bill=(100*1)+(100*2.50)+(300*4)+((CurrReading-500)*7);
break;

}
return Bill;
}

void display()

{

System.out.println("....");
System.out.println("ELCTRICITY BILL");
System.out.println("....");
System.out.println("Consumer Number: "+ConsumerNo);
System.out.println("Consumer Name: "+ConsumerName);
System.out.println("Consumer Previous Units: "+PrevReading);
System.out.println("Consumer Current Units: "+CurrReading);
System.out.println("Type of EBConnection: "+EBConn);
System.out.println("....");
System.out.println("Total Amount(Rs.): "+Bill);
}

}

class ElectBillGen

{
```

```
public static void main (String[] args)

{

ElectBill b=new ElectBill();
b.input_data();
b.calculate_bill();
b.display();
}

}
```

Sample Input/Output:

```
E:\JP>javac  ElectBillGen.java
E:\JP>java ElectBillGen
Enter Consumer Number:
102
Enter Consumer Name:
anil
Enter Previous Units:
310
Enter Current Units consumed:
480
Enter the types of EB Connection(domestic or commercial)domestic
----------------------

ELCTRICITY BILL
----------------------

Consumer Number: 102
Consumer Name: anil
Consumer Previous Units: 310
Consumer Current Units: 480
Type of EBConnection: domestic
----------------------

Total Amount(Rs.): 2030.0
```

11. Aim: Write a JAVA program give example for “super” keyword

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

Programs:

(i) Using super to call super class constructor (Without parameters)

```
class A{
int l,b;
        A(){
        l=10; b=20;
        }
}
class B extends A{
int h;
        B(){
        super();
        h=30;
        }
                int volume(){
                return l*b*h;
                }
}
class superdemo{
        public static void main(String args[]){
        B b1=new B();
        int r=b1.volume();
        System.out.println("The vol. is: "+r);
        }
}
```

Output:
The vol. is:6000

(ii) Using super to call super class constructor (With parameters)

```
class A{
int l,b;
                A(int u,int v){
                l=u; b=v;
                }
}
```

```
class B extends A{
int h;
                B(int u,int v,int w){
                super(u,v); h=w;
                }
        int volume(){
        return l*b*h;
        }
}
class superdemo{
        public static void main(String args[]){
        B b1=new B(30,20,30);
        int r=b1.volume();
        System.out.println("The vol. is: "+r);
        }
}
```

Output:
The vol. is:18000

12.Create a base class Fruit which has name ,taste and size as its attributes. A method called eat() is created which describes the name of the fruit and its taste. Inherit the same in 2 other class Apple and Orange and override the eat() method to represent each fruit taste.

Fruit.java

```
package com.basics;
public class Fruit {
        protected String name;
        protected String taste;
        protected int size;

        public Fruit() {
                name = "Fruit name";
                taste = "Tase of the fruit";
                size = 0;
        }
```

```
	public void eat () {
		System.out.println(name + " tastes like " + taste);
	}
}
```

Apple.java

```
package com.basics;
public class Apple extends Fruit {
	@Override
	public void eat() {
		System.out.println("It tastes like apple");
	}
}
```

Orange.java

```
package com.basics;
public class Orange extends Fruit {
	@Override
	public void eat() {
		System.out.println("It tastes like Orange");
	}
}
```

FruitImpl.java

```
package com.basics;
public class FruitImpl {
	public static void main(String[] args) {
		new Fruit().eat();
		new Apple().eat();
		new Orange().eat();

	}
}
```

Output:

```
Java FruitImpl.java
Java FruitImpl
Fruit name tastes like Tase of the fruit
It tastes like apple
It tastes like Orange
```

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

13. Write a program to create a class named shape. It should contain 2 methods, draw() and erase() that prints —Drawing Shape‖ and —Erasing Shape‖ respectively. For this class, create three sub classes, Circle, Triangle and Square and each class should override the parent class functions - draw () and erase (). The draw() method should print —Drawing Circle‖, —Drawing Triangle‖ and —Drawing Square‖ respectively. The erase() method should print —Erasing Circle‖, —Erasing Triangle‖ and —Erasing Square‖ respectively. Create objects of Circle, Triangle and Square in the following way and observe the polymorphic nature of the class by calling draw() and erase() method using each object. Shape c=new Circle(); Shape t=new Triangle(); Shape s=new Square();

Shape.java

```
public class Shape {
		public void draw() {
		System.out.println("Drawing Shape");
	}
	public void erase() {
		System.out.println("Erasing Shape");
	}
}
```

Circle.java

```
public class Circle extends Shape {
	@Override
	public void draw() {
		System.out.println("Drawing Circle");
	}

	@Override
	public void erase() {
		System.out.println("Erasing Circle");
	}
}
```

Square.java

```
public class Square extends Shape {
	@Override
	public void draw() {
		System.out.println("Drawing Square");
	}
```

```
	@Override
	public void erase() {
		System.out.println("Erasing Square");
	}
}
```

Triangle.java

```
public class Triangle extends Shape {
	@Override
	public void draw() {
		System.out.println("Drawing Triangle");
	}

	@Override
	public void erase() {
		System.out.println("Erasing Triangle");
	}
}
```

ShapeImpl.java

```
public class ShapeImpl {

	public static void main(String[] args) {
		Shape c = new Circle();
		Shape t = new Triangle();
		Shape s = new Square();

		c.draw();
		c.erase();

		t.draw();
		t.erase();

		s.draw();
		s.erase();
	}

}
```

Output:

```
Drawing Circle
Erasing Circle
Drawing Triangle
Erasing Triangle
Drawing Square
Erasing Square
```

14.Write a java program to illustrate the concept of class with method overloading.

Solution:

```
class Sum {
int sum(int x, int y)
{       return (x + y);  }
int sum(int x, int y, int z)
{       return (x + y + z);       }
 double sum(double x, double y)
{       return (x + y);  }
}
public class MethodOverloadingDemo {
public static void main(String args[])
{
Sum s = new Sum();
System.out.println(s.sum(10, 20));
System.out.println(s.sum(10, 20, 30));
System.out.println(s.sum(10.5, 20.5));
}
}
```

Output:

```
30
60
31.0
```

15. Write Java program(s) on use of inheritance, preventing inheritance using final, abstract classes.

Aim: To write a java program on use of inheritance, preventing inheritance using final, abstract classes.

Description:

The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of

the parent class. final is a keyword in java used for restricting some functionalities. We can declare variables, methods and classes with final keyword. A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).

Inheritance:

```
class Parent
{
public void p1()
{
System.out.println("Parent method");
}
}

public class Child extends Parent {
        public void c1()
        {
        System.out.println("Child method");
        }
        public static void main(String[] args)
        {
        Child cobj = new Child(); cobj.c1(); //method of Child class

        cobj.p1(); //method of Parent class
        }
}
```

Sample Input/output:
Child method Parent method

preventing inheritance using final:

```
// create a final class
final class FinalClass {
        public void display() {
        System.out.println("This is a final method.");
        }
}
```

```
// try to extend the final class
class Main extends FinalClass {
        public void display() {
        System.out.println("The final method is overridden.");
        }

        public static void main(String[] args) {
        Main obj = new Main(); obj.display();
        }
}
```

Sample Input/output:
Compile Time Error

Abstract Classes:

```
abstract class A{
        abstract void callme();
}
classB extends A{
        voidcallme(){
        System.out.println("this is callme.");
        }
        public static void main(String[] args){
        B b = new B(); b.callme();
        }
}
```

Sample Input/output:
this is callme

16. Write a java program to demonstrate interfaces and abstract classes.

```
abstract class Demo{
abstract void show();
}
interface Printable
{
void print();
}
```

```
class AbstractClassdemo extends Demo implements Printable{
void show() {
System.out.println("This is Abstract class method");
}
public void print() {
System.out.println("This is interface method");
}
public static void main(String [] args)
{
AbstractClassdemo a= new AbstractClassdemo();a.show();
a.print();
}
}
```

Output:
This is Abstract class methodThis is Interface method

17. Write Java program(s) on dynamic binding, differentiating method overloading and overriding.

Aim: To write a java program perform dynamic binding and Method overriding concept and overloading Methods.

Description:

Dynamic binding also called dynamic dispatch is the process of linking procedure call to a specific sequence of code (method) at run-time. Dynamic binding is an object oriented programming concept and it is related with polymorphism and inheritance.
If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.

If a class has multiple methods having same name but different in parameters, it is known as method overloading in Java.

Dynamic Binding:

```
class A{
        void samp(){
        System.out.println("hai...");
        }
}
```

```
class D extends A{
        voidsamp(){
        System.out.println("hello...");
        }

        public static void main(String args[]){
        A a=new D();
        a.samp();
        }
}
```

Sample Input/Output:

Hello

Overloading Methods:

```
class Overload{
        void demo (int a)
        {
        System.out.println ("a: " + a);
        }

        void demo (int a, int b){
        System.out.println ("a and b: " + a + "," + b);
        }

        double demo(double a) {
        System.out.println("double a: " + a); return a*a;
        }
}

classMethodOverloading{
        public static void main (String args [])
        {
        Overload Obj = new Overload(); double result;
        Obj .demo(10);
        Obj .demo(10, 20); result = Obj .demo(5.5);
```

```
        System.out.println("Output is : " + result);
        }
}
```

Sample Input/output:
a: 10
a and b: 10,20 double a: 5.5

Output is : 30.25

Method Overriding:

```
class BaseClass{
        public void methodToOverride() //Base class method
        {
        System.out.println ("I'm the method of BaseClass");
        }
}

class DerivedClass extends BaseClass{
        public void methodToOverride() //Derived Class method
        {
        System.out.println ("I'm the method of DerivedClass");
        }
}

class TestMethod{
public static void main (String args []) {
// BaseClass reference and object BaseClass obj1 = new BaseClass();
// BaseClass reference but DerivedClass object BaseClass obj2 = new DerivedClass();
// Calls the method from BaseClass class obj1.methodToOverride();
//Calls the method from DerivedClass class obj2.methodToOverride();
}

}
```

Sample Input/output:

I'm the method of BaseClass I'm the method of DerivedClass

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

18. Aim: To write a JAVA program that implements Runtime polymorphism

```
class Bank{
float getRateOfInterest(){return 0;}
}
class SBI extends Bank{
float getRateOfInterest(){return 8.4f;}
}
class ICICI extends Bank{
float getRateOfInterest(){return 7.3f;}
}
class AXIS extends Bank{
float getRateOfInterest(){return 9.7f;}
}
class TestPolymorphism{
public static void main(String args[]){
Bank b;
b=new SBI();
System.out.println("SBI Rate of Interest: "+b.getRateOfInterest());
b=new ICICI();
System.out.println("ICICI Rate of Interest: "+b.getRateOfInterest());
b=new AXIS();
System.out.println("AXIS Rate of Interest: "+b.getRateOfInterest());
}
}
```

Output:
SBI Rate of Interest: 8.4
ICICI Rate of Interest: 7.3
AXIS Rate of Interest: 9.7

Example2:

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void eat(){System.out.println("eating bread...");}
}
class Cat extends Animal{
```

```
void eat(){System.out.println("eating rat...");}
}
class Lion extends Animal{
void eat(){System.out.println("eating meat...");}
}
class TestPolymorphism3{
public static void main(String[] args){
Animal a;
a=new Dog();
a.eat();
a=new Cat();
a.eat();
a=new Lion();
a.eat();
}}
```

Output

```
eating bread...
eating rat...
eating meat...
```

19. Develop a java application to implement currency converter (Dollar to INR, EURO to INR, Yen) using Interfaces.

Aim: To develop a java application to implement currency converter using Interfaces.

Description:

An Interface in Java programming language is defined as an abstract type used to specify the behavior of a class. A Java interface contains static constants and abstract methods.
A class can implement multiple interfaces. In Java, interfaces are declared using the interface keyword. All methods in the interface are implicitly public and abstract.

Step 1:

Create a folder named Conversion Demo. This is the name of the package. Following Java files are stored in this folder.
Currency.java

```
package ConversionDemo;
public interface Currency {
public void converter();
```

```
}
```

Step2: Create a class dollarTOINR
DollarTOINR.java

```
import java.util.Scanner;

public class DollarToINR implements Currency{
@Override
public void converter() {
System.out.println("Enter the Dollar Price");
Scanner scanner = new Scanner(System.in);
int currency = scanner.nextInt();
System.out.printf("The INR price is %.2f", (double) currency*75);
}
}
```

Step3: Create a class euroTOINR
EuroTOINR.java

```
import java.util.Scanner;
public class EuroToINR implements Currency{
@Override
public void converter() {
System.out.println("Enter the EURO Price");
Scanner scanner = new Scanner(System.in);
int currency = scanner.nextInt();
System.out.printf("The INR price is %.2f" , (double) currency*88);
}
}
```

Step4: Create a class INRTODollar
INRTODollar.java

```
import java.util.Scanner;
public class INRToDollar implements Currency{
@Override
public void converter() {
System.out.println("Enter the INR Price");
Scanner scanner = new Scanner(System.in);
int currency = scanner.nextInt();
System.out.printf("The Dollar price is %.2f", (double)currency/75);
```

```
}
}
```

Step5: Create a class INRTOEuro

INRTOEuro.java

```
import java.util.Scanner;
public class INRToEuro implements Currency{
@Override
public void converter() {
System.out.println("Enter the INR Price");
Scanner scanner = new Scanner(System.in);
int currency = scanner.nextInt();
System.out.printf("The EURO price is %.2f" , (double) currency/88);
}
}
```

Step6: Create a class INRToYen

INRToYen.java

```
import java.util.Scanner;
public class INRToYen implements Currency{
@Override
public void converter() {
System.out.println("Enter the INR Price");
Scanner scanner = new Scanner(System.in);
int currency = scanner.nextInt();
System.out.printf("The Yen price is %.2f" , (double) currency*1.50);
}
}
```

Step7: create a class YenToINR

YenToINR.java

```
import java.util.Scanner;
public class YenToINR implements Currency{
@Override
public void converter() {
System.out.println(“Enter the Yen Price”);
Scanner scanner = new Scanner(System.in);
int currency = scanner.nextInt();
System.out.printf(“The INR price is %.2f” , (double) currency*0.67);
}
```

```
}
```

Step 8: Final Create a Main class to execute all class.
Main.java

```
import java.util.Scanner;
public class Main {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
while (true) {
System.out.println("\n Menu For Currency Conversion");
System.out.println("1. Dollar to INR");
System.out.println("2. INR to Dollar");
System.out.println("3. Euro to INR");
System.out.println("4. INR to Euro");
System.out.println("5. Yen to INR");
System.out.println("6. INR to Yen");
System.out.println("7. Exit");
System.out.println("Enter your choice: ");
int choice = scanner.nextInt();
switch (choice) {
case 1:
        new dollarToINR().converter(); break;
case 2:
        new INRToDollar().converter(); break;
case 3:
        new euroToINR().converter();break;
case 4:
        new INRToEuro().converter(); break;
case 5:
        new YenToINR().converter(); break;
case 6:
        new INRToYen().converter(); break;
case 7:
        System.exit(0);
default:
        System.out.println("Please select from above list");
}//switch close
}// while close
}//main method close
```

}//Main class close

Sample Input/Output:
Menu For Currency Conversion
1. Dollar to INR
2. INR to Dollar
3. Euro to INR
4. INR to Euro
5. Yen to INR
6. INR to Yen
7. Exit
Enter your choice:
2
Enter the INR Price 50000
The Dollar price is 666.67 Menu For Currency Conversion

1. Dollar to INR
2. INR to Dollar
3. Euro to INR
4. INR to Euro
5. Yen to INR
6. INR to Yen
7. Exit
Enter your choice:
1

Enter the Dollar Price 666
The INR price is 49950.00

20. Aim: To write a JAVA program to implement Interface.
Programs:
(i) First form of interface implementation

```
interface A{
        void display();
}
class B implements A{
        public void display(){
        System.out.println("B's method");
```

```
        }
}
class C extends B{
        public void callme(){
        System.out.println("C's method");
        }
}
class InterfaceDemo{
        public static void main(String args[]){
        C c1=new C();
        c1.display();
        c1.callme();
        }
}
```

Output:
B's method C's method

(ii) Second form of interface implementation

```
interface D{
        void display();
}
interface E extends D{
        void show();
}
class A{
        void callme(){
                System.out.println("This is in callme method");
        }
}

class B extends A implements E{
        public void display(){
                System.out.println("This is in display method");
        }
                public void show(){
                        System.out.println("This is in show method");
                }
}
```

```
class C extends B{
        void call(){
                System.out.println("This is in call method");
        }
}
class interfacedemo{
        public static void main(String args[]){
                C c1=new C();
                c1.display();
                c1.show();
                c1.callme();
                c1.call();
        }
}
```

Output:

This is in display method This is in show method This is in callme method This is in call method

(iii) Third form of interface implementation

```
interface A{
        void display();
}
class B implements A{
        public void display(){
                        System.out.println("This is in B's method");
        }
}
class C implements A{
        public void display(){
                System.out.println("This is C's method");
        }
}
class interfacedemo{
        public static void main(String args[]){
                B b1=new B();
                C c1=new C();
                b1.display();
                c1.display();
        }
}
```

Output:

This is in B's method This is C's method

(iv) Fourth form of interface implementation

```
interface A{
        void display();
}
interface B{
        void callme();
}
interface C extends A,B{
        void call();
}
class D implements C{
        public void display(){
                System.out.println("interface A");
        }
                public void callme(){
                        System.out.println("interface B");
                }
        public void call(){
                System.out.println("interface C");
        }
}
class InterfaceDemo{
        public static void main(String args[]){
                D d1=new D();
                d1.display();
                d1.callme();
                d1.call();
        }
}
```

Output: interface A interface B interface C

21. Aim: To write a JAVA program that describes exception handling mechanism

Program:

Usage of Exception Handling:

```
class trydemo{
        public static void main(String args[]){
```

```
        try{

                int a=10,b=0;
                int c=a/b;
                System.out.println(c);
        }
                catch(ArithmeticException e){
                System.out.println(e);
                }
                System.out.println("After the catch statement");
        }
}
```

Output:
java.lang.ArithmeticException: / by zero After the catch statement

22. Aim: To write a JAVA program Illustrating Multiple catch clauses

```
class MultitryDemo{
        public static void main(String args[]){
                try{
                        int a=10,b=5;
                        int c=a/b;
                        int d[]={0,1};
                        System.out.println(d[10]);
                        System.out.println(c);
                }

        catch(ArithmeticException e){
                System.out.println(e);
        }
        catch(ArrayIndexOutOfBoundsException e){
                System.out.println(e);
        }
                System.out.println("After the catch statement");
        }
}
```

Output: java.lang.ArrayIndexOutOfBoundsException: 10 After the catch statement

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

23. Aim: To write a JAVA program for creation of Illustrating throw

```
class ThrowDemo{
public static void main(String args[]){

try{
throw new NullPointerException("demo");
}
catch(NullPointerException e){
System.out.println(e);
}
}
}
```

Output:
java.lang.NullPointerException: demo

24. Aim: To write a JAVA program for creation of Java Built-in Exceptions Programs:

(i) Arithmetic exception

```
class ArithmeticDemo{
public static void main(String args[]){
 try{
int a = 10, b = 0;
int c = a/b;
System.out.println (c);
}
 catch(ArithmeticException e){
System.out.println (e);
}
}
}
```

Output:
java.lang.ArithmeticException: / by zero

(ii) NullPointer Exception

```
class NullpointerDemo{
public static void main(String args[]){
try{
        String a = null;
```

```
        System.out.println(a.charAt(0));
 }
catch(NullPointerException e)
{
        System.out.println(e);
}
}
}
```

Output:
java.lang.NullPointerException

(iii) StringIndexOutOfBound Exception

```
class StringBoundDemo{
public static void main(String args[]){
try{
 String a = "This is like chipping ";
 char c = a.charAt(24);
 System.out.println(c);
}
catch(StringIndexOutOfBoundsException e){
        System.out.println(e);
}
}
}
```

Output:
java.lang.StringIndexOutOfBoundsException: String index out of range: 24

(iv) FileNotFound Exception

```
import java.io.*;
class FileNotFoundDemo{
public static void main(String args[]){
 try{
File file = new File("E://file.txt");
FileReader fr = new FileReader(file);
}
catch (FileNotFoundException e){
System.out.println(e);
}
```

```
}
}
```

Output:

java.io.FileNotFoundException: E:\file.txt (The system cannot find the file specified)

(v) NumberFormat Exception

```
class NumberFormatDemo{
public static void main(String args[]){
try{
int num = Integer.parseInt ("akki") ;
System.out.println(num);
 }
catch(NumberFormatException e){
System.out.println(e);
}
}
}
```

Output:

java.lang.NumberFormatException: For input string: "akki"

(vi) ArrayIndexOutOfBounds Exception

```
class ArrayBoundDemo{
public static void main(String args[]){
try{
int a[] = new int[5];
a[6] = 9;
}
catch(ArrayIndexOutOfBoundsException e){
System.out.println (e);
}
}
}
```

Output:

java.lang.ArrayIndexOutOfBoundsException: 6

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

25. To write a JAVA program for creation of User Defined Exception Program:

```
class A extends Exception{
A(String s1){
super(s1);
}
}
class owndemo{
public static void main(String args[]){
 try{
        throw new A("demo ");
 }
catch(Exception e){
System.out.println(e);
}
}
}
```

Output:

A: demo

26 a). Write a Java Program to demonstrate the following String Handlings.

i. **String Length& Concatenation.**
ii. **Character Extraction.**
iii. **String Comparison.**
iv. **Searching and modifying String.**

//String Length& Concatenation.

```
public class LengthConcatExample{
public static void main(String args[]){
String s1="java string";
//Before apply concat
System.out.println("Before apply concat");
System.out.println("==========================================");
System.out.println("s1 length=" + s1.length()+" s1 contains = "+ s1);
//after concat but not assign result in s1
System.out.println("\nafter concat but not assign result in s1");
System.out.println("==========================================");
s1.concat("is immutable");
```

```
System.out.println("s1 length=" + s1.length()+" s1 contains = "+ s1);
//after concat and assign result in s1
System.out.println("\nafter concat and assign result in s1");
System.out.println("==========================================");
s1=s1.concat(" is immutable so assign it explicitly");
System.out.println("s1 length=" + s1.length()+" s1 contains = "+ s1);
}
}
```

Output:

```
Before apply concat
==========================================
s1 length=11 s1 contains = java string

after concat but not assign result in s1
==========================================
s1 length=11 s1 contains = java string

after concat and assign result in s1
==========================================
s1 length=48 s1 contains = java string is immutable so assign it explicitly
```

//Character Extraction.

// implement to display first character of each word in a string

```
package java_fundamentals.language_basics;

public class FirstCharacterInEachWord {
	public static void main(String[] args) {
		String str = "Annamacharya Institute of Technology and Sciences";
	  String[] word= str.split("\\s");

	  System.out.println("The first character of each word: ");
	  for(String word1:word) {
	    // Logic to implement first character of each word in a string
	    System.out.println(word1.charAt(0));
	  }
	}
}
```

Output:

```
A
I
```

o
T
a
S

//String Comparison
// Java program to demonstrate
// use of .compareTo operator in Java

```
class GFG {
	public static void main(String[] args)
	{

		// Get some Strings to compare
		String s1 = "A";
		String s2 = "A";
		String s3 = "a";
		String s4 = new String("A");

		// Compare s1 and s2
		// It should return 0 as they both
		// have the same ASCII value
		System.out.println(s1 + " .compareTo " + s2
						+ ": " + s1.compareTo(s2));

		// Compare s1 and s3
		// It should return -32 as they both
		// have the different ASCII value
		System.out.println(s1 + " .compareTo " + s3
						+ ": " + s1.compareTo(s3));

		// Compare s3 and s2
		// It should return 32 as they both
		// have the different ASCII value
		System.out.println(s3 + " .compareTo " + s2
						+ ": " + s3.compareTo(s2));

		// Compare s1 and s4
		// It should return 0 as they both
		// have the same ASCII value
```

```
                System.out.println(s1 + " .compareTo " + s4
                                                + ": " + s1.compareTo(s4));
        }
}
```

Output:

```
A .compareTo A: 0
A .compareTo a: -32
a .compareTo A: 32
A .compareTo A: 0
```

//Searching and modifying String

//You can specify a starting point for the search using these forms:

```
int indexOf(int ch, int startIndex)
int lastIndexOf(int ch, int startIndex)
int indexOf(String str, int startIndex)
int lastIndexOf(String str, int startIndex)
```

Here, startIndex specifies the index at which point the search begins.
For indexOf( ), the search runs from startIndex to the end of the string.
For lastIndexOf( ), the search runs from startIndex to zero.

```
// Demonstrate indexOf() and lastIndexOf().
class indexOfDemo {
   public static void main(String args[]) {
      String s = "Now is the time for all good men " + "to come to the aid of their country.";
      System.out.println(s);
      System.out.println("indexOf(t) = " + s.indexOf('t'));
      System.out.println("lastIndexOf(t) = " + s.lastIndexOf('t'));
      System.out.println("indexOf(the) = " + s.indexOf("the"));
      System.out.println("lastIndexOf(the) = " + s.lastIndexOf("the"));
      System.out.println("indexOf(t, 10) = " + s.indexOf('t', 10));
      System.out.println("lastIndexOf(t, 60) = " + s.lastIndexOf('t', 60));
      System.out.println("indexOf(the, 10) = " + s.indexOf("the", 10));
      System.out.println("lastIndexOf(the, 60) = " + s.lastIndexOf("the", 60));
   }
}
```

Output:

```
Now is the time for all good men to come to the aid of their country.
indexOf(t) = 7
lastIndexOf(t) = 65
```

indexOf(the) = 7
lastIndexOf(the) = 55
indexOf(t, 10) = 11
lastIndexOf(t, 60) = 55
indexOf(the, 10) = 44
lastIndexOf(the, 60) = 55

below methods for modifying a String objects.

- *substring()*
- *concat()*
- *replace()*
- *replaceAll()*
- *replaceFirst()*
- *trim()*

substring() methods

We can extract a substring using substring( ) methods. There are two forms of substring() methods.

- substring(int beginIndex) - Returns a string that is a substring of this string.
- substring(int beginIndex, int endIndex) - Returns a string that is a substring of this string.

```
public class SubStringExample {
 public static void main(String[] args) {
  String str = "javaguides";

  // substring from start to end
  String subStr = str.substring(0, str.length());
  System.out.println("substring from 0 to length of the string : " + subStr);

  subStr = str.substring(4);
  System.out.println("Sub string starts from index 4 : " + subStr);

  // Remember index starts from 0
  System.out.println(str.substring(1));

  System.out.println("unhappy".substring(2));
  System.out.println("Harbison".substring(3));
  System.out.println("emptiness".substring(8));
 }
```

```
}
```

Output:

```
substring from 0 to length of the string : javaguides
Sub string starts from index 4 : guides
avaguides
happy
bison
s
```

concat(String str)

We can concatenate two strings using concat( ). The *concat()* method concatenates the specified string to the end of this string.

This method creates a new object that contains the invoking string with the contents of str appended to the end. *concat( )* performs the same function as +.

```
public class ConcatExmaple {
 public static void main(String[] args) {
  String str = "javaguides";
  str = str.concat(".net");
  System.out.println("Concatenates the specified string to the end of this string : " + str);

  System.out.println("cares".concat("s"));
  System.out.println("to".concat("get"));
 }
}
```

Output:

```
Concatenates the specified string to the end of this string : javaguides.net
caress
toget
```

replace( ) methods

The replace( ) method has two forms. The first replaces all occurrences of one character in the invoking string with another character. It has the following general form:

String replace(char original, char replacement)

The second form of replace( ) replaces one character sequence with another. It has this general form:

String replace(CharSequence original, CharSequence replacement)

```
public class ReplaceExample {
```

```
 public static void main(String[] args) {
  String str = "javaguides";
  String subStr = str.replace('a', 'b');
  System.out.println("replace char 'a' with char 'b' from given string : " + subStr);

  subStr = str.replace("guides", "tutorials");
  System.out.println("replace guides with tutorials from given string : " + subStr);

  subStr = str.replaceAll("[a-z]", "java");
  System.out.println(subStr);

  subStr = str.replaceFirst("[a-z]", "java");
  System.out.println(subStr);
 }
}
```

Output:

```
replace char 'a' with char 'b' from given string : jbvbguides
replace guides with tutorials from given string : javatutorials
javajavajavajavajavajavajavajavajavajava
javaavaguides
```

replaceAll(String regex, String replacement)

Replaces each substring of this string that matches the given regular expression with the given replacement.

```
public class ReplaceExample {
 public static void main(String[] args) {
  String str = "javaguides";
  String subStr = str.replaceAll("[a-z]", "java");
  System.out.println(subStr);
 }
}
```

Output:

```
javajavajavajavajavajavajavajavajavajava
```

replaceFirst(String regex, String replacement)

Replaces the first substring of this string that matches the given regular expression with the given replacement.

```
public class ReplaceExample {
 public static void main(String[] args) {
  String str = "javaguides";
  String subStr = str.replaceFirst("[a-z]", "java");
  System.out.println(subStr);
 }
}
```

Output:

Javaavaguides

trim( )

The trim( ) method returns a copy of the invoking string from which any leading and trailing whitespace has been removed. It has this general form:

String trim( )

```
public class TrimExample {
 public static void main(String[] args) {
  String str = "javaguides ";
  String subStr = str.trim();
  System.out.println("trim the space from given string : " + subStr);
 }
}
```

Output:

trim the space from given string : javaguides

26 b). Write a Java Program to demonstrate String Buffer Class and String Builder Class

```
package java_fundamentals.language_basics;

public class StringBufferClass {
        public static void main(String[] args) {

    StringBuffer s = new StringBuffer("Welcome");

    // Getting the length of the string
    int p = s.length();

    // Getting the total allocated capacity of the string
    int q = s.capacity();
```

```
    // Printing the length and capacity of
    // above generated input string on console
    System.out.println("Length of string Welcome="
                    + p);
    System.out.println(
       "Capacity of string Welcome=" + q);

  // Usage of append() method
    s.append("To");

    // Returns Welcome To
    System.out.println(s);

    s.append("Java");
    // Returns Welcome To Java
    System.out.println(s);

    s.insert(0, "Hi ");
    // Returns Hi Welcome To Java
    System.out.println(s);

  // Invoking reverse() method
    s.reverse();

    // Returns "avaJoTemocleW iH"
    System.out.println(s);

    s.delete(0, 3);
    // Returns JoTemocleW iH
    System.out.println(s);
    s.reverse();
    System.out.println(s);
    s.deleteCharAt(12);
    // Returns forGeek
    System.out.println(s);

    s.replace(0, 2, "JAVA");
    System.out.println(s);
```

```
        }
}
```

Output:

```
Length of string Welcome=7
Capacity of string Welcome=23
WelcomeTo
WelcomeToJava
Hi WelcomeToJava
avaJoTemocleW iH
JoTemocleW iH
Hi WelcomeToJ
Hi WelcomeTo
JAVA WelcomeTo
```

```
        //Java Program to demonstrate the use of StringBuilder class.
        public class BuilderTest{
           public static void main(String[] args){
              StringBuilder builder=new StringBuilder("hello");
              builder.append("java");
              System.out.println(builder);
           }
        }
```

Output:

```
hellojava
```

27 a). Write a Java program for multi-thread implementation.

```
1. // ABC class implements the interface Runnable
2. class ABC implements Runnable
3. {
4. public void run()
5. {
6.
7. // try-catch block
8. try
9. {
10. // moving thread t2 to the state timed waiting
11. Thread.sleep(100);
```

```
12. }
13. catch (InterruptedException ie)
14. {
15. ie.printStackTrace();
16. }
17.
18.
19. System.out.println("The state of thread t1 while it invoked the method join() on thread t
    2 -"+ ThreadState.t1.getState());
20.
21. // try-catch block
22. try
23. {
24. Thread.sleep(200);
25. }
26. catch (InterruptedException ie)
27. {
28. ie.printStackTrace();
29. }
30. }
31. }
32.
33. // ThreadState class implements the interface Runnable
34. public class ThreadState implements Runnable
35. {
36. public static Thread t1;
37. public static ThreadState obj;
38.
39. // main method
40. public static void main(String argvs[])
41. {
42. // creating an object of the class ThreadState
43. obj = new ThreadState();
44. t1 = new Thread(obj);
45.
46. // thread t1 is spawned
47. // The thread t1 is currently in the NEW state.
48. System.out.println("The state of thread t1 after spawning it - " + t1.getState());
49.
```

```
50. // invoking the start() method on
51. // the thread t1
52. t1.start();
53.
54. // thread t1 is moved to the Runnable state
55. System.out.println("The state of thread t1 after invoking the method start() on it -
      " + t1.getState());
56. }
57.
58. public void run()
59. {
60. ABC myObj = new ABC();
61. Thread t2 = new Thread(myObj);
62.
63. // thread t2 is created and is currently in the NEW state.
64. System.out.println("The state of thread t2 after spawning it - "+ t2.getState());
65. t2.start();
66.
67. // thread t2 is moved to the runnable state
68. System.out.println("the state of thread t2 after calling the method start() on it -
      " + t2.getState());
69.
70. // try-catch block for the smooth flow of the  program
71. try
72. {
73. // moving the thread t1 to the state timed waiting
74. Thread.sleep(200);
75. }
76. catch (InterruptedException ie)
77. {
78. ie.printStackTrace();
79. }
80.
81. System.out.println("The state of thread t2 after invoking the method sleep() on it -
      "+ t2.getState() );
82.
83. // try-catch block for the smooth flow of the  program
84. try
85. {
```

```
86. // waiting for thread t2 to complete its execution
87. t2.join();
88. }
89. catch (InterruptedException ie)
90. {
91. ie.printStackTrace();
92. }
93. System.out.println("The state of thread t2 when it has completed it's execution -
      " + t2.getState());
94. }
95.
96. }
```

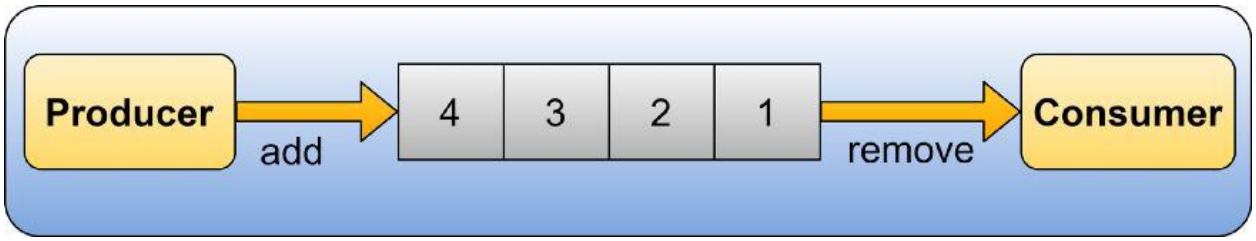
Output:

```
The state of thread t1 after spawning it - NEW
The state of thread t1 after invoking the method start() on it - RUNNABLE
The state of thread t2 after spawning it - NEW
the state of thread t2 after calling the method start() on it - RUNNABLE
The state of thread t1 while it invoked the method join() on thread t2 -TIMED_WAITING
The state of thread t2 after invoking the method sleep() on it - TIMED_WAITING
The state of thread t2 when it has completed it's execution - TERMINATED
```

27 b). Write a Java program to implement producer consumer problem using inter-thread communication mechanism

the Producer-Consumer problem in Java. This problem is **also known as the bounded-buffer problem**.

Producer and Consumer are two separate processes. Both processes share a common buffer or queue. The producer continuously produces certain data and pushes it onto the buffer, whereas the consumer consumes those data from the buffer.



Solution

The producer is to either go to sleep or discard data if the buffer is full. The next time the

consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same way, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer.
An inadequate solution could result in a deadlock where both processes are waiting to be awakened.

Implementation of Producer Consumer Class

- A **LinkedList list** – to store list of jobs in queue.
- **A Variable Capacity** – to check for if the list is full or not
- A mechanism to control the insertion and extraction from this list so that we do not insert into list if it is full or remove from it if it is empty.

```
// Java program to implement solution of producer
// consumer problem.

import java.util.LinkedList;

public class Threadexample {
	public static void main(String[] args)
		throws InterruptedException
	{
		// Object of a class that has both produce()
		// and consume() methods
		final PC pc = new PC();

		// Create producer thread
		Thread t1 = new Thread(new Runnable() {
			@Override
			public void run()
			{
				try {
					pc.produce();
				}
				catch (InterruptedException e) {
					e.printStackTrace();
				}
			}
		});

		// Create consumer thread
```

```
        Thread t2 = new Thread(new Runnable() {
                @Override
                public void run()
                {
                        try {
                                pc.consume();
                        }
                        catch (InterruptedException e) {
                                e.printStackTrace();
                        }
                }
        });

        // Start both threads
        t1.start();
        t2.start();

        // t1 finishes before t2
        t1.join();
        t2.join();
}

// This class has a list, producer (adds items to list
// and consumer (removes items).
public static class PC {

        // Create a list shared by producer and consumer
        // Size of list is 2.
        LinkedList<Integer> list = new LinkedList<>();
        int capacity = 2;

        // Function called by producer thread
        public void produce() throws InterruptedException
        {
                int value = 0;
                while (true) {
                        synchronized (this)
                        {
                                // producer thread waits while list
```

```
                        // is full
                        while (list.size() == capacity)
                                wait();

                        System.out.println("Producer produced-"
                                                + value);

                        // to insert the jobs in the list
                        list.add(value++);

                        // notifies the consumer thread that
                        // now it can start consuming
                        notify();

                        // makes the working of program easier
                        // to understand
                        Thread.sleep(1000);
                }
        }
}

// Function called by consumer thread
public void consume() throws InterruptedException
{
        while (true) {
                synchronized (this)
                {
                        // consumer thread waits while list
                        // is empty
                        while (list.size() == 0)
                                wait();

                        // to retrieve the first job in the list
                        int val = list.removeFirst();

                        System.out.println("Consumer consumed-"
                                                + val);

                        // Wake up producer thread
```

```
                        notify();

                        // and sleep
                        Thread.sleep(1000);
                    }
                }
            }
        }
}
```

Output:

```
Producer produced-0
Producer produced-1
Consumer consumed-0
Consumer consumed-1
Producer produced-2
```

Important Points

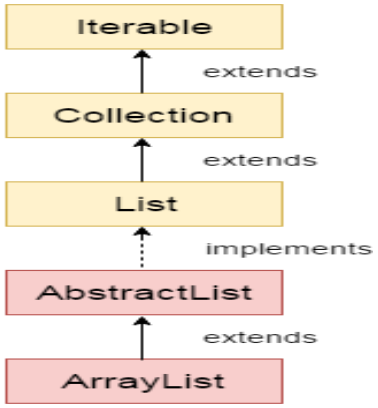
- In **PC class** (A class that has both produce and consume methods), a linked list of jobs and a capacity of the list is added to check that producer does not produce if the list is full.
- In **Producer class**, the value is initialized as 0.
    - Also, we have an infinite outer loop to insert values in the list. Inside this loop, we have a synchronized block so that only a producer or a consumer thread runs at a time.
    - An inner loop is there before adding the jobs to list that checks if the job list is full, the producer thread gives up the intrinsic lock on PC and goes on the waiting state.
    - If the list is empty, the control passes to below the loop and it adds a value in the list.
- In the **Consumer class**, we again have an infinite loop to extract a value from the list.
    - Inside, we also have an inner loop which checks if the list is empty.
    - If it is empty then we make the consumer thread give up the lock on PC and passes the control to producer thread for producing more jobs.
    - If the list is not empty, we go round the loop and removes an item from the list.
- In both the methods, we use notify at the end of all statements. The reason is simple, once you have something in list, you can have the consumer thread consume it, or if you have consumed something, you can have the producer produce something.

- sleep() at the end of both methods just make the output of program run in step wise manner and not display everything all at once so that you can see what actually is happening in the program.

28 a). Practice any two Programs on Collections.

Java **ArrayList** class uses a *dynamic array*

- Java ArrayList class can contain duplicate elements.
- Java ArrayList class maintains insertion order.
- Java ArrayList class is non synchronized.
- Java ArrayList allows random access because the array works on an index basis.
- In ArrayList, manipulation is a little bit slower than the LinkedList in Java because a lot of shifting needs to occur if any element is removed from the array list.
- We can not create an array list of the primitive types, such as int, float, char, etc. It is required to use the required wrapper class in such cases.




```
1. import java.util.*;
2. public class ArrayListExample2{
3.  public static void main(String args[]){
4.   ArrayList<String> list=new ArrayList<String>();//Creating arraylist
5.   list.add("Mango");//Adding object in arraylist
6.   list.add("Apple");
7.   list.add("Banana");
8.   list.add("Grapes");
9.   //Traversing list through Iterator
10.  Iterator itr=list.iterator();//getting the Iterator
11.  while(itr.hasNext()){//check if iterator has the elements
```

```
12.    System.out.println(itr.next());//printing the element and move to next
13.   }
14.  }
15. }
```

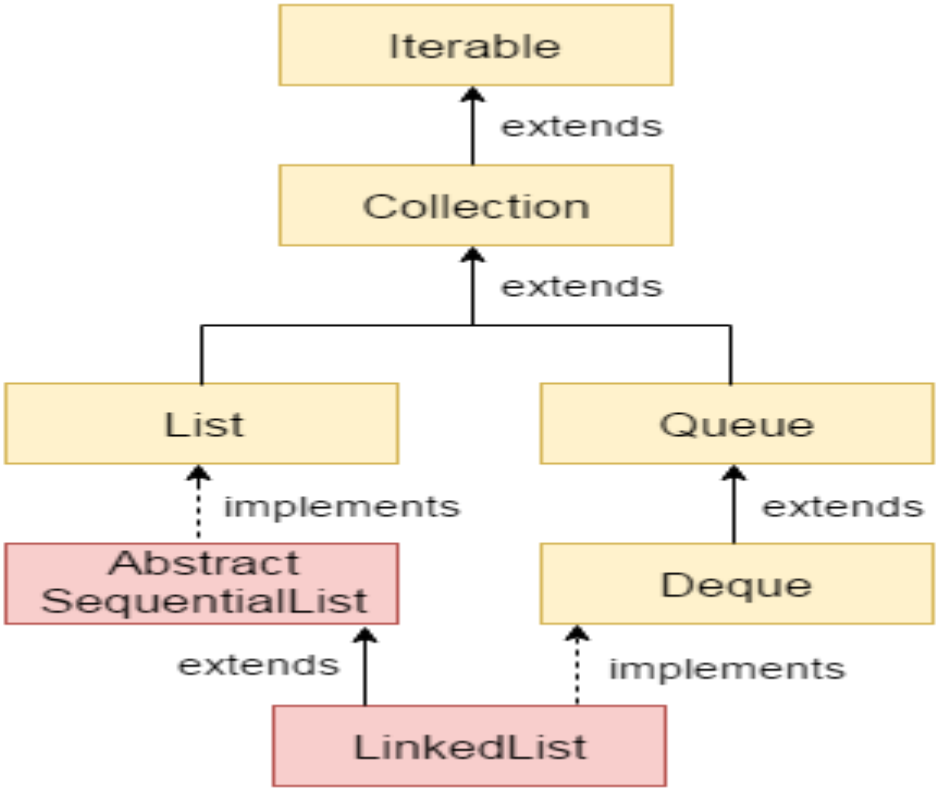
Output:

Mango
Apple
Banana
Grapes

Java LinkedList class

The important points about Java LinkedList are:

- Java LinkedList class can contain duplicate elements.
- Java LinkedList class maintains insertion order.
- Java LinkedList class is non synchronized.
- In Java LinkedList class, manipulation is fast because no shifting needs to occur.
- Java LinkedList class can be used as a list, stack or queue.




```
1. import java.util.*;
2. public class LinkedList3 {
```

```
3.
4.      public static void main(String [] args)
5.      {
6.        LinkedList<String> ll=new LinkedList<String>();
7.        ll.add("Ravi");
8.        ll.add("Vijay");
9.        ll.add("Ajay");
10.       ll.add("Anuj");
11.       ll.add("Gaurav");
12.       ll.add("Harsh");
13.       ll.add("Virat");
14.       ll.add("Gaurav");
15.       ll.add("Harsh");
16.       ll.add("Amit");
17.       System.out.println("Initial list of elements: "+ll);
18.     //Removing specific element from arraylist
19.         ll.remove("Vijay");
20.         System.out.println("After invoking remove(object) method: "+ll);
21.     //Removing element on the basis of specific position
22.         ll.remove(0);
23.         System.out.println("After invoking remove(index) method: "+ll);
24.         LinkedList<String> ll2=new LinkedList<String>();
25.         ll2.add("Ravi");
26.         ll2.add("Hanumat");
27.     // Adding new elements to arraylist
28.         ll.addAll(ll2);
29.         System.out.println("Updated list : "+ll);
30.     //Removing all the new elements from arraylist
31.         ll.removeAll(ll2);
32.         System.out.println("After invoking removeAll() method: "+ll);
33.     //Removing first element from the list
34.         ll.removeFirst();
35.         System.out.println("After invoking removeFirst() method: "+ll);
36.      //Removing first element from the list
37.         ll.removeLast();
38.         System.out.println("After invoking removeLast() method: "+ll);
39.      //Removing first occurrence of element from the list
40.         ll.removeFirstOccurrence("Gaurav");
41.         System.out.println("After invoking removeFirstOccurrence() method: "+ll);
```

```
42.         //Removing last occurrence of element from the list
43.            ll.removeLastOccurrence("Harsh");
44.            System.out.println("After invoking removeLastOccurrence() method: "+ll);
45.
46.            //Removing all the elements available in the list
47.            ll.clear();
48.            System.out.println("After invoking clear() method: "+ll);
49.       }
50.    }
```

```
Initial list of elements: [Ravi, Vijay, Ajay, Anuj, Gaurav, Harsh, Virat, Gaurav, Harsh, Amit]
After invoking remove(object) method: [Ravi, Ajay, Anuj, Gaurav, Harsh, Virat, Gaurav, Harsh,
Amit]
After invoking remove(index) method: [Ajay, Anuj, Gaurav, Harsh, Virat, Gaurav, Harsh, Amit]
Updated list : [Ajay, Anuj, Gaurav, Harsh, Virat, Gaurav, Harsh, Amit, Ravi, Hanumat]
After invoking removeAll() method: [Ajay, Anuj, Gaurav, Harsh, Virat, Gaurav, Harsh, Amit]
After invoking removeFirst() method: [Gaurav, Harsh, Virat, Gaurav, Harsh, Amit]
After invoking removeLast() method: [Gaurav, Harsh, Virat, Gaurav, Harsh]
After invoking removeFirstOccurrence() method: [Harsh, Virat, Gaurav, Harsh]
After invoking removeLastOccurrence() method: [Harsh, Virat, Gaurav]
After invoking clear() method: []
```

Java LinkedList Example: Book

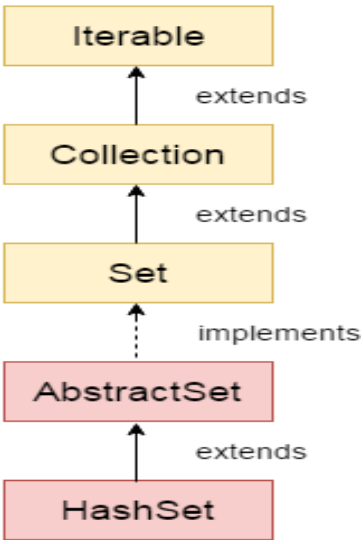
```
1. import java.util.*;
2. class Book {
3. int id;
4. String name,author,publisher;
5. int quantity;
6. public Book(int id, String name, String author, String publisher, int quantity) {
7.     this.id = id;
8.     this.name = name;
9.     this.author = author;
10.    this.publisher = publisher;
11.    this.quantity = quantity;
12. }
13. }
14. public class LinkedListExample {
15. public static void main(String[] args) {
16.    //Creating list of Books
17.    List<Book> list=new LinkedList<Book>();
```

```
18.  //Creating Books
19.  Book b1=new Book(101,"Let us C","Yashwant Kanetkar","BPB",8);
20.  Book b2=new Book(102,"Data Communications & Networking","Forouzan","Mc Graw
     Hill",4);
21.  Book b3=new Book(103,"Operating System","Galvin","Wiley",6);
22.  //Adding Books to list
23.  list.add(b1);
24.  list.add(b2);
25.  list.add(b3);
26.  //Traversing list
27.  for(Book b:list){
28.  System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);
29.  }
30. }
31. }
```

Output:

101 Let us C Yashwant Kanetkar BPB 8
102 Data Communications & Networking Forouzan Mc Graw Hill 4
103 Operating System Galvin Wiley 6

Java HashSet



Java HashSet class is used to create a collection that uses a hash table for storage. It inherits the AbstractSet class and implements Set interface.

The important points about Java HashSet class are:

- HashSet stores the elements by using a mechanism called **hashing.**
- HashSet contains unique elements only.

- HashSet allows null value.
- HashSet class is non synchronized.
- HashSet doesn't maintain the insertion order. Here, elements are inserted on the basis of their hashcode.
- HashSet is the best approach for search operations.
- The initial default capacity of HashSet is 16, and the load factor is 0.75.

```
1. import java.util.*;
2. class HashSet3{
3.  public static void main(String args[]){
4.   HashSet<String> set=new HashSet<String>();
5.         set.add("Ravi");
6.         set.add("Vijay");
7.         set.add("Arun");
8.         set.add("Sumit");
9.         System.out.println("An initial list of elements: "+set);
10.        //Removing specific element from HashSet
11.        set.remove("Ravi");
12.        System.out.println("After invoking remove(object) method: "+set);
13.        HashSet<String> set1=new HashSet<String>();
14.        set1.add("Ajay");
15.        set1.add("Gaurav");
16.        set.addAll(set1);
17.        System.out.println("Updated List: "+set);
18.        //Removing all the new elements from HashSet
19.        set.removeAll(set1);
20.        System.out.println("After invoking removeAll() method: "+set);
21.        //Removing elements on the basis of specified condition
22.        set.removeIf(str->str.contains("Vijay"));
23.        System.out.println("After invoking removeIf() method: "+set);
24.        //Removing all the elements available in the set
25.        set.clear();
26.        System.out.println("After invoking clear() method: "+set);
27. }
28. }
```

```
An initial list of elements: [Vijay, Ravi, Arun, Sumit]
After invoking remove(object) method: [Vijay, Arun, Sumit]
Updated List: [Vijay, Arun, Gaurav, Sumit, Ajay]
After invoking removeAll() method: [Vijay, Arun, Sumit]
```

After invoking removeIf() method: [Arun, Sumit]
After invoking clear() method: []

Java HashSet Example: Book

Let's see a HashSet example where we are adding books to set and printing all the books.

```
1. import java.util.*;
2. class Book {
3. int id;
4. String name,author,publisher;
5. int quantity;
6. public Book(int id, String name, String author, String publisher, int quantity) {
7.     this.id = id;
8.     this.name = name;
9.     this.author = author;
10.    this.publisher = publisher;
11.    this.quantity = quantity;
12. }
13. }
14. public class HashSetExample {
15. public static void main(String[] args) {
16.    HashSet<Book> set=new HashSet<Book>();
17.    //Creating Books
18.    Book b1=new Book(101,"Let us C","Yashwant Kanetkar","BPB",8);
19.    Book b2=new Book(102,"Data Communications & Networking","Forouzan","Mc Graw
       Hill",4);
20.    Book b3=new Book(103,"Operating System","Galvin","Wiley",6);
21.    //Adding Books to HashSet
22.    set.add(b1);
23.    set.add(b2);
24.    set.add(b3);
25.    //Traversing HashSet
26.    for(Book b:set){
27.    System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);
28.    }
29. }
30. }
```

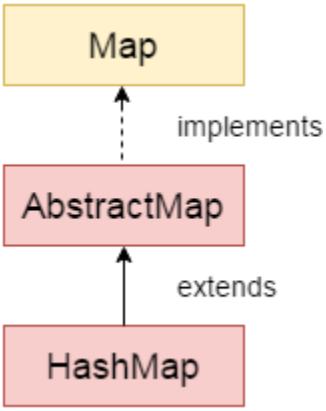
Output:

```
101 Let us C Yashwant Kanetkar BPB 8
102 Data Communications & Networking Forouzan Mc Graw Hill 4
```

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

103 Operating System Galvin Wiley 6

Java HashMap



Java **HashMap** class implements the Map interface which allows us *to store key and value pair*, where keys should be unique. If you try to insert the duplicate key, it will replace the element of the corresponding key. It is easy to perform operations using the key index like updation, deletion, etc. HashMap class is found in the java.util package.
HashMap in Java is like the legacy Hashtable class, but it is not synchronized. It allows us to store the null elements as well, but there should be only one null key. Since Java 5, it is denoted as HashMap<K,V>, where K stands for key and V for value. It inherits the AbstractMap class and implements the Map interface.
Points to remember

- Java HashMap contains values based on the key.
- Java HashMap contains only unique keys.
- Java HashMap may have one null key and multiple null values.
- Java HashMap is non synchronized.
- Java HashMap maintains no order.
- The initial default capacity of Java HashMap class is 16 with a load factor of 0.75.

```
1. import java.util.*;
2. class HashMap3{
3.  public static void main(String args[]){
4.    HashMap<Integer,String> hm=new HashMap<Integer,String>();
5.      hm.put(100,"Amit");
6.      hm.put(101,"Vijay");
7.      hm.put(102,"Rahul");
8.      System.out.println("Initial list of elements:");
9.     for(Map.Entry m:hm.entrySet())
10.    {
```

```
11.       System.out.println(m.getKey()+" "+m.getValue());
12.     }
13.     System.out.println("Updated list of elements:");
14.     hm.replace(102, "Gaurav");
15.     for(Map.Entry m:hm.entrySet())
16.     {
17.       System.out.println(m.getKey()+" "+m.getValue());
18.     }
19.     System.out.println("Updated list of elements:");
20.     hm.replace(101, "Vijay", "Ravi");
21.     for(Map.Entry m:hm.entrySet())
22.     {
23.       System.out.println(m.getKey()+" "+m.getValue());
24.     }
25.     System.out.println("Updated list of elements:");
26.     hm.replaceAll((k,v) -> "Ajay");
27.     for(Map.Entry m:hm.entrySet())
28.     {
29.       System.out.println(m.getKey()+" "+m.getValue());
30.     }
31.  }
32. }
```

```
Initial list of elements:
100 Amit
101 Vijay
102 Rahul
Updated list of elements:
100 Amit
101 Vijay
102 Gaurav
Updated list of elements:
100 Amit
101 Ravi
102 Gaurav
Updated list of elements:
100 Ajay
101 Ajay
102 Ajay
```

Difference between HashSet and HashMap

HashSet contains only values whereas HashMap contains an entry(key and value).
Java HashMap Example: Book

```
1. import java.util.*;
2. class Book {
3. int id;
4. String name,author,publisher;
5. int quantity;
6. public Book(int id, String name, String author, String publisher, int quantity) {
7.     this.id = id;
8.     this.name = name;
9.     this.author = author;
10.    this.publisher = publisher;
11.    this.quantity = quantity;
12. }
13. }
14. public class MapExample {
15. public static void main(String[] args) {
16.    //Creating map of Books
17.    Map<Integer,Book> map=new HashMap<Integer,Book>();
18.    //Creating Books
19.    Book b1=new Book(101,"Let us C","Yashwant Kanetkar","BPB",8);
20.    Book b2=new Book(102,"Data Communications & Networking","Forouzan","Mc Graw
    Hill",4);
21.    Book b3=new Book(103,"Operating System","Galvin","Wiley",6);
22.    //Adding Books to map
23.    map.put(1,b1);
24.    map.put(2,b2);
25.    map.put(3,b3);
26.
27.    //Traversing map
28.    for(Map.Entry<Integer, Book> entry:map.entrySet()){
29.       int key=entry.getKey();
30.       Book b=entry.getValue();
31.       System.out.println(key+" Details:");
32.       System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);
33.    }
34. }
35. }
```

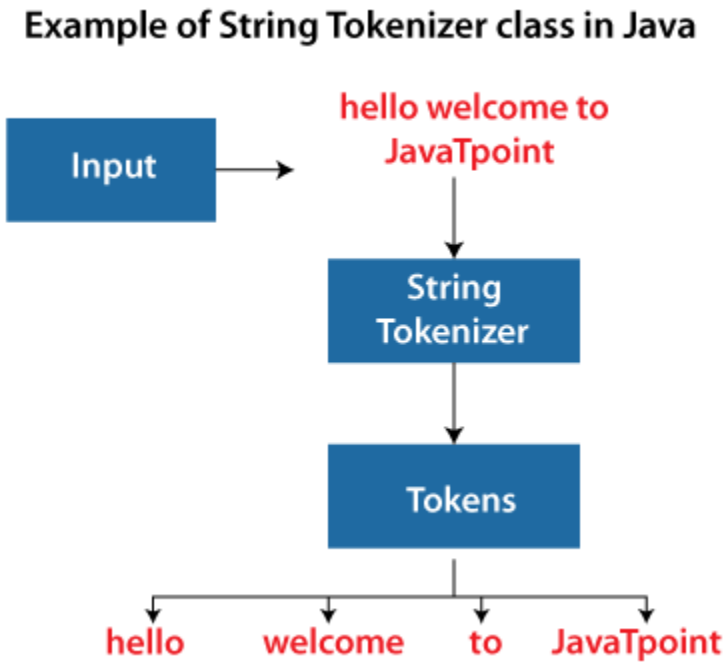
Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

Output:

```
1 Details:
101 Let us C Yashwant Kanetkar BPB 8
2 Details:
102 Data Communications and Networking Forouzan Mc Graw Hill 4
3 Details:
103 Operating System Galvin Wiley 6
```

28 b). Practice any two Programs on String Tokenizer & Scanner.

The **java.util.StringTokenizer** class allows you to break a String into tokens. It is simple way to break a String. It is a legacy class of Java.




```
1. import java.util.StringTokenizer;
2. public class StringTokenizer1
3. {
4.  /* Driver Code */
5.  public static void main(String args[])
6.  {
7.    /* StringTokenizer object */
```

```
8.    StringTokenizer st = new StringTokenizer("Demonstrating methods from StringTokeniz
      er class"," ");
9.      /* Checks if the String has any more tokens */
10.    while (st.hasMoreTokens())
11.    {
12.       System.out.println(st.nextToken());
13.    }
14. }
15. }
```

Output:

```
Demonstrating
methods
from
StringTokenizer
class
```

Scanner Class in Java

Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double, etc. and strings. It is the easiest way to read input in a Java program, though not very efficient if you want an input method for scenarios where time is a constraint like in competitive programming.

- To create an object of Scanner class, we usually pass the predefined object System.in, which represents the standard input stream. We may pass an object of class File if we want to read input from a file.
- To read numerical values of a certain data type XYZ, the function to use is nextXYZ(). For example, to read a value of type short, we can use nextShort()
- To read strings, we use nextLine().
- To read a single character, we use next().charAt(0). next() function returns the next token/word in the input as a string and charAt(0) function returns the first character in that string.

```
// Java program to read data of various types using Scanner class.
import java.util.Scanner;
public class ScannerDemo1
{
        public static void main(String[] args)
        {
                // Declare the object and initialize with
                // predefined standard input object
                Scanner sc = new Scanner(System.in);
```

```
        // String input
        String name = sc.nextLine();

        // Character input
        char gender = sc.next().charAt(0);

        // Numerical data input
        // byte, short and float can be read
        // using similar-named functions.
        int age = sc.nextInt();
        long mobileNo = sc.nextLong();
        double cgpa = sc.nextDouble();

        // Print the values to check if the input was correctly obtained.
        System.out.println("Name: "+name);
        System.out.println("Gender: "+gender);
        System.out.println("Age: "+age);
        System.out.println("Mobile Number: "+mobileNo);
        System.out.println("CGPA: "+cgpa);
    }
}
```

Input :

```
Geek
F
40
9876543210
9.9
```

Output :

```
Name: Geek
Gender: F
Age: 40
Mobile Number: 9876543210
CGPA: 9.9
```

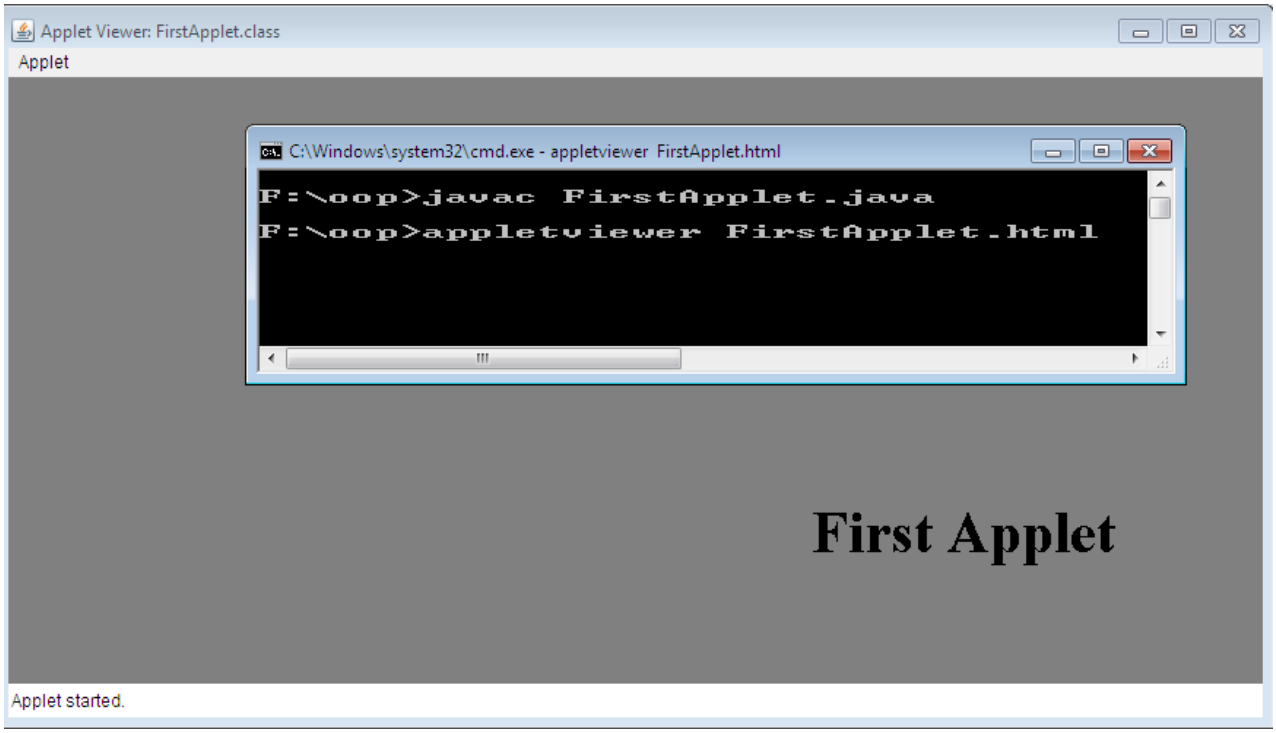
29 a). Write a Java Program to develop an applet that displays a simple message.

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

```
//FirstApplet.java import java.awt.*; import java.applet.*;
public class FirstApplet extends Applet
{
public void init()
{
}
public void paint(Graphics g)
{
setBackground(Color.gray);
Font f1=new Font("Times New Roman",Font.BOLD,40); g.setFont(f1);
g.drawString("First Applet",550,325);
}
}
```

```
//FirstApplet.html
<html>
<body>
<Applet code="FirstApplet.class" width="1000" height="600">
</Applet>
</body>
</html>
```

Output:



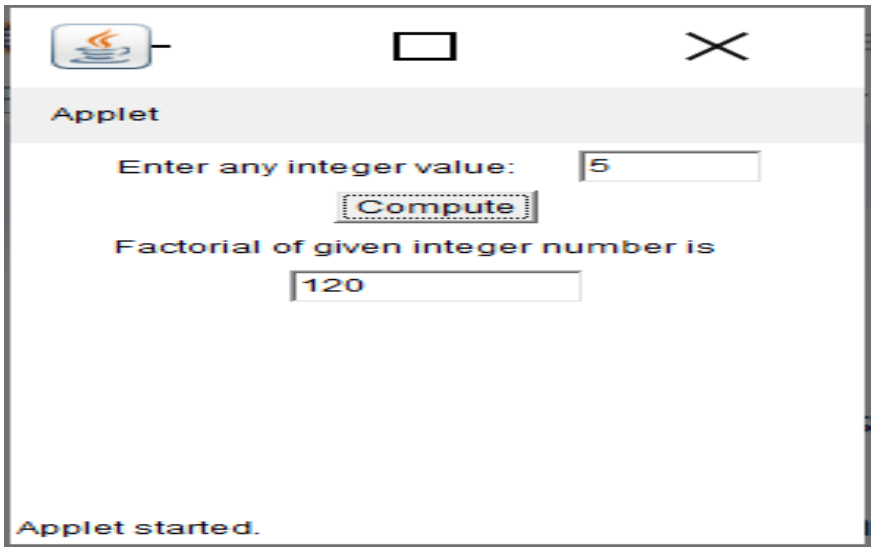
29 b). Develop an applet that receives an integer in one text field, and computes its factorial Value and returns it in another text field, when the button named —Compute‖ is clicked.

```
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;

/*<applet code="Fact.class" height=300 width=300></applet>*/

public class Factorial extends Applet implements ActionListener
{
        Label inputLable,outputLable;
        TextField inputTextField,outputTextField;
        Button btn;
        public void init(){
                inputLable=new Label("Enter any integer value: ");
                inputTextField=new TextField(5);
                btn=new Button("Compute");
```

```
                btn.addActionListener(this);
                outputLable=new Label("Factorial of given integer number is ");
                outputTextField=new TextField(10);
                add(inputLable);
                add(inputTextField);
                add(btn);
                add(outputLable);
                add(outputTextField);
        }
        public void actionPerformed(ActionEvent e)
        {
                if(e.getSource()==btn)
                {
                        int fact=fact(Integer.parseInt(inputTextField.getText()));
                        outputTextField.setText(String.valueOf(fact));
                }
        }
        int fact(int f)
        {
                if(f==0)
                        return 1;
                else
                        return f*fact(f-1);
        }
}
```

Output:

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			



29 c). Write a java program to handle keyboard events.

Problem Solution

1. The program uses the interfaces **KeyListener** and **ActionListener** of the **java.awt** package.

a) The **KeyListener** interface has three member methods :

i) **public void keyReleased(KeyEvent)**: This method gives the Unicode of the key released and its character equivalent if the key pressed is a letter.

ii) **public void keyPressed(KeyEvent)**: This method gives the Unicode of the key pressed and its character equivalent if the key pressed is a letter.

iii) **public void keyTyped(KeyEvent)**: This method gives the character equivalent of the key pressed provided it is a valid character.

b) The **ActionListener** interface has member method :

public void actionPerformed(ActionEvent): This method works on the click of the exit button and the functions closes the frame.

2. **@Override** is a keyword used to override any method on the parent class. When the sub class has any method defined in parent/super class with same name and parameters, the Override keyword is used. This overrides the function of super class and executed the function defined in the sub class.

3. Create a class that implements the two required interfaces – **KeyListener** and **ActionListener**.

4. Create a frame, two text fields for input & output and an exit button with required dimensions. After positioning the co-ordinates of the text fields and button, add them to the frame.

5. Associate **KeyListener** with the input text field and **ActionListener** with the exit button.

6. Display the frame.

7. The functions **getKeyCode()** and **getKeyChar()** give the unicode and character representation

of the key pressed respectively. Use them to display the necessary message in the output text box.

```
1.  /* Java Program to demonstrate the event actions associated with a keyboard */
2.  import javax.swing.*;
3.  import java.awt.*;
4.  import java.awt.event.*;
5.  class Keyboard_Event implements KeyListener,ActionListener
6.  {
7.     static JFrame frame;
8.     static JTextField output;
9.     static JTextField input;
10.    //Driver function
11.    public static void main(String args[])
12.    {
13.       //Create a frame
14.       frame=new JFrame("Keyboard Event");
15.       frame.setBackground(Color.white);
16.       frame.setSize(500,500);
17.       frame.setLayout(null);
18.       //Create a text field for output
19.       output=new JTextField();
20.       output.setBounds(0,0,500,50);
21.       frame.add(output);
22.       //Create a text field for input
23.       input=new JTextField();
24.       input.setBounds(0,400,500,50);
25.       frame.add(input);
26.       //Create an exit button
27.       JButton exit=new JButton("Exit");
28.       exit.setBounds(220,200,60,30);
29.       frame.add(exit);
30.       //Create an object of the class
31.       Keyboard_Event obj=new Keyboard_Event();
32.       //Associate KeyListener with input
33.       input.addKeyListener(obj);
34.       //Associate ActionListener with exit
35.       exit.addActionListener(obj);
36.       frame.setVisible(true);
37.    }
```

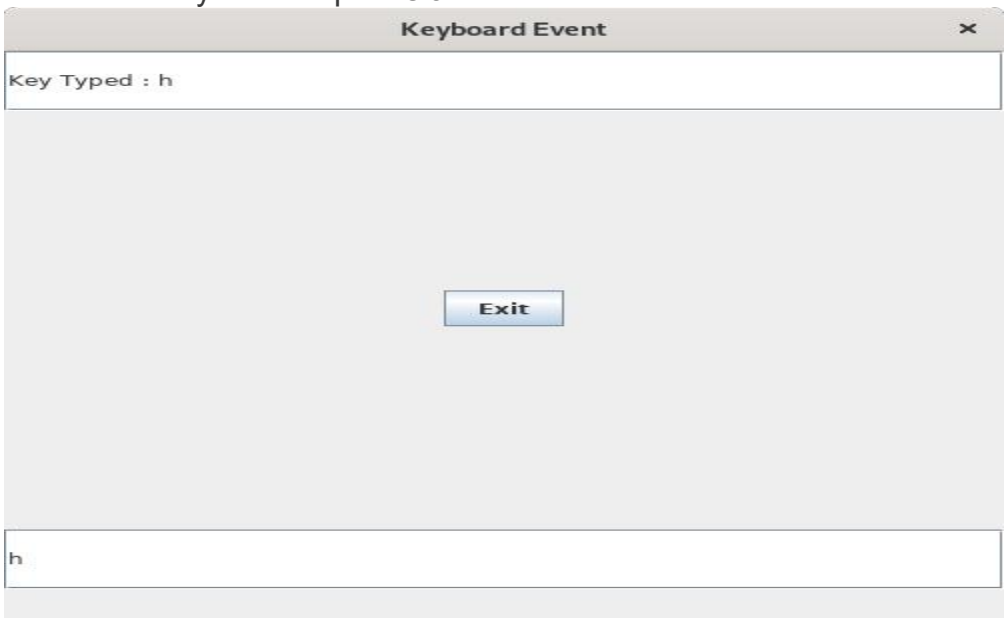
```
38.  //function to dispose the frame when exit button is clicked
39.  @Override
40.  public void actionPerformed(ActionEvent e)
41.  {
42.     frame.dispose();
43.  }
44.  /*function to display the unicode of key released
45.    and the character if it is a letter or a digit*/
46.  @Override
47.  public void keyReleased(KeyEvent e)
48.  {
49.     output.setText("");
50.     output.setText("Key Released : "+e.getKeyCode());
51.     if(Character.isLetter(e.getKeyChar()))
52.        keyTyped(e);
53.     if(Character.isDigit(e.getKeyChar()))
54.        keyTyped(e);
55.  }
56.  /*function to display the unicode of key pressed
57.    and the character if it is a letter or a digit*/
58.  @Override
59.  public void keyPressed(KeyEvent e)
60.  {
61.     output.setText("");
62.     output.setText("Key Pressed : "+e.getKeyCode());
63.     if(Character.isLetter(e.getKeyChar()))
64.        keyTyped(e);
65.     if(Character.isDigit(e.getKeyChar()))
66.        keyTyped(e);
67.  }
68.  //function to display the character of the key typed
69.  @Override
70.  public void keyTyped(KeyEvent e)
71.  {
72.     output.setText("");
73.     output.setText("Key Typed : "+e.getKeyChar());
74.  }
75. }
```

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

Runtime Test Cases

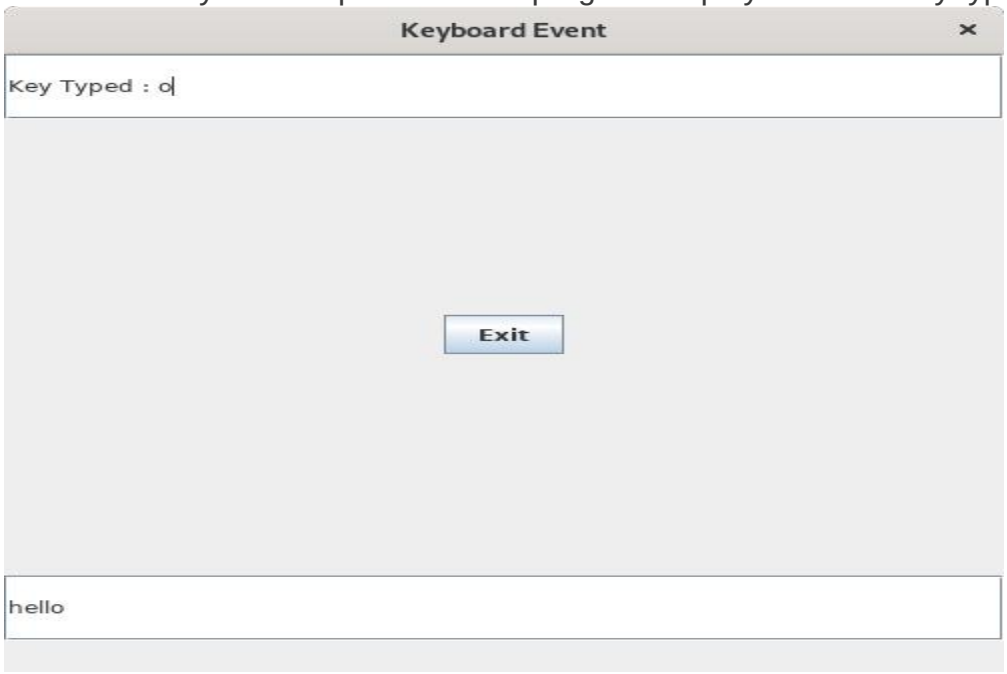
Here's the run time test cases for keyboard events.

Test case 1 – Here's the runtime output of the key typed operation with key "h", where "h" is first entry of the input field.



Test case 2 – Here's the runtime output of the multiple entries in key typed operation by using alphabets in the input field. For example, here the entered key is "hello", where "o"

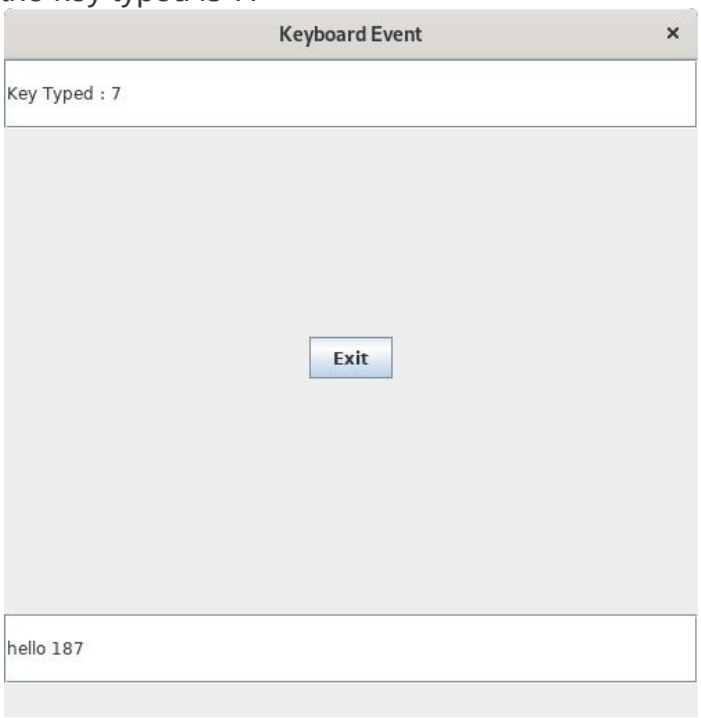
is the last entry of the input field. The program displays that the key typed is o.



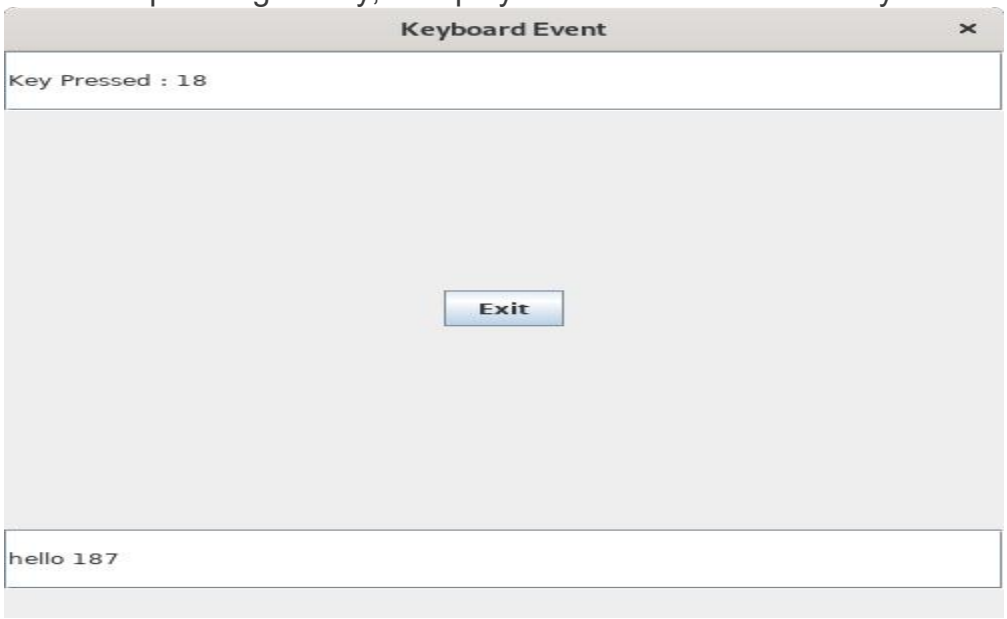
Test case 3 – Here's the runtime output of the multiple entries in key typed operation by using alphabets and digits in the input field. For example, here the entered key is "hello 1", where "1" is the last entry of the input field. The program displays that the key typed is 1.

Test case 4 – Here's the runtime output of the multiple entries in key typed sequence operation by using alphabets and digits in the input field. For example, here the entered key is "hello 187", where "7" is the last entry of the input field. The program displays that

the key typed is 7.



Test case 5 – Here's the runtime output of the special key pressed operation with key "alt". After pressing alt key, it displays the unicode of the alt key which is 18.



29 d). Write a java program to handle Mouse events
Problem Solution

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			

1. The program uses the interfaces **MouseListener** and **ActionListener** of the **java.awt** package.
a) The **MouseListener** interface has five member methods :
i) **public void mouseEntered(MouseEvent):** This method gives the co-ordinates of the point from where the mouse has entered the frame.
ii) **public void mouseExited(MouseEvent):** This method gives the co-ordinates of the point from where the mouse has left the frame.
iii) **public void mouseReleased(MouseEvent):** This method gives the co-ordinates of the point till where the mouse cursor was dragged before release and also the button clicked on the mouse.
iv) **public void mousePressed(MouseEvent):** This method gives the co-ordinates of the point from where the mouse was pressed and also the button.
v) **public void mouseClicked(MouseEvent):** This method gives the co-ordinates of the point where the mouse cursor was clicked and also the button clicked.
b) The **ActionListener** interface has member method :
public void actionPerformed(ActionEvent): This method works on the click of the exit button and the functions closes the frame.
2. **@Override** is a keyword used to override any method on the parent class. When the sub class has any method defined in parent/super class with same name and parameters, the Override keyword is used. This overrides the function of super class and executed the function defined in the sub class.
3. Create a class that implements the two required interfaces –
MouseListener and **ActionListener**.
4. Create a frame, text field and an exit button with required dimensions. After positioning the co-ordinates of the text field and button, add them to the frame.
5. Associate MouseListener with the frame and ActionListener with the exit button. Display the frame.
6. The functions **getX()** and **getY()** gives the x & y co-ordinates of the cursor respectively. Use them to display the necessary message in the text box.

```
1.  /* Java Program to demonstrate the event actions associated with a mouse */
2.  import javax.swing.*;
3.  import java.awt.*;
4.  import java.awt.event.*;
5.  public class Mouse_Event implements MouseListener,ActionListener
6.  {
7.      static JFrame frame;
8.      static JTextField text;
9.      //Driver function
10.     public static void main(String[] args)
```

```
{
   //Create a Frame
   frame=new JFrame("Mouse Event");
   frame.setBackground(Color.white);
   frame.setSize(500,500);
   frame.setLayout(null);
   //Create a TextField
   text=new JTextField();
   text.setBounds(0,0,500,50);
   frame.add(text);
   //Create a exit button to close the frame
   JButton exit=new JButton("Exit");
   exit.setBounds(220,235,60,30);
   frame.add(exit);
   //Create an object of the class Mouse_Event
   Mouse_Event obj=new Mouse_Event();
   //Associate MouseListener with the frame
   frame.addMouseListener(obj);
   //Associate ActionListener with button exit
   exit.addActionListener(obj);
   //Display frame
   frame.setVisible(true);
}
//function to dispose the frame on click of exit button
@Override
public void actionPerformed(ActionEvent e)
{
   frame.dispose();
}
//function to get co-ordinates from where cursor entered the frame
@Override
public void mouseEntered(MouseEvent e)
{
   text.setText("");
   text.setText("Mouse Entered the frame from point ");
   text.setText(text.getText()+e.getX()+" "+e.getY());
}
//function to get co-ordinates from where cursor exited the frame
@Override
```

```
50.  public void mouseExited(MouseEvent e)
51.  {
52.     text.setText("");
53.     text.setText("Mouse Exited the frame from point ");
54.     text.setText(text.getText()+e.getX()+" "+e.getY());
55.  }
56.  //function to get co-ordinates where mouse was released
57.  @Override
58.  public void mouseReleased(MouseEvent e)
59.  {
60.     text.setText("");
61.     String button="Right";
62.     if(e.getButton()==MouseEvent.BUTTON1)
63.        button="Left";
64.     text.setText(button+" Button Released at point ");
65.     text.setText(text.getText()+e.getX()+" "+e.getY());
66.  }
67.  //function to get co-ordinates where and which button of mouse was pressed
68.  @Override
69.  public void mousePressed(MouseEvent e)
70.  {
71.     text.setText("");
72.     String button="Right";
73.     if(e.getButton()==MouseEvent.BUTTON1)
74.        button="Left";
75.     text.setText(button+" Button Pressed at point ");
76.     text.setText(text.getText()+e.getX()+" "+e.getY());
77.  }
78.  //function to get co-ordinates where and which button of mouse was clicked
79.  @Override
80.  public void mouseClicked(MouseEvent e)
81.  {
82.     text.setText("");
83.     String button="Right";
84.     if(e.getButton()==MouseEvent.BUTTON1)
85.        button="Left";
86.     text.setText(button+" Button Clicked at point ");
87.     text.setText(text.getText()+e.getX()+" "+e.getY());
88.  }
```

89. }

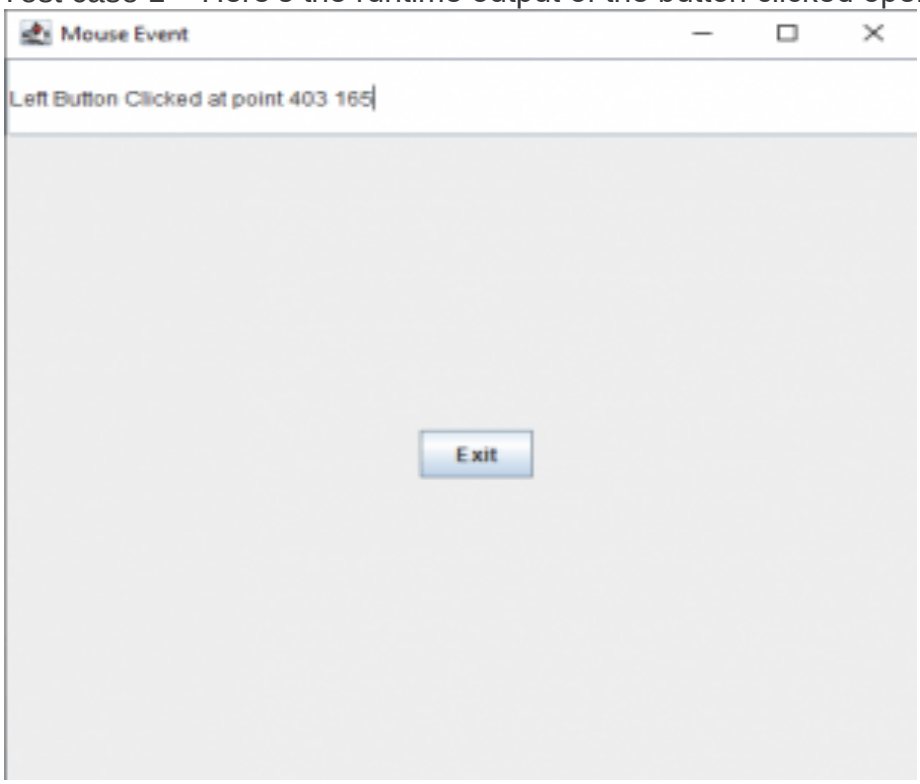
Program Explanation

1. The program demonstrates the functions of the **MouseListener** interface.
a) When the cursor enters the frame, the **mouseEntered** function is called.
b) When the cursor exits the frame, the **mouseExited** function is called.
c) When the cursor is released after being pressed, the **mouseReleased** function is called.
d) When the cursor is kept pressed at the frame, the **mousePressed** function is called.
e) When the cursor is clicked at a point on the frame, the **mouseClicked** function is called.
2. The **@Override** keyword overrides the functions of the parent class and executes the functions of sub class.

Runtime Test Cases

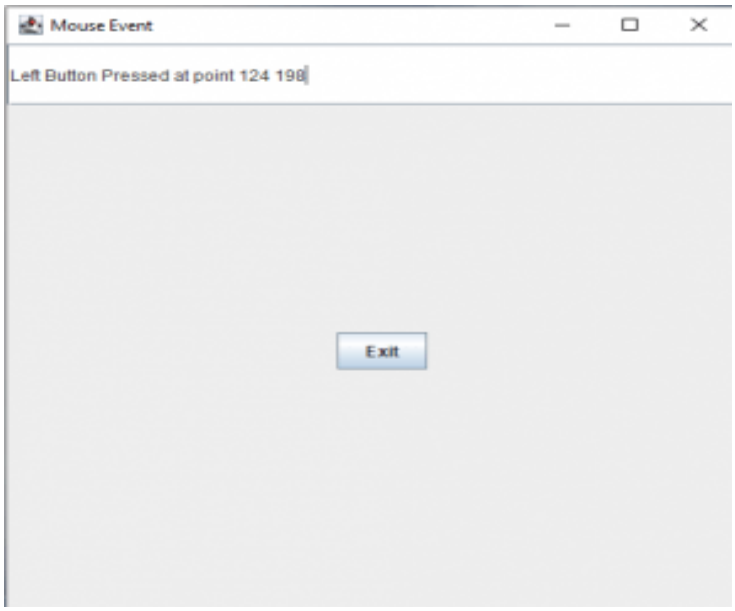
Here's the run time test cases for mouse events.

Test case 1 – Here's the runtime output of the button clicked operation.



Test case 2 – Here's the runtime output of the button pressed operation.

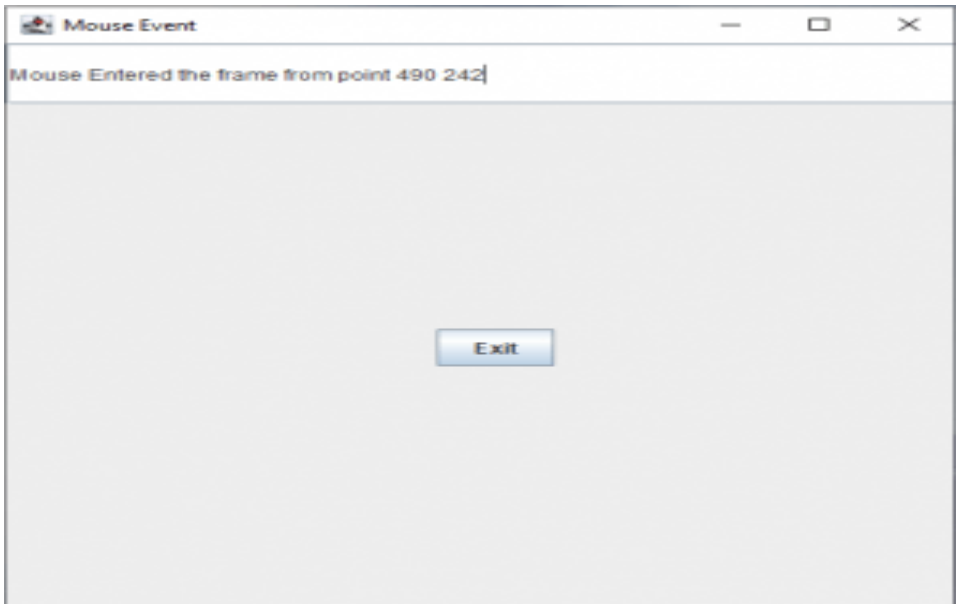
Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			



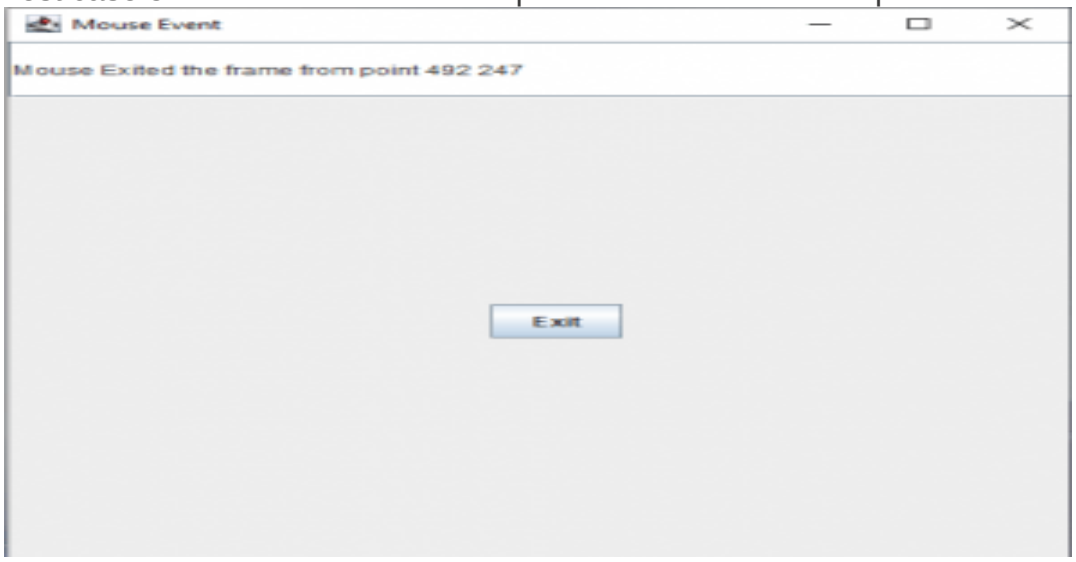
Test case 3 – Here's the runtime output of the button released operation.

Test case 4 – Here's the runtime output of the mouse entered operation.

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			



Test case 5 – Here's the runtime output of the mouse exited operation.



30 a). Write a Java Program to demonstrate AWT Label & Button.

Java AWT Button

A button is basically a control component with a label that generates an event when pushed. The **Button** class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed.

When we press a button and release it, AWT sends an instance of **ActionEvent** to that button by calling **processEvent** on the button. The **processEvent** method of the button receives the all

the events, then it passes an action event by calling its own method **processActionEvent**. This method passes the action event on to action listeners that are interested in the action events generated by the button.

Java AWT Label

The object of the Label class is a component for placing text in a container. It is used to display a single line of **read only text**. The text can be changed by a programmer but a user cannot edit it directly.

It is called a passive control as it does not create any event when it is accessed. To create a label, we need to create the object of **Label** class.

```
import java.awt.*;
import java.awt.event.*;
class MyLoginWindow extends Frame
{
   TextField name,pass;
   Button b1,b2;
   MyLoginWindow()
   {
      setLayout(new FlowLayout());
      this.setLayout(null);
      Label n=new Label("Name:",Label.CENTER);
      Label p=new Label("password:",Label.CENTER);
      name=new TextField(20);
      pass=new TextField(20);
      pass.setEchoChar('#');
      b1=new Button("submit");
      b2=new Button("cancel");
      this.add(n);
      this.add(name);
      this.add(p);
      this.add(pass);
      this.add(b1);
      this.add(b2);
      n.setBounds(70,90,90,60);
      p.setBounds(70,130,90,60);
      name.setBounds(200,100,90,20);
      pass.setBounds(200,140,90,20);
```

```
        b1.setBounds(100,260,70,40);
        b2.setBounds(180,260,70,40);

    }
    public static void main(String args[])
    {
        MyLoginWindow ml=new MyLoginWindow();
        ml.setVisible(true);
        ml.setSize(400,400);
        ml.setTitle("my login window");

    }
}
```

OUTPUT:

C:\java>javac MyLoginWindow.java

C:\java>java MyLoginWindow

30 b). Write a Java Program to demonstrate JLabel, JTextField & JButton.

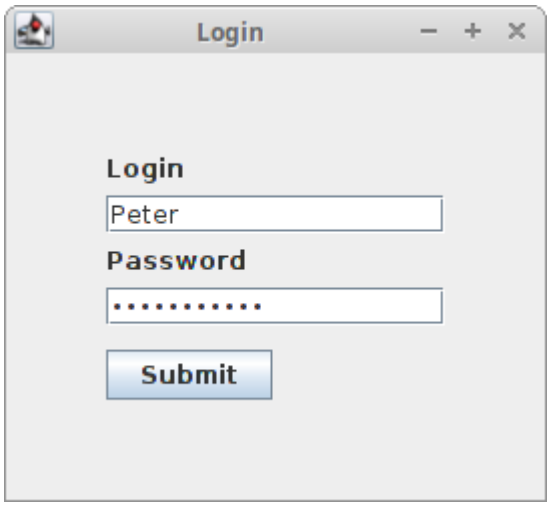
JLabel is a component used for displaying a label for some components. It is commonly partnered with a text field or a password field. JTextField is an input component allowing users to add some text. JPasswordField in Java is a special type of text field that allows you to hide or change the character being displayed to the user.

```
import javax.swing.AbstractAction;
import javax.swing.GroupLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import java.awt.Component;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.util.Arrays;

import static javax.swing.LayoutStyle.ComponentPlacement.UNRELATED;

public class PasswordEx extends JFrame {

   private JTextField loginField;
   private JPasswordField passField;

   public PasswordEx() {

      initUI();
   }

   private void initUI() {

      var lbl1 = new JLabel("Login");
      var lbl2 = new JLabel("Password");

      loginField = new JTextField(15);
      passField = new JPasswordField(15);

      var submitButton = new JButton("Submit");
      submitButton.addActionListener(new SubmitAction());
```

```
    createLayout(lbl1, loginField, lbl2, passField, submitButton);

    setTitle("Login");
    setLocationRelativeTo(null);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

private class SubmitAction extends AbstractAction {

    @Override
    public void actionPerformed(ActionEvent e) {

        doSubmitAction();
    }

    private void doSubmitAction() {

        var login = loginField.getText();
        var passwd = passField.getPassword();

        if (!login.isEmpty() && passwd.length != 0) {

            System.out.format("User %s entered %s password%n",
                    login, String.valueOf(passwd));
        }

        Arrays.fill(passwd, '0');
    }
}

private void createLayout(Component... arg) {

    var pane = getContentPane();
    var gl = new GroupLayout(pane);
    pane.setLayout(gl);

    gl.setAutoCreateGaps(true);
    gl.setAutoCreateContainerGaps(true);
```

```
        gl.setHorizontalGroup(gl.createSequentialGroup()
                .addGap(50)
                .addGroup(gl.createParallelGroup()
                        .addComponent(arg[0])
                        .addComponent(arg[1])
                        .addComponent(arg[2])
                        .addComponent(arg[3])
                        .addComponent(arg[4]))
                .addGap(50)
        );

        gl.setVerticalGroup(gl.createSequentialGroup()
                .addGap(50)
                .addGroup(gl.createSequentialGroup()
                        .addComponent(arg[0])
                        .addComponent(arg[1], GroupLayout.DEFAULT_SIZE,
                                GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                        .addComponent(arg[2])
                        .addComponent(arg[3], GroupLayout.DEFAULT_SIZE,
                                GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(UNRELATED)
                        .addComponent(arg[4]))
                .addGap(50)
        );

        pack();
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> {

            var ex = new PasswordEx();
            ex.setVisible(true);
        });
    }
}
```

Regulation: AK20	Subject Code: CSE(20APC0514)/ CIC(20APC3610)	Subject Name : Object Oriented Programming Through JAVA LAB	AY: 2021-2022 II B.Tech II Sem
JAVA Laboratory Record Programs			



30 c). Write a program to design a calculator using event driven programming paradigm of java

To design a calculator using event driven programming paradigm of Java with the following options

1. Decimal Manipulations
2. Scientific Manipulations

Procedure:

1. Import the swing packages and awt packages.
2. Create the class scientificcalculator that implements action listener.
3. Create the container and add controls for digits , scientific calculations and decimal Manipulations.
4. The different layouts can be used to lay the controls.
5. When the user presses the control , the event is generated and handled .
6. The corresponding decimal , numeric and scientific calculations are performed.

Program:

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;
public class ScientificCalculator extends JFrame implements ActionListener
{
JTextField tfield;
double temp, temp1, result, a;
static double m1, m2;
int k = 1, x = 0, y = 0, z = 0;
```

```
char ch;
JButton b1, b2, b3, b4, b5, b6, b7, b8, b9, zero, clr, pow2, pow3, exp,
fac, plus, min, div, log, rec, mul, eq, addSub, dot, mr, mc, mp,
mm, sqrt, sin, cos, tan;
Container cont;
JPanel textPanel, buttonpanel;
ScientificCalculator()
{
cont = getContentPane();
cont.setLayout(new BorderLayout());
JPanel textpanel = new JPanel();
tfield = new JTextField(25);
tfield.setHorizontalAlignment(SwingConstants.RIGHT);
tfield.addKeyListener(new KeyAdapter() {
public void keyTyped(KeyEvent keyevent) {
char c = keyevent.getKeyChar();
if (c >= '0' && c <= '9') {
}
else
{
keyevent.consume();
}
}
});
textpanel.add(tfield);
buttonpanel = new JPanel();
buttonpanel.setLayout(new GridLayout(8, 4, 2, 2));
boolean t = true;
mr = new JButton("MR");
buttonpanel.add(mr);
mr.addActionListener(this);
mc = new JButton("MC");
buttonpanel.add(mc);
mc.addActionListener(this);
mp = new JButton("M+");
buttonpanel.add(mp);
mp.addActionListener(this);
mm = new JButton("M-");
buttonpanel.add(mm);
```

```
mm.addActionListener(this);
b1 = new JButton("1");
buttonpanel.add(b1);
b1.addActionListener(this);
b2 = new JButton("2");
buttonpanel.add(b2);
b2.addActionListener(this);
b3 = new JButton("3");
buttonpanel.add(b3);
b3.addActionListener(this);
b4 = new JButton("4");
buttonpanel.add(b4);
b4.addActionListener(this);
b5 = new JButton("5");
buttonpanel.add(b5);
b5.addActionListener(this);
b6 = new JButton("6");
buttonpanel.add(b6);
b6.addActionListener(this);
b7 = new JButton("7");
buttonpanel.add(b7);
b7.addActionListener(this);
b8 = new JButton("8");
buttonpanel.add(b8);
b8.addActionListener(this);
b9 = new JButton("9");
buttonpanel.add(b9);
b9.addActionListener(this);
zero = new JButton("0");
buttonpanel.add(zero);
zero.addActionListener(this);
plus = new JButton("+");
buttonpanel.add(plus);
plus.addActionListener(this);
min = new JButton("-");
buttonpanel.add(min);
min.addActionListener(this);
mul = new JButton("*");
buttonpanel.add(mul);
```

```
mul.addActionListener(this);
div = new JButton("/");
div.addActionListener(this);
buttonpanel.add(div);
addSub = new JButton("+/-");
buttonpanel.add(addSub);
addSub.addActionListener(this);
dot = new JButton(".");
buttonpanel.add(dot);
dot.addActionListener(this);
eq = new JButton("=");
buttonpanel.add(eq);
eq.addActionListener(this);
rec = new JButton("1/x");
buttonpanel.add(rec);
rec.addActionListener(this);
sqrt = new JButton("Sqrt");
buttonpanel.add(sqrt);
sqrt.addActionListener(this);
log = new JButton("log");
buttonpanel.add(log);
log.addActionListener(this);
sin = new JButton("SIN");
buttonpanel.add(sin);
sin.addActionListener(this);
cos = new JButton("COS");
buttonpanel.add(cos);
cos.addActionListener(this);
tan = new JButton("TAN");
buttonpanel.add(tan);
tan.addActionListener(this);
pow2 = new JButton("x^2");
buttonpanel.add(pow2);
pow2.addActionListener(this);
pow3 = new JButton("x^3");
buttonpanel.add(pow3);
pow3.addActionListener(this);
exp = new JButton("Exp");
exp.addActionListener(this);
```

```
buttonpanel.add(exp);
fac = new JButton("n!");
fac.addActionListener(this);
buttonpanel.add(fac);
clr = new JButton("AC");
buttonpanel.add(clr);
clr.addActionListener(this);
cont.add("Center", buttonpanel);
cont.add("North", textpanel);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
public void actionPerformed(ActionEvent e)
{
String s = e.getActionCommand();
if (s.equals("1"))
{
if (z == 0)
{
tfield.setText(tfield.getText() + "1");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "1");
z = 0;
}
}
if (s.equals("2")) {
if (z == 0) {
tfield.setText(tfield.getText() + "2");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "2");
z = 0;
}
}
if (s.equals("3")) {
```

```
if (z == 0) {
tfield.setText(tfield.getText() + "3");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "3");
z = 0;
}
}
if (s.equals("4")) {
if (z == 0) {
tfield.setText(tfield.getText() + "4");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "4");
z = 0;
}
}
if (s.equals("5")) {
if (z == 0) {
tfield.setText(tfield.getText() + "5");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "5");
z = 0;
}
}
if (s.equals("6")) {
if (z == 0) {
tfield.setText(tfield.getText() + "6");
}
else
 {
tfield.setText("");
```

```
tfield.setText(tfield.getText() + "6");
z = 0;
}
}
if (s.equals("7")) {
if (z == 0) {
tfield.setText(tfield.getText() + "7");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "7");
z = 0;
}
}
if (s.equals("8")) {
if (z == 0) {
tfield.setText(tfield.getText() + "8");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "8");
z = 0;
}
}
if (s.equals("9")) {
if (z == 0) {
tfield.setText(tfield.getText() + "9");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "9");
z = 0;
}
}
if (s.equals("0"))
{
```

```
if (z == 0) {
tfield.setText(tfield.getText() + "0");
}
else
{
tfield.setText("");
tfield.setText(tfield.getText() + "0");
z = 0;
}
}
if (s.equals("AC")) {
tfield.setText("");
x = 0;
y = 0;
z = 0;
}
if (s.equals("log"))
{
if (tfield.getText().equals("")) {
tfield.setText("");
}
else
 {
a = Math.log(Double.parseDouble(tfield.getText()));
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
if (s.equals("1/x")) {
if (tfield.getText().equals("")) {
tfield.setText("");
}
else
{
a = 1 / Double.parseDouble(tfield.getText());
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
```

```
if (s.equals("Exp")) {
if (tfield.getText().equals("")) {
tfield.setText("");
}
else
{
a = Math.exp(Double.parseDouble(tfield.getText()));
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
if (s.equals("x^2")) {
if (tfield.getText().equals("")) {
tfield.setText("");
}
else
{
a = Math.pow(Double.parseDouble(tfield.getText()), 2);
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
if (s.equals("x^3")) {
if (tfield.getText().equals("")) {
tfield.setText("");
}
else
{
a = Math.pow(Double.parseDouble(tfield.getText()), 3);
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
if (s.equals("+/-")) {
if (x == 0) {
tfield.setText("-" + tfield.getText());
x = 1;
}
else
```

```
{
tfield.setText(tfield.getText());
}
}
if (s.equals(".")) {
if (y == 0) {
tfield.setText(tfield.getText() + ".");
y = 1;
}
else
{
tfield.setText(tfield.getText());
}
}
if (s.equals("+"))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
temp = 0;
ch = '+';
}
else
{
temp = Double.parseDouble(tfield.getText());
tfield.setText("");
ch = '+';
y = 0;
x = 0;
}
tfield.requestFocus();
}
if (s.equals("-"))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
temp = 0;
ch = '-';
```

```
}
 else
 {
x = 0;
y = 0;
temp = Double.parseDouble(tfield.getText());
tfield.setText("");
ch = '-';
}
tfield.requestFocus();
}
if (s.equals("/")) {
if (tfield.getText().equals(""))
{
tfield.setText("");
temp = 1;
ch = '/';
}
else
{
x = 0;
y = 0;
temp = Double.parseDouble(tfield.getText());
ch = '/';
tfield.setText("");
}
tfield.requestFocus();
}
if (s.equals("*")) {
if (tfield.getText().equals(""))
{
tfield.setText("");
temp = 1;
ch = '*';
}
else
{
x = 0;
y = 0;
```

```
temp = Double.parseDouble(tfield.getText());
ch = '*';
tfield.setText("");
}
tfield.requestFocus();
}
if (s.equals("MC"))
{
m1 = 0;
tfield.setText("");
}
if (s.equals("MR"))
{
tfield.setText("");
tfield.setText(tfield.getText() + m1);
}
if (s.equals("M+"))
{
if (k == 1) {
m1 = Double.parseDouble(tfield.getText());
k++;
}
else
{
m1 += Double.parseDouble(tfield.getText());
tfield.setText("" + m1);
}
}
if (s.equals("M-"))
{
if (k == 1) {
m1 = Double.parseDouble(tfield.getText());
k++;
}
else
 {
m1 -= Double.parseDouble(tfield.getText());
tfield.setText("" + m1);
}
```

```
}
if (s.equals("Sqrt"))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
}
else
{
a = Math.sqrt(Double.parseDouble(tfield.getText()));
tfield.setText("");
field.setText(tfield.getText() + a);
}
}
if (s.equals("SIN"))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
}
else
{
a = Math.sin(Double.parseDouble(tfield.getText()));
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
if (s.equals("COS"))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
}
else
{
a = Math.cos(Double.parseDouble(tfield.getText()));
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
```

```
}
if (s.equals("TAN")) {
if (tfield.getText().equals("")) {
tfield.setText("");
}
else
{
a = Math.tan(Double.parseDouble(tfield.getText()));
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
if (s.equals("="))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
}
else
{
temp1 = Double.parseDouble(tfield.getText());
switch (ch)
{
case '+':
result = temp + temp1;
break;
case '-':
result = temp - temp1;
break;
case '/':
result = temp / temp1;
break;
case '*':
result = temp * temp1;
break;
}
tfield.setText("");
tfield.setText(tfield.getText() + result);
z = 1;
```

```
}
}
if (s.equals("n!"))
{
if (tfield.getText().equals(""))
{
tfield.setText("");
}
else
{
a = fact(Double.parseDouble(tfield.getText()));
tfield.setText("");
tfield.setText(tfield.getText() + a);
}
}
tfield.requestFocus();
}
double fact(double x)
{
int er = 0;
if (x < 0)
{
er = 20;
return 0;
}
double i, s = 1;
for (i = 2; i <= x; i += 1.0)
s *= i;
return s;
}
public static void main(String args[])
{
try
{
UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
}
catch (Exception e)
{
}
```

```
ScientificCalculator f = new ScientificCalculator();
f.setTitle("ScientificCalculator");
f.pack();
f.setVisible(true);
}
}
```

Output: