INTRODUCTION TO CLOUD COMPUTING

What is Cloud Computing?

Cloud computing means that instead of all the computer hardware and software you're using sitting on your desktop, or somewhere inside your company's network, it's provided for you as a service by another company and accessed over the Internet, usually in a completely seamless way. Exactly where the hardware and software is located and how it all works doesn't matter to you, the user—it's just somewhere up in the nebulous "cloud" that the Internet represents.

Cloud computing is a buzzword that means different things to different people. For some, it's just another way of describing IT (information technology) "outsourcing"; others use it to mean any computing service provided over the Internet or a similar network; and some define it as any bought-in computer service you use that sits outside your firewall.

Types of cloud computing

IT people talk about three different kinds of cloud computing, where different services are being provided for you. Note that there's a certain amount of vagueness about how these things are defined and some overlap between them.

1. **Infrastructure as a Service (IaaS)** means you're buying access to raw computing hardware over the Net,such as servers or storage. Since you buy what you need and pay-as-you-go, this is often referred to as utility computing. Ordinary web hosting is a simple example of IaaS: you pay a monthly subscription or a per-megabyte/gigabyte fee to have a hosting company serve up files for your website from their servers.

2. **Software as a Service (SaaS)** means you use a complete application running on someone else's system.Web-based email and Google Documents are perhaps the best-known examples. Zoho is another well-known SaaS provider offering a variety of office applications online.

3. **Platform as a Service (PaaS)** means you develop applications using Web-based tools so they run onsystems software and hardware provided by another company. So, for example, you might develop your own ecommerce website but have the whole thing, including the shopping cart, checkout, and payment mechanism running on a merchant's server. App Cloud (from salesforce.com) and the Google App Engine are examples of PaaS.

Task-1

Programs on SOFTWARE AS A SERVICE

1. Create an word document of your class time table and store locally and on the cloud with doc,and pdf format . (use www.zoho.com and docs.google.com)

AIM: To create a word document of class time table and store locally on the cloud with using ZOHO and Google docs.

Steps:

1. Go to docs.google.com
2. Select main menu → Click Docs option and select Blank Document.
3. Write the title name is Annamacharya Institute of &Technology & Sciences, Tirupati (A1 address)
4. Write the sub title name is IV.B.Tech TIME TABLE FOR ACADEMIC YEAR 2019-2020 at (A2 address)

5. Write the date is =Today() at G3 address
6. Write the fields A4 address to J4 address location

I	II		III	IV		V	VI	VI I
9:009:50	9:50 10:40	10:40 - 10:50	10:50 11:40	11:40-12:30	12:30-1:10	1:20-2:10	2:10-3:00	3:10-4:00

In the above format the DAY/TIME will appear as above and go to Format cells and select Alignment .In that under the Text control select tickon wrap text then click on OK button

7. Write the MON at A5 address location and drag cell with auto fill option up sat(A10 address)
8. Write subject name GCC fill all cellsif you have lab hour's like GCC LAB, Enter the lab name leave 2 cells.
9. If any break or lunch leave the cell

FormattingStyles:

1. Select the cell A1 to J1, click on HOME then click the merge and center button, set font size to 16.
2. Select the cell A2 to J2, click on HOME then click the merge and center button, set font size is 12.
3. Select the cell address G5 to G10, click on merge and center button, then type "LUNCH" and click on orientation button &select "Vertical Text".
4. Fill the cellswith particular subjects according toschedule.
5. Write "GCC" in B5, "Maths" in C5 and so on.
6. Select D5&D6, click on merge &center button then type "BREAK", click on orientation & select ""Vertical Text".
7. Select H5, I5,J5 cells & type "GCC LAB".
8. Repeat above steps for the remaining cells B5 to J5.

Create a new document

You can create a new document right in Docs or in Google Drive.

In Docs, click Create new document.

In Drive, click New > Google Docs > Blank document or From a template.

Import and convert old documents to Docs

If you have existing text documents, such as Microsoft® Word® or Adobe®PDF files, you can import and convert them to Docs.

- Go to Drive.
- Click New > File Upload and choose a text document from your computer. Supported files include .doc, .docx, .dot, .html, plain text (.txt), .odt, and .rtf.
- Right-click the file you want to convert and select Open with > Google Docs.

Converting your document from another program creates a copy of your original file in Docs format. You can then edit it in your browser like any other document.

Share documents

1. Open the file you want to share.
2. Click Share.
3. Enter the email addresses or Google Groups you want to share with.

Note: If you can't add people outside your company, see your G Suite administrator.

4. Choose what kind of access you want to grant people:
   - **Can edit**—Collaborators can add and edit content as well as add comments.
   - **Can comment**—Collaborators can add comments, but not edit content.
   - **Can view**—People can view the file, but not edit or add comments.

   Click Send.

Everyone you shared the document with receives an email with a link to the document.

OUTPUT:

Result:

Task-2

2. Create a spread sheet which contains employee salary information and calculate gross and total sal using the formula

DA=10% OF BASIC
HRA=30% OF BASIC
PF=10% OF BASIC IF BASIC<=3000 12% OF BASIC IF BASIC>3000
TAX=10% OF BASIC IF BASIC<=1500
=11% OF BASIC IF BASIC>1500 AND BASIC<=2500
=12% OF BASIC IF BASIC>2500
(use www.zoho.com and docs.google.com)
NET_SALARY=BASIC_SALARY+DA+HRA-PF-TAX

AIM: To create a spread Sheet contains employee salary information .

Steps:

1. Navigate to Google's home page and click on Google Docs. Sign in if you have an account.
2. Select main menu →Click sheets option and select Blank Document.
3. Enter Employee details from A1 to J1.
4. At last select A1 to J10 cellsand keep borders by clicking on Borders Button and select All Borders.
5. Write formulas in formula box "fx".

S.NO	EMP NAME	Designation	Basic Salary	DA	HRA	PF	TAX	Net Salary	Gross Salary

S.NO	EMP NAME	Desigantion	Salary
1	A	Manager	70000
2	B	Project Manager	60000
3	C	Project Lead	50000
4	D	Team lead	40000
5	E	Sr.Developer	30000
6	F	Developer 1	20000
7	G	Developer 2	20000
8	H	Developer 3	20000
9	I	Tester	20000
10	J	Production	20000

Find DA,HRA,PF,TAX and NET SALARY

6. DA=10% OF BASIC

   Ex: DA=Basic Salary*10/100

7. HRA=30% OF BASIC
   Ex: HRA= Basic Salary*30/100
8. PF=10% OF BASIC IF BASIC<=3000 12% OF BASIC IF BASIC>3000
   Ex: PF= =if(D2<=3000,D2*10/100,D2*12/100) (D2=Basic Salary)
9. TAX=10% OF BASIC IF BASIC<=1500=11% OF BASIC IF BASIC>1500 AND BASIC<=2500=12% OF BASIC IF BASIC>2500
   Ex:
   TAX
   =If(D2<=1500,D2*10/100,if(D2>1500&D2<=2500,D2*11/100,if(D2>2500,D2*12/100)))
10. Net Salary=Salary+DA+HRA-PF-TAX
    Ex: Net Salary=D2+E2+F2-G2-H2
11. Gross Salary= Salary+DA+HRA+PF+TAX
12. At last save the file by clicking "Make a copy" or " Down load as" or "Email as attachment", give file name as Time Table then press "ok".

OUTPUT:

Result:

Task-3

3. Prepare a PPT on cloud computing – introduction, models, services, and architecture
Ppt should contain explanations, images and at least 20 pages
(use www.zoho.com and docs.google.com)

Aim: To create a ppt on cloud computing using ZOHO & Google docs.

PROCEDURE:

Inserting a Slide:

Toinsert a new slide…

- From the **Home** tab, select "New Slide"
- As well, you can right-click between any 2 slides in the preview frame located on the left-hand side of the window
- Normal view and select "New Slide"

Deleting a Slide:

To delete a slide…

- As well, you can right-click on any slide in the preview frame located on the left-hand side of the Normal view and select "Delete Slide"

Inserting Pictures:

To insert pictures…

- From the **Insert** tab you can insert pictures from your computer, images, clip art, shapes, etc…
- Also, when you have a blank slide (or parts of a blank slide), you can also click on options within these blank compartments to insert pictures.

Inserting Charts:

To insert charts…

- From the **Insert** tab you can insert pictures from your computer, clip art, shapes, etc…
- Also, when you have a blank slide (or parts of a blank slide), you can also click on options within these blank compartments to insert charts.

Steps:

1. Navigate to Google's home page and click on Google Docs. Sign in if you have an account.
2. Select main menu→click option →select Blank document.
3. Ppt should contain explanations, images and at least 20 pages
4. In first slide enter title name and sub-titles
5. In second slide, write introduction in title box and information in text box as point wise
6. In next slides, write models, services, and Architecture

Formatting styles

Apply Formatting Styles (font size, font style, font type, and color), Bullets and Numbering for the title and text from **Format tab**.

7. From the **Insert tab**, write the Slide number
8. Taka the picture from the Insert tab.
9. Select the **theme** and **Background** Color for the slides.
10. Select **Transition** to this slide and **Transition sound.**
11. From the insert tab, apply Animation for the Title, Text and picture from Animations box. If you want more go to the custom animation tab.
12. At last save the file by clicking "Make a copy" or " Down load as" or "Email as attachment", give file name as Time Table then press "ok".

OUTPUT:

Result:

Task-4

4. Create your resume in a neat format using google and zoho cloud
AIM:

PROCUDURE

1. Navigate to Google's home page and click on Google Docs. Sign in if you have an account.
2. Click on "Template Gallery" to see a list of template options. There are multiple letter formats you can use for your cover letter, and multiple resume formats as well. You can find additional templates by clicking the "More" arrows and scrolling through the options.
3. Select a template you like. Click on the template you want to use, and it will open in a new window.
4. Personalize the template with your information. The templates are filled with *lorem ipsum* dummy text. Simply click where you want to edit, delete the dummy text and
5. The template name appears at the top of your screen, above the toolbars.

   - For example, if you selected the basic Resume template, "Resume" appears above the toolbars.
   - To rename the file, simply click on the template name. It opens in a textbox for editing. After you've changed the name, click out of the textbox, and your new name is saved.
   - If you are making multiple versions of your resume or cover letter, be sure to label each one with a specific title that will help you remember which is which (such as the title of the job you're applying for).

6. Once you've completed your basic resume but want to customize it for a particular job application, make a copy of the resume or cover letter through the "File" menu and give it a different name. Google Docs automatically saves your new file with your other docs.
7. At last save the file by clicking "Make a copy" or " Down load as" or "Email as attachment", give file name as Time Table then press "ok".

Storing and Sharing Your Google Docs Resume or Cover Letter

Once you have created a final version of your resume or cover letter, you'll be able to store it on Google Docs, update it, use it to apply for jobs, and share it with hiring managers and recruiters.

You can also choose to store it on Google Drive, an organizational system in which you can create, upload, edit, save, and share documents. Keep in mind that many hiring managers prefer to receive resumes as attachments in an email or documents uploaded directly to their corporate job site, rather than shared via link.

If you're applying online, follow the instructions in the job posting. If you're sending your resume directly to a recruiter or hiring manager, through a networking contact, ask your connection about the preferred method of delivery.

OUTPUT:

Result:

GRID COMPUTING PROGRAMS USING USE GLOBUS TOOLKIT OR EQUIVALENT

1. Develop a new Web Service for Calculator.

OBJECTIVE:

To develop a new Web service for Calculator applications.

PROCEDURE:

When you start Globus toolkit container, there will be number of services starts up. The service for this task will be a simple Math service that can perform basic arithmetic for a client.

The Math service will access a resource with two properties:

1. An integer value that can be operated upon by the service

2. A string values that holds string describing the last operation

The service itself will have three remotely accessible operations that operate upon

value:

(a) add, that adds *a* to the resource property *value*.

(b) subtract that subtracts *a* from the resource property *value*.

(c) getValueRP that returns the current value of *value*.

Usually, the best way for any programming task is to begin with an overall description of what you want the code to do, which in this case is the service interface. The service interface describes how what the service provides in terms of names of operations, their arguments and return values. A Java interface for our service is:

```
public interface Math {

public void add(int a);

public void subtract(int a);

public int getValueRP();

}
```

It is possible to start with this interface and create the necessary WSDL file using the standard Web service tool called Java2WSDL. However, the WSDL file for GT 4 has to include details of resource properties that are not given explicitly in the interface above. Hence, we will provide the WSDL file.

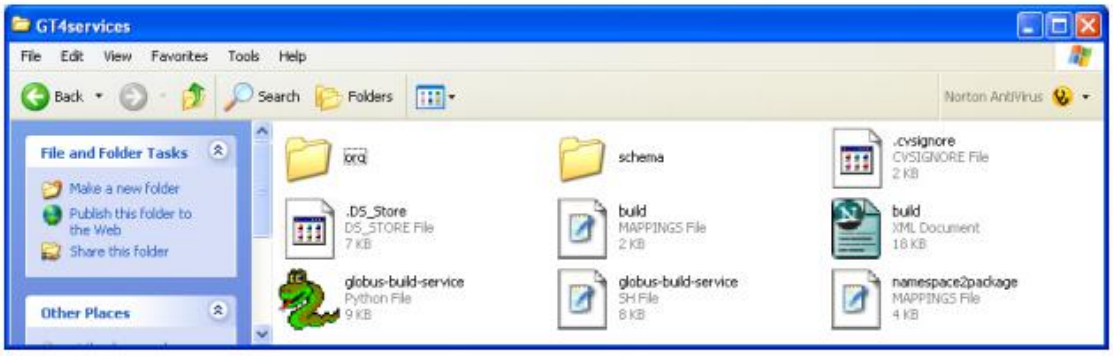
Step 1 Getting the Files

All the required files are provided and comes directly from [1]. The MathService source code files can be found from http://www.gt4book.com

(http://www.gt4book.com/downloads/gt4book-examples.tar.gz)

A Windows zip compressed version can be found at

http://www.cs.uncc.edu/~abw/ITCS4146S07/gt4book-examples.zip. Download and uncompress the file into a directory called **GT4services**. Everything is included (the java source WSDL and deployment files, etc.):



WSDL service interface description file -- ***The WSDL service interface description***

file is provided within the GT4services folder at:

GT4Services\schema\examples\MathService_instance\Math.wsdl

This file, and discussion of its contents, can be found in Appendix A. Later on we will need to modify this file, but first we will use the existing contents that describe the Math service above.

Service code in Java -- For this assignment, both the code for service operations and for the resource properties are put in the same class for convenience. More complex services and resources would be defined in separate classes. The Java code for the service and its resource properties is located within the GT4services folder at:

GT4services\org\globus\examples\services\core\first\impl\MathService.java.

Deployment Descriptor -- The deployment descriptor gives several different important sets of information about the service once it is deployed. It is located within the **GT4services** folder at:

GT4services\org\globus\examples\services\core\first\deploy-server.wsdd.

Step 2 – Building the Math Service

It is now necessary to package all the required files into a GAR (Grid Archive) file. The build tool ant from the Apache Software Foundation is used to achieve this as shown overleaf:

Generating a GAR file with Ant (from http://gdp.globus.org/gt4-

tutorial/multiplehtml/ch03s04.html)

Ant is similar in concept to the Unix make tool but a java tool and XML based.

Build scripts are provided by Globus 4 to use the ant build file. The windows version of the build script for MathService is the Python file called **globus-build-service.py**, which held in the **GT4services** directory. The build script takes one argument, the name of your service that you want to deploy. To keep with the naming convention in [1], this service will be called **first**.

In the *Client Window*, run the build script from the **GT4services** directory with:

globus-build-service.py first

The output should look similar to the following:

Buildfile: build.xml

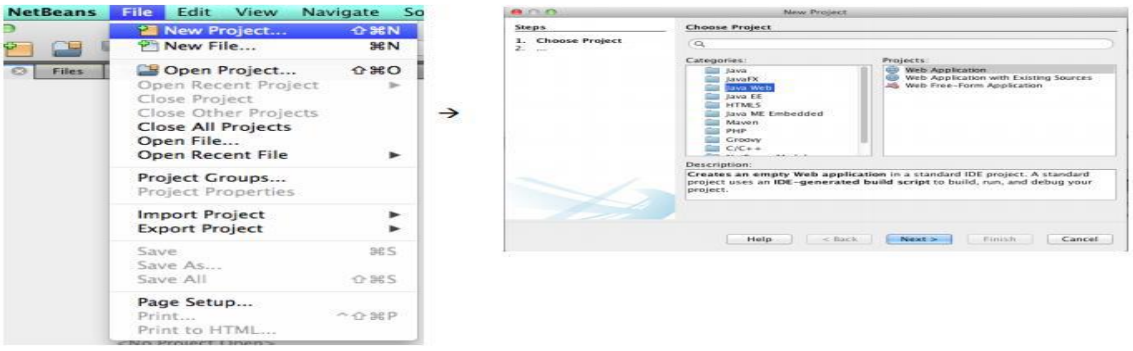
Creating a Web Service in Java using NetBeans IDE

This document provides step-by-step instructions to create and deploy a web service in Java using NetBeans IDE and GlassFish 4.0. In the project, we will create a calculation service.

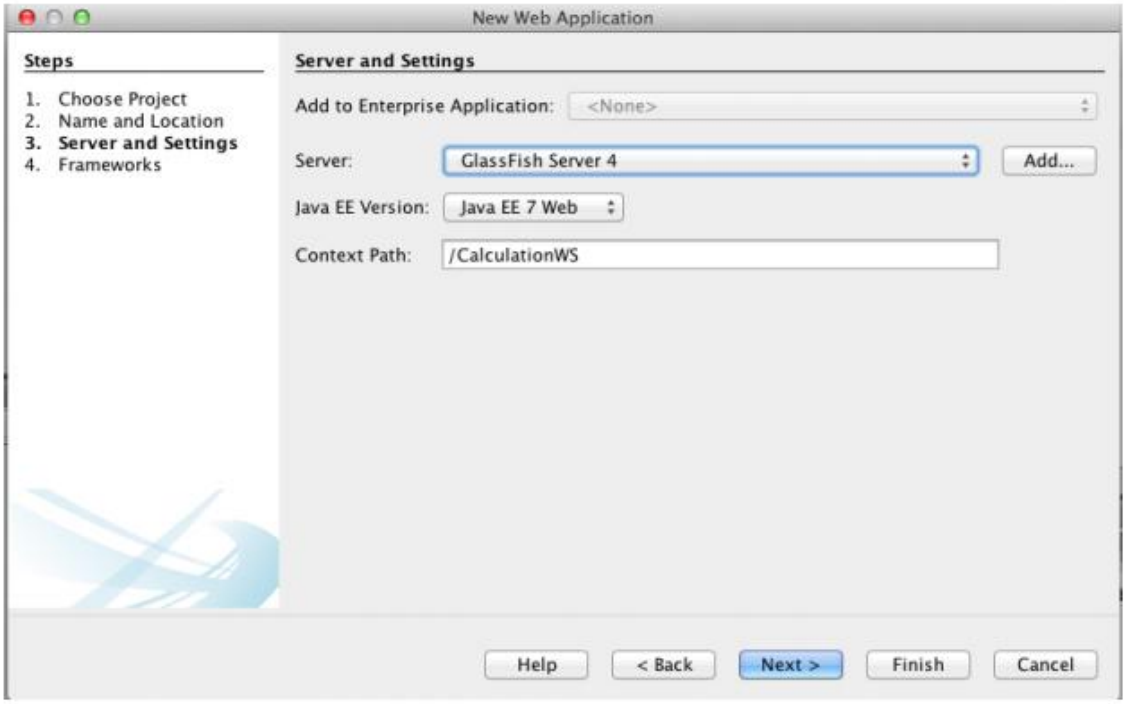
Step 1: Create a Java Web Project

Open NetBeans IDE

Click on New Project and choose Java Web → Web Application



Enter the Project Name: CalculationWS, using the default settings and then click on "Finish".

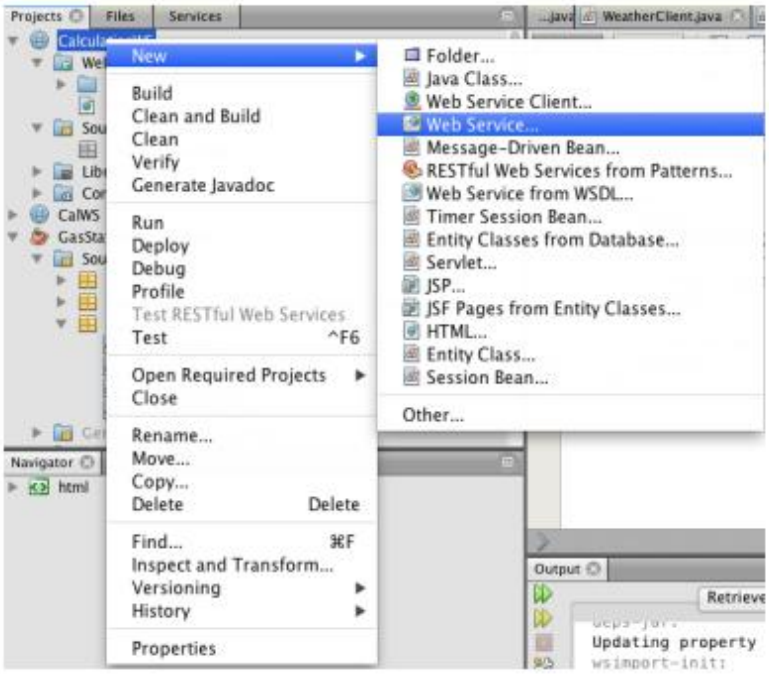


Now the Project has been created.

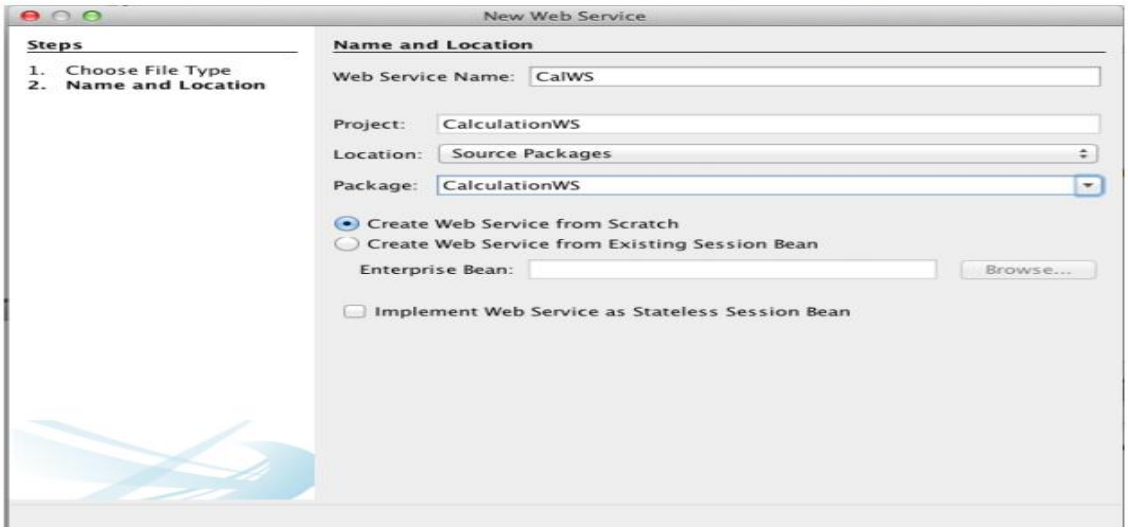
Step 2: Create a Web Service

Now go to the Project Tree Structure on the left side of the window.

Right click on the project and select "New" and then choose "Web Service"



Specify web service name "CalWS" and package name "CalculationWS". Click on "Finish".



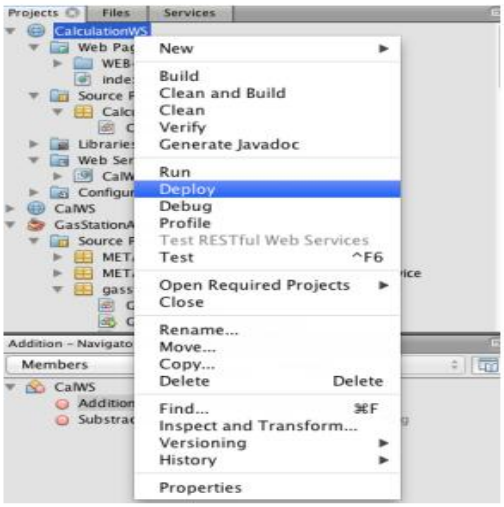
Open CalWS.java file, replace the original hello() function with the following code:

```
@WebMethod(operationName = "Addition")
   public String Addition(@WebParam(name = "value1") String value1,@WebParam(name = "value2") String value2
) {
      float value=Float.valueOf(value1)+Float.valueOf(value2);
      return (Float.toString(value));
   }
```

Now the web service is created.

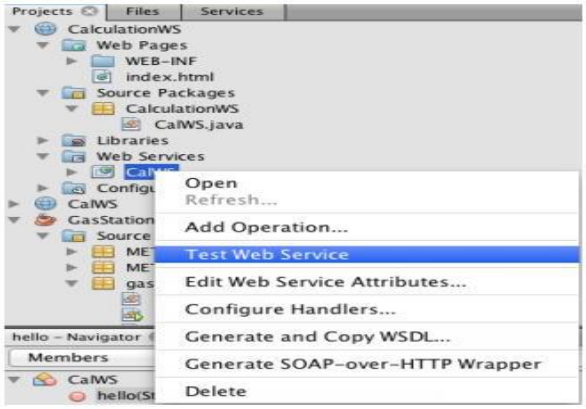
Step 3: Deploy and Test Web Service

Right click on the project and select "Deploy"



This is to deploy all the web services in this project. If success, you will see:

To test the web service, right click on the service and select "Test Web Service"



You will see:

Result:

Right Click on the project and select "Clean and Build", a war file will be automatically generated under "dist" sub-directory.

GRID COMPUTING PROGRAMS USING GRIDSIM

GridSim allows modeling and simulation of entities in parallel and distributed computing systems such as users, applications, resources, and resource brokers/schedulers for design and evaluation of scheduling algorithms. The resource brokers use scheduling algorithms or policies for mapping jobs to resources to optimize system or user objectives depending on their goals.

Overview of GridSim functionalities:

- Incorporates failures of Grid resources during runtime.
- New allocation policy can be made and integrated into the GridSim Toolkit, by extending from AllocPolicy class.
- Has the infrastructure or framework to support advance reservation of a grid system.
- Incorporates a functionality that reads workload traces taken from supercomputers for simulating a realistic grid environment.
- Incorporates an auction model into GridSim.
- Incorporates a data grid extension into GridSim.
- Incorporates a network extension into GridSim. Now, resources and other entities can be linked in a network topology.
- Incorporates multiple regional Grid Information Service (GIS) entities connected in a network topology. Hence, you can simulate an experiment with multiple Virtual Organizations (VOs).
- Adds ant build file to compile GridSim source files.

SOFTWARE REQUIREMENTS:

1. Gridsim Toolkit
2. JDK (java development kit) new version.
3. Eclipse for java developers.

Installation steps:

1.Install JDK toolkit

2.Set path for JDK toolkit Path=C:/jdk1.8/binClasspath=C:/jdk1.8/jre/lib/rt.jar;

3.Download GridSim 5.2

4.Extract GridSim into one folder.

5.Set path = C:/gridsim/bin;

6.SetClasspath = C:/gridsim/jar/*;

7.Set Classpath = C:/gridsim/examples;

8.Set variable GridSim=C:/gridsim

1. Program to creates one Grid resource with three machines.

Aim: To creates one Grid resource with three machines.

Description:

GridSim creates modelling and scheduling. The speciality of gridsim tool kit is that it includes the network buffer management policies which cannot be simulated in Optorsim, Microsim etc. Grid computing having the high variable environment. The grid resource is very important in gridsim and it is flexibility to implement various scheduling algorithms.

User – Each instance of the User entity represents a Grid user. Each user may differ from the rest of users .

Source code:

```
package gridsim.example01;

	import java.util.Calendar;
	import java.util.LinkedList;

	import gridsim.*;


	class Example1
	{

	    @SuppressWarnings("unused")
	       public static void main(String[] args)
	    {
	        System.out.println("Starting example of how to create one Grid " +
	                "resource");

	        try
	        {

	            int num_user = 0;
	            Calendar calendar = Calendar.getInstance();
	            boolean trace_flag = true; // mean trace GridSim events/activities

	            // list of files or processing names to be excluded from any
	            //statistical measures
	            String[] exclude_from_file = { "" };
	            String[] exclude_from_processing = { "" };


	            String report_name = null;

	            // Initialize the GridSim package
	            System.out.println("Initializing GridSim package");
	            GridSim.init(num_user, calendar, trace_flag, exclude_from_file,
	                    exclude_from_processing, report_name);



	            // Second step: Create one Grid resource
	            GridResource gridResource = createGridResource();
	            System.out.println("Finish the 1st example");
```

```
    }
    catch (Exception e)
    {
        e.printStackTrace();
        System.out.println("Unwanted error happens");
    }
}



@SuppressWarnings("deprecation")
   private static GridResource createGridResource()
{
    System.out.println("Starting to create one Grid resource with " +
            "3 Machines ...");


    MachineList mList = new MachineList();
    System.out.println("Creates a Machine list");


    int mipsRating = 377;
    mList.add( new Machine(0, 4, mipsRating));   // First Machine
    System.out.println("Creates the 1st Machine that has 4 PEs and " +
            "stores it into the Machine list");


    mList.add( new Machine(1, 4, mipsRating));   // Second Machine
    System.out.println("Creates the 2nd Machine that has 4 PEs and " +
            "stores it into the Machine list");

    mList.add( new Machine(2, 2, mipsRating));   // Third Machine
    System.out.println("Creates the 3rd Machine that has 2 PEs and " +
            "stores it into the Machine list");


    String arch = "Sun Ultra";      // system architecture
    String os = "Solaris";          // operating system
    double time_zone = 9.0;         // time zone this resource located
    double cost = 3.0;              // the cost of using this resource

    ResourceCharacteristics resConfig = new ResourceCharacteristics(
            arch, os, mList, ResourceCharacteristics.TIME_SHARED,
            time_zone, cost);

    System.out.println();
    System.out.println("Creates the properties of a Grid resource and " +
            "stores the Machine list");

    // 5. Finally, we need to create a GridResource object.
    String name = "Resource_0";         // resource name
    double baud_rate = 100.0;           // communication speed
    long seed = 11L*13*17*19*23+1;
    double peakLoad = 0.0;        // the resource load during peak hour
    double offPeakLoad = 0.0;     // the resource load during off-peak hr
    double holidayLoad = 0.0;     // the resource load during holiday
```

```
                // incorporates weekends so the grid resource is on 7 days a week
                LinkedList<Integer> Weekends = new LinkedList<Integer>();
                Weekends.add(new Integer(Calendar.SATURDAY));
                Weekends.add(new Integer(Calendar.SUNDAY));

                // incorporates holidays. However, no holidays are set in this example
                LinkedList<Integer> Holidays = new LinkedList<Integer>();

                GridResource gridRes = null;
                try
                {
gridRes = new GridResource(name, baud_rate, seed,resConfig, peakLoad, offPeakLoad,
holidayLoad, Weekends, Holidays);
                }
                catch (Exception e) {
                    e.printStackTrace();
                }

                System.out.println("Finally, creates one Grid resource and stores " +
                        "the properties of a Grid resource");

                return gridRes;
            }
        }
```

Output:

```
Starting example of how to create one Grid resource
Initializing GridSim package
Initialising...
Starting to create one Grid resource with 3 Machines ...
Creates a Machine list
Creates the 1st Machine that has 4 PEs and stores it into the Machine list
Creates the 2nd Machine that has 4 PEs and stores it into the Machine list
Creates the 3rd Machine that has 2 PEs and stores it into the Machine list

Creates the properties of a Grid resource and stores the Machine list
Finally, creates one Grid resource and stores the properties of a Grid resource
Finish the 1st example
```

2. Program to create one or more Grid users. A Grid user contains one or more Gridlets.

Aim: To create a java program one or more Grid users and it contains one or more Gridlets.

Description:

The grid systems may have their own grid login ID separate from the one on the operating system. A grid login is usually more convenient for grid users.

It eliminates the ID matching problems among different machines. To the user, it makes the grid look more like one large virtual computer rather than a collection of individual machines.

User – Each instance of the User entity represents a Grid user. Each user may differ from the rest of users .

Source code:

```
package gridsim.example01;

import java.util.*;
import gridsim.*;


class Example2
{

   public static void main(String[] args)
   {
      System.out.println("Starting example of how to create Grid users");
      System.out.println();

      try
      {
         // Creates a list of Gridlets
         GridletList list = createGridlet();
         System.out.println("Creating " + list.size() + " Gridlets");

         ResourceUserList userList = createGridUser(list);
         System.out.println("Creating " + userList.size() + " Grid users");

         // print the Gridlets
         printGridletList(list);

         System.out.println("Finish the example");
      }
      catch (Exception e)
      {
         e.printStackTrace();
         System.out.println("Unwanted error happens");
      }
   }



   private static GridletList createGridlet()
   {
      // Creates a container to store Gridlets
      GridletList list = new GridletList();

      // We create three Gridlets or jobs/tasks manually without the help
```

```
// of GridSimRandom
int id = 0;
double length = 3500.0;
long file_size = 300;
long output_size = 300;
Gridlet gridlet1 = new Gridlet(id, length, file_size, output_size);
id++;
Gridlet gridlet2 = new Gridlet(id, 5000, 500, 500);
id++;
Gridlet gridlet3 = new Gridlet(id, 9000, 900, 900);

// Store the Gridlets into a list
list.add(gridlet1);
list.add(gridlet2);
list.add(gridlet3);

// We create 5 Gridlets with the help of GridSimRandom and
// GriSimStandardPE class
Random random = new Random();

// sets the PE MIPS Rating
GridSimStandardPE.setRating(100);

// creates 5 Gridlets
int count = 5;
double min_range = 0.10;
double max_range = 0.50;
for (int i = 1; i < count+1; i++)
{

   length = GridSimStandardPE.toMIs(random.nextDouble()*output_size);


   file_size = (long) GridSimRandom.real(100, min_range, max_range,
                     random.nextDouble());


   output_size = (long) GridSimRandom.real(250, min_range, max_range,
                     random.nextDouble());

   // creates a new Gridlet object
   Gridlet gridlet = new Gridlet(id + i, length, file_size,
                     output_size);
```

```
            // add the Gridlet into a list
            list.add(gridlet);
        }

        return list;
    }


    private static ResourceUserList createGridUser(GridletList list)
    {
        ResourceUserList userList = new ResourceUserList();

        userList.add(0);    // user ID starts from 0
        userList.add(1);
        userList.add(2);

        int userSize = userList.size();
        int gridletSize = list.size();
        int id = 0;

        // assign user ID to particular Gridlets
        for (int i = 0; i < gridletSize; i++)
        {
            if (i != 0 && i % userSize == 0)
                id++;

            ( (Gridlet) list.get(i) ).setUserID(id);
        }

        return userList;
    }


    private static void printGridletList(GridletList list)
    {
        int size = list.size();
        Gridlet gridlet;

        String indent = "    ";
        System.out.println();
        System.out.println("Gridlet ID" + indent + "User ID" + indent +
                "length" + indent + " file size" + indent +
                "output size");
```

```
    for (int i = 0; i < size; i++)
    {
       gridlet = (Gridlet) list.get(i);
       System.out.println(indent + gridlet.getGridletID() + indent +
            indent + indent + gridlet.getUserID() + indent + indent +
            (int) gridlet.getGridletLength() + indent + indent +
            (int) gridlet.getGridletFileSize() + indent + indent +
            (int) gridlet.getGridletOutputSize() );
    }
  }
  } // end class
```

Output:

```
Starting example of how to create Grid users

Creating 8 Gridlets
Creating 3 Grid users

Gridlet ID    User ID    length     file size    output size
    0             0        3500         300         300
    1             0        5000         500         500
    2             0        9000         900         900
    3             1        15759         120         233
    4             1        14003         131         303
    5             1        7038         92         330
    6             2        4829         138         289
    7             2        1774         101         241
Finish the example
```

3. Program to shows how two GridSim entities interact with each other ; main(ie example3) class creates Gridlets and sends them to the other GridSim entities, i.e. Test class

Aim: To create a java program how two GridSim entities interact with each other.

Description:

An instance of this class simulates a collection of machines. It is up to the GridSim users to define the connectivity among the machines in a collection.

The process of creating a Grid resource is as follows: first create PE objects with a suitable MIPS rating, second assemble them together to create a Machine and finally group multiple Machine objects together to form a resource. A resource having a single machine with one or more PEs is managed as time-shared system using round robin scheduling algorithm.

Source code:

```
package gridsim.example03;
import java.util.*;
import gridsim.*;
class Example3 extends GridSim
{
    private String entityName_;
    private GridletList list_;

    // Gridlet lists received from Test object
    private GridletList receiveList_;
    Example3(String name, double baud_rate, GridletList list) throws Exception
```

```
{
    super(name);
    this.list_ = list;
    receiveList_ = new GridletList();

    // creates a Test entity, and refer it as "entityName"
    entityName_ = "Test";
    new Test(entityName_, baud_rate);
}


public void body()
{
    int size = list_.size();
    Gridlet obj, gridlet;

    // a loop to get one Gridlet at one time and sends it to other GridSim
    // entity
    for (int i = 0; i < size; i++)
    {
        obj = (Gridlet) list_.get(i);
        System.out.println("Inside Example3.body() => Sending Gridlet " +
                obj.getGridletID());


        super.send(entityName_, GridSimTags.SCHEDULE_NOW,
                   GridSimTags.GRIDLET_SUBMIT, obj);

        // Receiving a Gridlet back
        gridlet = super.gridletReceive();

        System.out.println("Inside Example3.body() => Receiving Gridlet "+
                gridlet.getGridletID());

        // stores the received Gridlet into a new GridletList object
        receiveList_.add(gridlet);
    }

    // Signals the end of simulation to "entityName"
    super.send(entityName_, GridSimTags.SCHEDULE_NOW,
               GridSimTags.END_OF_SIMULATION);
}

public GridletList getGridletList() {
    return receiveList_;
}

public static void main(String[] args)
{
    System.out.println("Starting Example3");
    System.out.println();

    try
    {

        int num_user = 0;   // number of users need to be created
        Calendar calendar = Calendar.getInstance();
        boolean trace_flag = true;  // mean trace GridSim events
```

```
            String[] exclude_from_file = { "" };
            String[] exclude_from_processing = { "" };


            String report_name = null;

            // Initialize the GridSim package
            System.out.println("Initializing GridSim package");
            GridSim.init(num_user, calendar, trace_flag, exclude_from_file,
                    exclude_from_processing, report_name);


            // Second step: Creates a list of Gridlets
            GridletList list = createGridlet();
            System.out.println("Creating " + list.size() + " Gridlets");


            // Third step: Creates the Example3 object
            Example3 obj = new Example3("Example3", 560.00, list);


            // Fourth step: Starts the simulation
            GridSim.startGridSimulation();

            // Final step: Prints the Gridlets when simulation is over
            GridletList newList = obj.getGridletList();
            printGridletList(newList);

            System.out.println("Finish Example3");
        }
        catch (Exception e)
        {
            e.printStackTrace();
            System.out.println("Unwanted errors happen");
        }
    }



    private static GridletList createGridlet()
    {
        // Creates a container to store Gridlets
        GridletList list = new GridletList();

        int id = 0;
        double length = 3500.0;
        long file_size = 300;
        long output_size = 300;
        Gridlet gridlet1 = new Gridlet(id, length, file_size, output_size);
        id++;
        Gridlet gridlet2 = new Gridlet(id, 5000, 500, 500);
        id++;
        Gridlet gridlet3 = new Gridlet(id, 9000, 900, 900);

        // Store the Gridlets into a list
        list.add(gridlet1);
```

```
        list.add(gridlet2);
        list.add(gridlet3);


        long seed = 11L*13*17*19*23+1;
        Random random = new Random(seed);


        GridSimStandardPE.setRating(100);

        // creates 5 Gridlets
        int count = 5;
        for (int i = 1; i < count+1; i++)
        {

            length = GridSimStandardPE.toMIs(random.nextDouble()*50);


            file_size = (long) GridSimRandom.real(100, 0.10, 0.40,
                                random.nextDouble());


            output_size = (long) GridSimRandom.real(250, 0.10, 0.50,
                                random.nextDouble());

            // creates a new Gridlet object
            Gridlet gridlet = new Gridlet(id + i, length, file_size,
                                output_size);

            // add the Gridlet into a list
            list.add(gridlet);
        }

        return list;
    }


    private static void printGridletList(GridletList list)
    {
        int size = list.size();
        Gridlet gridlet;

        String indent = "    ";
        System.out.println();
        System.out.println("========== OUTPUT ==========");
        System.out.println("Gridlet ID" + indent + "STATUS");

        for (int i = 0; i < size; i++)
        {
            gridlet = (Gridlet) list.get(i);
            System.out.print(indent + gridlet.getGridletID() + indent
                    + indent);

            if (gridlet.getGridletStatus() == Gridlet.SUCCESS)
                System.out.println("SUCCESS");
        }
    }
}
```

Test.java:

```
package gridsim.example03;
import java.util.*;
import gridsim.*;
import eduni.simjava.Sim_event;

class Test extends GridSim
{

    Test(String name, double baud_rate) throws Exception
    {

        super(name, baud_rate);
        System.out.println("... Creating a new Test object");
    }


    public void body()
    {
        int entityID;
        Sim_event ev = new Sim_event();
        Gridlet gridlet;

        // Gets one event at a time
        for ( sim_get_next(ev); ev.get_tag() != GridSimTags.END_OF_SIMULATION;
                sim_get_next(ev) )
        {
            // Gets the Gridlet object sent by Example3 class
            gridlet = (Gridlet) ev.get_data();

            // Change the Gridlet status, meaning that the Gridlet has been
            // received successfully
            try {
                gridlet.setGridletStatus(Gridlet.SUCCESS);
            }
            catch (Exception e) {
                e.printStackTrace();
            }

            System.out.println("... Inside Test.body() => Receiving Gridlet "+
                    gridlet.getGridletID() + " from Example3 object");

            // get the sender ID, i.e Example3 class
            entityID = ev.get_src();

            // sends back the modified Gridlet to the sender
            super.send(entityID, GridSimTags.SCHEDULE_NOW,
                       GridSimTags.GRIDLET_RETURN, gridlet);
        }

        // when simulation ends, terminate the Input and Output entities
        super.terminateIOEntities();
    }
}
```

Output:

```
Starting Example3

Initializing GridSim package
Initialising...
Creating 8 Gridlets
... Creating a new Test object
Starting GridSim version 5.0
Entities started.
Inside Example3.body() => Sending Gridlet 0
GridInformationService: Notify all GridSim entities for shutting down.
... Inside Test.body() => Receiving Gridlet 0 from Example3 object
Inside Example3.body() => Receiving Gridlet 0
Inside Example3.body() => Sending Gridlet 1
... Inside Test.body() => Receiving Gridlet 1 from Example3 object
Inside Example3.body() => Receiving Gridlet 1
Inside Example3.body() => Sending Gridlet 2
... Inside Test.body() => Receiving Gridlet 2 from Example3 object
Inside Example3.body() => Receiving Gridlet 2
Inside Example3.body() => Sending Gridlet 3
... Inside Test.body() => Receiving Gridlet 3 from Example3 object
Inside Example3.body() => Receiving Gridlet 3
Inside Example3.body() => Sending Gridlet 4
... Inside Test.body() => Receiving Gridlet 4 from Example3 object
Inside Example3.body() => Receiving Gridlet 4
Inside Example3.body() => Sending Gridlet 5
... Inside Test.body() => Receiving Gridlet 5 from Example3 object
Inside Example3.body() => Receiving Gridlet 5
Inside Example3.body() => Sending Gridlet 6
... Inside Test.body() => Receiving Gridlet 6 from Example3 object
Inside Example3.body() => Receiving Gridlet 6
Inside Example3.body() => Sending Gridlet 7
... Inside Test.body() => Receiving Gridlet 7 from Example3 object
Inside Example3.body() => Receiving Gridlet 7
Sim_system: No more future events
Gathering simulation data.
Simulation completed.
```



```
========== OUTPUT ==========
Gridlet ID    STATUS
    0        SUCCESS
    1        SUCCESS
    2        SUCCESS
    3        SUCCESS
    4        SUCCESS
    5        SUCCESS
    6        SUCCESS
    7        SUCCESS
Finish Example3
```

4. Program shows how a grid user submits its Gridlets or tasks to one gridresource entity.

Aim: To create a java program how a grid user submits its Gridlets or tasks to one gridresource entity.

Description:

An instance of this class represents one Grid-enabled task. Individual users create tasks that are processed by Grid resources through resource brokers that implement various scheduling policies.
An instance of this class represents an experiment that consists of a GridletList and an optimization policy. On receiving an experiment form its user, a resource broker starts scheduling individual Gridlets according to the optimization policy set for the experiment.

Source code:

```
package gridsim.example01;
import java.util.*;
import gridsim.*;

// creates Gridlets and sends them to a grid resource entity

class Example4 extends GridSim
{
   private Integer ID_;
   private String name_;
   private GridletList list_;
   private GridletList receiveList_;
   Example4(String name, double baud_rate) throws Exception
   {
      super(name, baud_rate);
      this.name_ = name;
      this.receiveList_ = new GridletList();

      // Gets an ID for this entity
      this.ID_ = new Integer( getEntityId(name) );
      System.out.println("Creating a grid user entity with name = " +
            name + ", and id = " + this.ID_);

      // Creates a list of Gridlets or Tasks for this grid user
      this.list_ = createGridlet( this.ID_.intValue() );
      System.out.println("Creating " + this.list_.size() + " Gridlets");
   }
   public void body()
   {
      int resourceID = 0;
```

```
String resourceName;
LinkedList resList;
ResourceCharacteristics resChar;


while (true)
{

   super.gridSimHold(1.0);    // hold by 1 second
   resList = super.getGridResourceList();
   if (resList.size() > 0)
   {

      Integer num = (Integer) resList.get(0);
      resourceID = num.intValue();

      // Requests to resource entity to send its characteristics
      super.send(resourceID, GridSimTags.SCHEDULE_NOW,
            GridSimTags.RESOURCE_CHARACTERISTICS, this.ID_);

      // waiting to get a resource characteristics
      resChar = (ResourceCharacteristics) super.receiveEventObject();
      resourceName = resChar.getResourceName();

      System.out.println("Received ResourceCharacteristics from " +
          resourceName + ", with id = " + resourceID);

      // record this event into "stat.txt" file
      super.recordStatistics("\"Received ResourceCharacteristics " +
          "from " + resourceName + "\"", "");

      break;
   }
   else
      System.out.println("Waiting to get list of resources ...");
}

Gridlet gridlet;
String info;


for (int i = 0; i < this.list_.size(); i++)
{
   gridlet = (Gridlet) this.list_.get(i);
```

```
            info = "Gridlet_" + gridlet.getGridletID();

            System.out.println("Sending " + info + " to " + resourceName +
                    " with id = " + resourceID);

            // Sends one Gridlet to a grid resource specified in "resourceID"
            super.gridletSubmit(gridlet, resourceID);


            super.recordStatistics("\"Submit " + info + " to " + resourceName +
                    "\"", "");

            // waiting to receive a Gridlet back from resource entity
            gridlet = super.gridletReceive();
            System.out.println("Receiving Gridlet " + gridlet.getGridletID());

            // Recods this event into "GridSim_stat.txt" file for statistical
            // purposes
            super.recordStatistics("\"Received " + info +  " from " +
                    resourceName + "\"", gridlet.getProcessingCost());

            // stores the received Gridlet into a new GridletList object
            this.receiveList_.add(gridlet);
        }

        // shut down all the entities, including GridStatistics entity since
        // we used it to record certain events.
        super.shutdownGridStatisticsEntity();
        super.shutdownUserEntity();
        super.terminateIOEntities();
    }


    public GridletList getGridletList() {
        return this.receiveList_;
    }


    private GridletList createGridlet(int userID)
    {
        // Creates a container to store Gridlets
        GridletList list = new GridletList();

        int id = 0;
```

```
double length = 3500.0;
long file_size = 300;
long output_size = 300;
Gridlet gridlet1 = new Gridlet(id, length, file_size, output_size);
id++;
Gridlet gridlet2 = new Gridlet(id, 5000, 500, 500);
id++;
Gridlet gridlet3 = new Gridlet(id, 9000, 900, 900);

// setting the owner of these Gridlets
gridlet1.setUserID(userID);
gridlet2.setUserID(userID);
gridlet3.setUserID(userID);

// Store the Gridlets into a list
list.add(gridlet1);
list.add(gridlet2);
list.add(gridlet3);

// We create 5 Gridlets with the help of GridSimRandom and
// GriSimStandardPE class
long seed = 11L*13*17*19*23+1;
Random random = new Random(seed);

// sets the PE MIPS Rating
GridSimStandardPE.setRating(100);

// creates 5 Gridlets
int count = 5;
for (int i = 1; i < count+1; i++)
{
   // the Gridlet length determines from random values and the
   // current MIPS Rating for a PE
   length = GridSimStandardPE.toMIs(random.nextDouble()*50);

   // determines the Gridlet file size that varies within the range
   // 100 + (10% to 40%)
   file_size = (long) GridSimRandom.real(100, 0.10, 0.40,
                     random.nextDouble());

   // determines the Gridlet output size that varies within the range
   // 250 + (10% to 50%)
   output_size = (long) GridSimRandom.real(250, 0.10, 0.50,
                     random.nextDouble());
```

```
        // creates a new Gridlet object
        Gridlet gridlet = new Gridlet(id + i, length, file_size,
                        output_size);

        gridlet.setUserID(userID);

        // add the Gridlet into a list
        list.add(gridlet);
    }

    return list;
}


////////////////////////// STATIC METHODS //////////////////////

/**
 * Creates main() to run this example
 */
public static void main(String[] args)
{
    System.out.println("Starting Example4");

    try
    {

        int num_user = 1;   // number of grid users
        Calendar calendar = Calendar.getInstance();
        boolean trace_flag = true;  // mean trace GridSim events


        String[] exclude_from_file = { "" };
        String[] exclude_from_processing = { "" };


        String report_name = null;


        System.out.println("Initializing GridSim package");
        GridSim.init(num_user, calendar, trace_flag, exclude_from_file,
              exclude_from_processing, report_name);
```

```
        String name = "Resource_0";
        GridResource resource = createGridResource(name);


        Example4 obj = new Example4("Example4", 560.00);


        GridSim.startGridSimulation();

        // Final step: Prints the Gridlets when simulation is over
        GridletList newList = obj.getGridletList();
        printGridletList(newList);

        System.out.println("Finish Example4");
    }
    catch (Exception e)
    {
        e.printStackTrace();
        System.out.println("Unwanted errors happen");
    }
}


private static GridResource createGridResource(String name)
{
    System.out.println();
    System.out.println("Starting to create one Grid resource with " +
            "3 Machines");


    MachineList mList = new MachineList();
    System.out.println("Creates a Machine list");


    int mipsRating = 377;
    mList.add( new Machine(0, 4, mipsRating));   // First Machine
    System.out.println("Creates the 1st Machine that has 4 PEs and " +
            "stores it into the Machine list");


    mList.add( new Machine(1, 4, mipsRating));   // Second Machine
    System.out.println("Creates the 2nd Machine that has 4 PEs and " +
            "stores it into the Machine list");
```

```
mList.add( new Machine(2, 2, mipsRating));   // Third Machine
System.out.println("Creates the 3rd Machine that has 2 PEs and " +
      "stores it into the Machine list");


String arch = "Sun Ultra";      // system architecture
String os = "Solaris";          // operating system
double time_zone = 9.0;         // time zone this resource located
double cost = 3.0;              // the cost of using this resource

ResourceCharacteristics resConfig = new ResourceCharacteristics(
      arch, os, mList, ResourceCharacteristics.TIME_SHARED,
      time_zone, cost);

System.out.println("Creates the properties of a Grid resource and " +
      "stores the Machine list");

// 5. Finally, we need to create a GridResource object.
double baud_rate = 100.0;           // communication speed
long seed = 11L*13*17*19*23+1;
double peakLoad = 0.0;        // the resource load during peak hour
double offPeakLoad = 0.0;     // the resource load during off-peak hr
double holidayLoad = 0.0;     // the resource load during holiday

// incorporates weekends so the grid resource is on 7 days a week
LinkedList Weekends = new LinkedList();
Weekends.add(new Integer(Calendar.SATURDAY));
Weekends.add(new Integer(Calendar.SUNDAY));

// incorporates holidays. However, no holidays are set in this example
LinkedList Holidays = new LinkedList();
GridResource gridRes = null;
try
{
   gridRes = new GridResource(name, baud_rate, seed,
      resConfig, peakLoad, offPeakLoad, holidayLoad, Weekends,
      Holidays);
}
catch (Exception e) {
   e.printStackTrace();
}

System.out.println("Finally, creates one Grid resource and stores " +
      "the properties of a Grid resource");
```

```
        System.out.println();

        return gridRes;
    }

        private static void printGridletList(GridletList list)
    {
        int size = list.size();
        Gridlet gridlet;

        String indent = "    ";
        System.out.println();
        System.out.println("========== OUTPUT ==========");
        System.out.println("Gridlet ID" + indent + "STATUS" + indent +
              "Resource ID" + indent + "Cost");

        for (int i = 0; i < size; i++)
        {
           gridlet = (Gridlet) list.get(i);
           System.out.print(indent + gridlet.getGridletID() + indent
                 + indent);

           if (gridlet.getGridletStatus() == Gridlet.SUCCESS)
              System.out.print("SUCCESS");

           System.out.println( indent + indent + gridlet.getResourceID() +
                 indent + indent + gridlet.getProcessingCost() );
        }
    }

}
```

Output:

```
Starting GridSim version 5.0
Entities started.
Received ResourceCharacteristics from Resource_0, with id = 5
Sending Gridlet_0 to Resource_0 with id = 5
Receiving Gridlet 0
Sending Gridlet_1 to Resource_0 with id = 5
Receiving Gridlet 1
Sending Gridlet_2 to Resource_0 with id = 5
Receiving Gridlet 2
Sending Gridlet_3 to Resource_0 with id = 5
Receiving Gridlet 3
Sending Gridlet_4 to Resource_0 with id = 5
Receiving Gridlet 4
Sending Gridlet_5 to Resource_0 with id = 5
Receiving Gridlet 5
Sending Gridlet_6 to Resource_0 with id = 5
Receiving Gridlet 6
Sending Gridlet_7 to Resource_0 with id = 5
Receiving Gridlet 7
GridInformationService: Notify all GridSim entities for shutting down.
Sim_system: No more future events
Gathering simulation data.
Simulation completed.

========== OUTPUT ==========
Gridlet ID    STATUS    Resource ID    Cost
    0         SUCCESS        5        27.851458885941646
    1         SUCCESS        5        39.78779840848807
    2         SUCCESS        5        71.6180371352786
    3         SUCCESS        5        12.260185925286578
    4         SUCCESS        5        32.160803970344375
    5         SUCCESS        5        26.989166622345806
    6         SUCCESS        5        16.721823905496365
    7         SUCCESS        5        20.974565689486667
Finish Example4
```


5. Program to show how a grid user submits its Grid lets or task to many grid resource entities.

Aim: To create a java program how a grid user submits its Grid lets or task to many grid resource entities.

Description:

An instance of this class simulates a machine with one or more CPUs. A Machine object can model both uniprocessor and multiprocessor. class gridsim.MachineList – An instance of this class simulates a collection of machines. It is up to the GridSim users to define the connectivity among the machines in a collection.

An instance of this class represents all static properties of a Resource such as resource architecture, OS, management policy (time or space shared), cost, and time zone at which resource is located.

Source code:

```
package gridsim.example01;

import java.util.*;
import gridsim.*;
```

```
//class creates Gridlets and sends them to many grid resource

class Example5 extends GridSim
{
   private Integer ID_;
   private String name_;
   private GridletList list_;
   private GridletList receiveList_;
   private int totalResource_;



   Example5(String name, double baud_rate, int total_resource)
         throws Exception
   {
      super(name, baud_rate);
      this.name_ = name;
      this.totalResource_ = total_resource;
      this.receiveList_ = new GridletList();

      // Gets an ID for this entity
      this.ID_ = new Integer( getEntityId(name) );
      System.out.println("Creating a grid user entity with name = " +
            name + ", and id = " + this.ID_);

      // Creates a list of Gridlets or Tasks for this grid user
      this.list_ = createGridlet( this.ID_.intValue() );
      System.out.println("Creating " + this.list_.size() + " Gridlets");
   }

   /**
    * The core method that handles communications among GridSim entities
    */
   public void body()
   {
      int resourceID[] = new int[this.totalResource_];
      double resourceCost[] = new double[this.totalResource_];
      String resourceName[] = new String[this.totalResource_];

      LinkedList resList;
      ResourceCharacteristics resChar;


      while (true)
```

```
{

   super.gridSimHold(1.0);    // hold by 1 second

   resList = super.getGridResourceList();
   if (resList.size() == this.totalResource_)
      break;
   else
      System.out.println("Waiting to get list of resources ...");
}

int i = 0;


for (i = 0; i < this.totalResource_; i++)
{

   resourceID[i] = ( (Integer)resList.get(i) ).intValue();


   super.send(resourceID[i], GridSimTags.SCHEDULE_NOW,
           GridSimTags.RESOURCE_CHARACTERISTICS, this.ID_);


   resChar = (ResourceCharacteristics) super.receiveEventObject();
   resourceName[i] = resChar.getResourceName();
   resourceCost[i] = resChar.getCostPerSec();

   System.out.println("Received ResourceCharacteristics from " +
         resourceName[i] + ", with id = " + resourceID[i]);

   // record this event into "stat.txt" file
   super.recordStatistics("\"Received ResourceCharacteristics " +
         "from " + resourceName[i] + "\"", "");
}

Gridlet gridlet;
String info;

// a loop to get one Gridlet at one time and sends it to a random grid
  Random random = new Random();
int id = 0;
for (i = 0; i < this.list_.size(); i++)
{
```

```
        gridlet = (Gridlet) this.list_.get(i);
        info = "Gridlet_" + gridlet.getGridletID();

        id = random.nextInt(this.totalResource_);
        System.out.println("Sending " + info + " to " + resourceName[id] +
                " with id = " + resourceID[id]);

        // Sends one Gridlet to a grid resource specified in "resourceID"
        super.gridletSubmit(gridlet, resourceID[id]);
       super.recordStatistics("\"Submit " + info + " to " +
                resourceName[id] + "\"", "");


        gridlet = super.gridletReceive();
        System.out.println("Receiving Gridlet " + gridlet.getGridletID());


        super.recordStatistics("\"Received " + info +  " from " +
                resourceName[id] + "\"", gridlet.getProcessingCost());

        this.receiveList_.add(gridlet);
    }


    super.shutdownGridStatisticsEntity();
    super.shutdownUserEntity();
    super.terminateIOEntities();
}


public GridletList getGridletList() {
    return this.receiveList_;
}


private GridletList createGridlet(int userID)
{

    GridletList list = new GridletList();

   int id = 0;
    double length = 3500.0;
    long file_size = 300;
    long output_size = 300;
```

```
Gridlet gridlet1 = new Gridlet(id, length, file_size, output_size);
id++;
Gridlet gridlet2 = new Gridlet(id, 5000, 500, 500);
id++;
Gridlet gridlet3 = new Gridlet(id, 9000, 900, 900);

// setting the owner of these Gridlets
gridlet1.setUserID(userID);
gridlet2.setUserID(userID);
gridlet3.setUserID(userID);

// Store the Gridlets into a list
list.add(gridlet1);
list.add(gridlet2);
list.add(gridlet3);


long seed = 11L*13*17*19*23+1;
Random random = new Random(seed);

// sets the PE MIPS Rating
GridSimStandardPE.setRating(100);

// creates 5 Gridlets
int count = 5;
for (int i = 1; i < count+1; i++)
{

   length = GridSimStandardPE.toMIs(random.nextDouble()*50);


   file_size = (long) GridSimRandom.real(100, 0.10, 0.40,
                     random.nextDouble());

   output_size = (long) GridSimRandom.real(250, 0.10, 0.50,
                     random.nextDouble());

   // creates a new Gridlet object
   Gridlet gridlet = new Gridlet(id + i, length, file_size,
                     output_size);

   gridlet.setUserID(userID);

   // add the Gridlet into a list
```

```
        list.add(gridlet);
    }

    return list;
}


////////////////////////// STATIC METHODS //////////////////////

/**
 * Creates main() to run this example
 */
public static void main(String[] args)
{
    System.out.println("Starting Example5");

    try
    {

        int num_user = 1;   // number of grid users
        Calendar calendar = Calendar.getInstance();
        boolean trace_flag = false;  // mean don't trace GridSim events

        String[] exclude_from_file = { "" };
        String[] exclude_from_processing = { "" };

        String report_name = null;

         System.out.println("Initializing GridSim package");
        GridSim.init(num_user, calendar, trace_flag, exclude_from_file,
              exclude_from_processing, report_name);

        GridResource resource0 = createGridResource("Resource_0");
        GridResource resource1 = createGridResource("Resource_1");
        GridResource resource2 = createGridResource("Resource_2");
        int total_resource = 3;

        // Third step: Creates the Example5 object
        Example5 obj = new Example5("Example5", 560.00, total_resource);

        // Fourth step: Starts the simulation
        GridSim.startGridSimulation();

        // Final step: Prints the Gridlets when simulation is over
```

```
        GridletList newList = obj.getGridletList();
        printGridletList(newList);

        System.out.println("Finish Example5");
    }
    catch (Exception e)
    {
        e.printStackTrace();
        System.out.println("Unwanted errors happen");
    }
}


private static GridResource createGridResource(String name)
{
    System.out.println();
    System.out.println("Starting to create one Grid resource with " +
            "3 Machines");


    MachineList mList = new MachineList();
    System.out.println("Creates a Machine list");



    int mipsRating = 377;
    mList.add( new Machine(0, 4, mipsRating));   // First Machine
    System.out.println("Creates the 1st Machine that has 4 PEs and " +
            "stores it into the Machine list");

      mList.add( new Machine(1, 4, mipsRating));   // Second Machine
    System.out.println("Creates the 2nd Machine that has 4 PEs and " +
            "stores it into the Machine list");

    mList.add( new Machine(2, 2, mipsRating));   // Third Machine
    System.out.println("Creates the 3rd Machine that has 2 PEs and " +
            "stores it into the Machine list");


    String arch = "Sun Ultra";      // system architecture
    String os = "Solaris";          // operating system
    double time_zone = 9.0;         // time zone this resource located
    double cost = 3.0;              // the cost of using this resource
```

```
    ResourceCharacteristics resConfig = new ResourceCharacteristics(
            arch, os, mList, ResourceCharacteristics.TIME_SHARED,
            time_zone, cost);

    System.out.println("Creates the properties of a Grid resource and " +
            "stores the Machine list");

    // 5. Finally, we need to create a GridResource object.
    double baud_rate = 100.0;          // communication speed
    long seed = 11L*13*17*19*23+1;
    double peakLoad = 0.0;      // the resource load during peak hour
    double offPeakLoad = 0.0;    // the resource load during off-peak hr
    double holidayLoad = 0.0;    // the resource load during holiday

    // incorporates weekends so the grid resource is on 7 days a week
    LinkedList Weekends = new LinkedList();
    Weekends.add(new Integer(Calendar.SATURDAY));
    Weekends.add(new Integer(Calendar.SUNDAY));

    // incorporates holidays. However, no holidays are set in this example
    LinkedList Holidays = new LinkedList();
    GridResource gridRes = null;
    try
    {
       gridRes = new GridResource(name, baud_rate, seed,
          resConfig, peakLoad, offPeakLoad, holidayLoad, Weekends,
          Holidays);
    }
    catch (Exception e) {
       e.printStackTrace();
    }

    System.out.println("Finally, creates one Grid resource and stores " +
            "the properties of a Grid resource");
    System.out.println();

    return gridRes;
}

private static void printGridletList(GridletList list)
{
    int size = list.size();
    Gridlet gridlet;
```

```
        String indent = "    ";
        System.out.println();
        System.out.println("========== OUTPUT ==========");
        System.out.println("Gridlet ID" + indent + "STATUS" + indent +
              "Resource ID" + indent + "Cost");

        for (int i = 0; i < size; i++)
        {
           gridlet = (Gridlet) list.get(i);
           System.out.print(indent + gridlet.getGridletID() + indent
                 + indent);

           if (gridlet.getGridletStatus() == Gridlet.SUCCESS)
              System.out.print("SUCCESS");

           System.out.println( indent + indent + gridlet.getResourceID() +
                 indent + indent + gridlet.getProcessingCost() );
        }
     }

} // end class
```

Output:

```
Starting Example5
Initializing GridSim package
Initialising...

Starting to create one Grid resource with 3 Machines
Creates a Machine list
Creates the 1st Machine that has 4 PEs and stores it into the Machine list
Creates the 2nd Machine that has 4 PEs and stores it into the Machine list
Creates the 3rd Machine that has 2 PEs and stores it into the Machine list
Creates the properties of a Grid resource and stores the Machine list
Finally, creates one Grid resource and stores the properties of a Grid resource
```

```
Creating a grid user entity with name = Example5, and id = 17
Creating 8 Gridlets
Starting GridSim version 5.0
Entities started.
Waiting to get list of resources ...
Waiting to get list of resources ...
Received ResourceCharacteristics from Resource_0, with id = 5
Received ResourceCharacteristics from Resource_2, with id = 13
Received ResourceCharacteristics from Resource_1, with id = 9
Sending Gridlet_0 to Resource_1 with id = 9
Receiving Gridlet 0
Sending Gridlet_1 to Resource_1 with id = 9
Receiving Gridlet 1
Sending Gridlet_2 to Resource_1 with id = 9
Receiving Gridlet 2
Sending Gridlet_3 to Resource_0 with id = 5
Receiving Gridlet 3
Sending Gridlet_4 to Resource_1 with id = 9
Receiving Gridlet 4
Sending Gridlet_5 to Resource_0 with id = 5
Receiving Gridlet 5
Sending Gridlet_6 to Resource_2 with id = 13
Receiving Gridlet 6
Sending Gridlet_7 to Resource_2 with id = 13
Receiving Gridlet 7
GridInformationService: Notify all GridSim entities for shutting down.
Sim_system: No more future events
Gathering simulation data.
Simulation completed.

========== OUTPUT ==========
Gridlet ID     STATUS     Resource ID     Cost
    0          SUCCESS          9          27.851458885941646
    1          SUCCESS          9          39.78779840848807
    2          SUCCESS          9          71.6180371352786
    3          SUCCESS          5          12.260185925286578
```


6. Program to show how to create one or more grid users and submits its Gridlets or task to many grid resource entities.

Aim: To create a java program one or more grid users and submits its Gridlets or task to many grid resource entities.

Description:

GridSim supports entities for simulation of single processor and multiprocessor, heterogeneous resources that can be configured as time or space shared systems. It allows setting their clock to different time zones to simulate geographic distribution of resources. It supports entities that simulate networks used for communication among resources. It extends the GridSim class and gains communication and concurrent entity capability

Source code:

```
package gridsim.example01;



import java.util.*;
import gridsim.*;

class Example6 extends GridSim
{
   private Integer ID_;
   private String name_;
   private GridletList list_;
   private GridletList receiveList_;
   private int totalResource_;
```

```
Example6(String name, double baud_rate, int total_resource)
      throws Exception
{
   super(name, baud_rate);
   this.name_ = name;
   this.totalResource_ = total_resource;
   this.receiveList_ = new GridletList();

   // Gets an ID for this entity
   this.ID_ = new Integer( getEntityId(name) );
   System.out.println("Creating a grid user entity with name = " +
         name + ", and id = " + this.ID_);

   // Creates a list of Gridlets or Tasks for this grid user
   this.list_ = createGridlet( this.ID_.intValue() );
   System.out.println(name + ":Creating "+ this.list_.size() +
         " Gridlets");
}


public void body()
{
   int resourceID[] = new int[this.totalResource_];
   double resourceCost[] = new double[this.totalResource_];
   String resourceName[] = new String[this.totalResource_];

   LinkedList resList;
   ResourceCharacteristics resChar;

   // waiting to get list of resources. Since GridSim package uses
   // multi-threaded environment, your request might arrive earlier
   // before one or more grid resource entities manage to register
   // themselves to GridInformationService (GIS) entity.
   // Therefore, it's better to wait in the first place
   while (true)
   {
      // need to pause for a while to wait GridResources finish
      // registering to GIS
      super.gridSimHold(1.0);    // hold by 1 second

      resList = super.getGridResourceList();
      if (resList.size() == this.totalResource_)
         break;
```

```
    else
    {
        System.out.println(this.name_ +
                ":Waiting to get list of resources ...");
    }
}


// a loop to get all the resources available
int i = 0;
for (i = 0; i < this.totalResource_; i++)
{
    // Resource list contains list of resource IDs not grid resource
    // objects.
    resourceID[i] = ( (Integer)resList.get(i) ).intValue();

    // Requests to resource entity to send its characteristics
    super.send(resourceID[i], GridSimTags.SCHEDULE_NOW,
            GridSimTags.RESOURCE_CHARACTERISTICS, this.ID_);

    // waiting to get a resource characteristics
    resChar = (ResourceCharacteristics) super.receiveEventObject();
    resourceName[i] = resChar.getResourceName();
    resourceCost[i] = resChar.getCostPerSec();

    System.out.println(this.name_ +
            ":Received ResourceCharacteristics from " +
            resourceName[i] + ", with id = " + resourceID[i]);

    // record this event into "stat.txt" file
    super.recordStatistics("\"Received ResourceCharacteristics " +
            "from " + resourceName[i] + "\"", "");
}

Gridlet gridlet;
String info;

// a loop to get one Gridlet at one time and sends it to a random grid
// resource entity. Then waits for a reply
int id = 0;
for (i = 0; i < this.list_.size(); i++)
{
    gridlet = (Gridlet) this.list_.get(i);
    info = "Gridlet_" + gridlet.getGridletID();
```

```
            id = GridSimRandom.intSample(this.totalResource_);
            System.out.println(this.name_ + ":Sending " + info + " to " +
                    resourceName[id] + " with id = " + resourceID[id]);

            // Sends one Gridlet to a grid resource specified in "resourceID"
            super.gridletSubmit(gridlet, resourceID[id]);

            // Recods this event into "stat.txt" file for statistical purposes
            super.recordStatistics("\"Submit " + info + " to " +
                    resourceName[id] + "\"", "");

            // waiting to receive a Gridlet back from resource entity
            gridlet = super.gridletReceive();
            System.out.println(this.name_ + ":Receiving Gridlet " +
                    gridlet.getGridletID() );

            // Recods this event into "stat.txt" file for statistical purposes
            super.recordStatistics("\"Received " + info +  " from " +
                    resourceName[id] + "\"", gridlet.getProcessingCost());

            // stores the received Gridlet into a new GridletList object
            this.receiveList_.add(gridlet);
        }

        // shut down all the entities, including GridStatistics entity since
        // we used it to record certain events.
        super.shutdownGridStatisticsEntity();
        super.shutdownUserEntity();
        super.terminateIOEntities();
        System.out.println(this.name_ + ":%%%% Exiting body()");
    }


    public GridletList getGridletList() {
        return this.receiveList_;
    }


    private GridletList createGridlet(int userID)
    {
        // Creates a container to store Gridlets
        GridletList list = new GridletList();
```

```
int id = 0;
double length = 3500.0;
long file_size = 300;
long output_size = 300;
Gridlet gridlet1 = new Gridlet(id, length, file_size, output_size);
id++;
Gridlet gridlet2 = new Gridlet(id, 5000, 500, 500);
id++;
Gridlet gridlet3 = new Gridlet(id, 9000, 900, 900);

// setting the owner of these Gridlets
gridlet1.setUserID(userID);
gridlet2.setUserID(userID);
gridlet3.setUserID(userID);

// Store the Gridlets into a list
list.add(gridlet1);
list.add(gridlet2);
list.add(gridlet3);


GridSimStandardPE.setRating(100);

// creates 5 Gridlets
int max = 5;
int count = GridSimRandom.intSample(max);
for (int i = 1; i < count+1; i++)
{

   length = GridSimStandardPE.toMIs(GridSimRandom.doubleSample()*50);


   file_size = (long) GridSimRandom.real(100, 0.10, 0.40,
                                          GridSimRandom.doubleSample());

   // determines the Gridlet output size that varies within the range
   // 250 + (10% to 50%)
   output_size = (long) GridSimRandom.real(250, 0.10, 0.50,
            GridSimRandom.doubleSample());

   // creates a new Gridlet object
   Gridlet gridlet = new Gridlet(id + i, length, file_size,
                  output_size);
```

```
            gridlet.setUserID(userID);

            // add the Gridlet into a list
            list.add(gridlet);
        }

        return list;
    }


    ////////////////////// STATIC METHODS ///////////////////////////////

    /**
     * Creates main() to run this example
     */
    public static void main(String[] args)
    {
        System.out.println("Starting Example6");

        try
        {

            int num_user = 3;   // number of grid users
            Calendar calendar = Calendar.getInstance();
            boolean trace_flag = false;  // mean don't trace GridSim events


            String[] exclude_from_file = { "" };
            String[] exclude_from_processing = { "" };


            String report_name = null;


            GridSim.init(num_user, calendar, trace_flag, exclude_from_file,
                    exclude_from_processing, report_name);

            // Second step: Creates one or more GridResource objects
            GridResource resource0 = createGridResource("Resource_0");
            GridResource resource1 = createGridResource("Resource_1");
            GridResource resource2 = createGridResource("Resource_2");
            int total_resource = 3;
```

```
        // Third step: Creates grid users
        Example6 user0 = new Example6("User_0", 560.00, total_resource);
        Example6 user1 = new Example6("User_1", 250.00, total_resource);
        Example6 user2 = new Example6("User_2", 150.00, total_resource);

        // Fourth step: Starts the simulation
        GridSim.startGridSimulation();

        // Final step: Prints the Gridlets when simulation is over
        GridletList newList = null;
        newList = user0.getGridletList();
        printGridletList(newList, "User_0");

        newList = user1.getGridletList();
        printGridletList(newList, "User_1");

        newList = user2.getGridletList();
        printGridletList(newList, "User_2");

        System.out.println("Finish Example6");
      }
      catch (Exception e)
      {
        e.printStackTrace();
        System.out.println("Unwanted errors happen");
      }
    }


    private static GridResource createGridResource(String name)
    {

      MachineList mList = new MachineList();


      int mipsRating = 377;
      mList.add( new Machine(0, 4, mipsRating));   // First Machine


      mList.add( new Machine(1, 4, mipsRating));   // Second Machine
      mList.add( new Machine(2, 2, mipsRating));   // Third Machine


      String arch = "Sun Ultra";      // system architecture
```

```
    String os = "Solaris";          // operating system
    double time_zone = 9.0;         // time zone this resource located
    double cost = 3.0;              // the cost of using this resource

    ResourceCharacteristics resConfig = new ResourceCharacteristics(
          arch, os, mList, ResourceCharacteristics.TIME_SHARED,
          time_zone, cost);

    // 5. Finally, we need to create a GridResource object.
    double baud_rate = 100.0;           // communication speed
    long seed = 11L*13*17*19*23+1;
    double peakLoad = 0.0;       // the resource load during peak hour
    double offPeakLoad = 0.0;    // the resource load during off-peak hr
    double holidayLoad = 0.0;    // the resource load during holiday

    // incorporates weekends so the grid resource is on 7 days a week
    LinkedList Weekends = new LinkedList();
    Weekends.add(new Integer(Calendar.SATURDAY));
    Weekends.add(new Integer(Calendar.SUNDAY));

    // incorporates holidays. However, no holidays are set in this example
    LinkedList Holidays = new LinkedList();
    GridResource gridRes = null;
    try {
       gridRes = new GridResource(name, baud_rate, seed,
          resConfig, peakLoad, offPeakLoad, holidayLoad, Weekends,
          Holidays);
    }
    catch (Exception e) {
       e.printStackTrace();
    }

    System.out.println("Creates one Grid resource with name = " + name);
    return gridRes;
}


private static void printGridletList(GridletList list, String name)
{
    int size = list.size();
    Gridlet gridlet;

    String indent = "    ";
    System.out.println();
```

```
        System.out.println("========== OUTPUT for " + name + " ==========");
        System.out.println("Gridlet ID" + indent + "STATUS" + indent +
            "Resource ID" + indent + "Cost");

        for (int i = 0; i < size; i++)
        {
          gridlet = (Gridlet) list.get(i);
          System.out.print(indent + gridlet.getGridletID() + indent
              + indent);

          if (gridlet.getGridletStatus() == Gridlet.SUCCESS)
            System.out.print("SUCCESS");

          System.out.println( indent + indent + gridlet.getResourceID() +
              indent + indent + gridlet.getProcessingCost() );
        }
      }

} // end class
```

Output:

```
Starting Example6
Initialising...
Creates one Grid resource with name = Resource_0
Creates one Grid resource with name = Resource_1
Creates one Grid resource with name = Resource_2
Creating a grid user entity with name = User_0, and id = 17
User_0:Creating 4 Gridlets
Creating a grid user entity with name = User_1, and id = 20
User_1:Creating 6 Gridlets
Creating a grid user entity with name = User_2, and id = 23
User_2:Creating 4 Gridlets
Starting GridSim version 5.0
Entities started.
User_0:Waiting to get list of resources ...
User_1:Waiting to get list of resources ...
User_2:Waiting to get list of resources ...
User_0:Waiting to get list of resources ...
User_1:Waiting to get list of resources ...
User_2:Waiting to get list of resources ...
User_2:Received ResourceCharacteristics from Resource_1, with id = 9
User_1:Received ResourceCharacteristics from Resource_1, with id = 9
User_2:Received ResourceCharacteristics from Resource_0, with id = 5
User_1:Received ResourceCharacteristics from Resource_0, with id = 5
User_2:Received ResourceCharacteristics from Resource_2, with id = 13
User_0:Received ResourceCharacteristics from Resource_1, with id = 9
User_2:Sending Gridlet_0 to Resource_2 with id = 13
```

```
User_1:Receiving Gridlet 4
User_1:Sending Gridlet_5 to Resource_1 with id = 9
User_1:Receiving Gridlet 5
User_1:%%%% Exiting body()
GridInformationService: Notify all GridSim entities for shutting down.
Sim_system: No more future events
Gathering simulation data.
Simulation completed.

========== OUTPUT for User_0 ==========
Gridlet ID    STATUS    Resource ID    Cost
    0        SUCCESS        9        27.851458885941668
    1        SUCCESS        5        39.78779840848807
    2        SUCCESS        13        71.6180371352786
    3        SUCCESS        5        25.406943940810606

========== OUTPUT for User_1 ==========
Gridlet ID    STATUS    Resource ID    Cost
    0        SUCCESS        13        27.851458885941668
    1        SUCCESS        5        39.78779840848807
    2        SUCCESS        9        71.6180371352786
    3        SUCCESS        9        3.008155830880753
    4        SUCCESS        5        11.493787340610936
    5        SUCCESS        9        12.037352690631678

========== OUTPUT for User_2 ==========
Gridlet ID    STATUS    Resource ID    Cost
    0        SUCCESS        13        27.851458885941646
    1        SUCCESS        13        39.78779840848807
    2        SUCCESS        13        71.6180371352786
    3        SUCCESS        13        31.412221561590513
Finish Example6
```


7. Program to creates one Grid resource with three machines.

Aim: To create a java program one Grid resource with three machines.

Description:

Each instance of the Resource entity represents a grid resource. During simulation, GridSim creates a number of multi-threaded entities, each of which runs in parallel in its own thread. An entity's behavior needs to be simulated within its body() method, as dictated by SimJava. It extends the GridSim class and gains communication and concurrent entity capability

Source code:

```
package gridsim.example01;

      import java.util.Calendar;
      import java.util.LinkedList;

      import gridsim.*;


      class Example1
      {

          @SuppressWarnings("unused")
             public static void main(String[] args)
```

```
{
    System.out.println("Starting example of how to create one Grid " +
            "resource");

    try
    {

        int num_user = 0;
        Calendar calendar = Calendar.getInstance();
        boolean trace_flag = true; // mean trace GridSim events/activities

        // list of files or processing names to be excluded from any
        //statistical measures
        String[] exclude_from_file = { "" };
        String[] exclude_from_processing = { "" };


        String report_name = null;

        // Initialize the GridSim package
        System.out.println("Initializing GridSim package");
        GridSim.init(num_user, calendar, trace_flag, exclude_from_file,
                exclude_from_processing, report_name);



        // Second step: Create one Grid resource
        GridResource gridResource = createGridResource();
        System.out.println("Finish the 1st example");


    }
    catch (Exception e)
    {
        e.printStackTrace();
        System.out.println("Unwanted error happens");
    }
}



@SuppressWarnings("deprecation")
   private static GridResource createGridResource()
{
    System.out.println("Starting to create one Grid resource with " +
            "3 Machines ...");


    MachineList mList = new MachineList();
    System.out.println("Creates a Machine list");


    int mipsRating = 377;
    mList.add( new Machine(0, 4, mipsRating));   // First Machine
    System.out.println("Creates the 1st Machine that has 4 PEs and " +
            "stores it into the Machine list");
```

```
        mList.add( new Machine(1, 4, mipsRating));   // Second Machine
        System.out.println("Creates the 2nd Machine that has 4 PEs and " +
                "stores it into the Machine list");

        mList.add( new Machine(2, 2, mipsRating));   // Third Machine
        System.out.println("Creates the 3rd Machine that has 2 PEs and " +
                "stores it into the Machine list");


        String arch = "Sun Ultra";      // system architecture
        String os = "Solaris";          // operating system
        double time_zone = 9.0;         // time zone this resource located
        double cost = 3.0;              // the cost of using this resource

        ResourceCharacteristics resConfig = new ResourceCharacteristics(
                arch, os, mList, ResourceCharacteristics.TIME_SHARED,
                time_zone, cost);

        System.out.println();
        System.out.println("Creates the properties of a Grid resource and " +
                "stores the Machine list");

        // 5. Finally, we need to create a GridResource object.
        String name = "Resource_0";         // resource name
        double baud_rate = 100.0;           // communication speed
        long seed = 11L*13*17*19*23+1;
        double peakLoad = 0.0;        // the resource load during peak hour
        double offPeakLoad = 0.0;     // the resource load during off-peak hr
        double holidayLoad = 0.0;     // the resource load during holiday

        // incorporates weekends so the grid resource is on 7 days a week
        LinkedList<Integer> Weekends = new LinkedList<Integer>();
        Weekends.add(new Integer(Calendar.SATURDAY));
        Weekends.add(new Integer(Calendar.SUNDAY));

        // incorporates holidays. However, no holidays are set in this example
        LinkedList<Integer> Holidays = new LinkedList<Integer>();

        GridResource gridRes = null;
        try
        {
            gridRes = new GridResource(name, baud_rate, seed,
                resConfig, peakLoad, offPeakLoad, holidayLoad, Weekends,
                Holidays);
        }
        catch (Exception e) {
            e.printStackTrace();
        }

        System.out.println("Finally, creates one Grid resource and stores " +
                "the properties of a Grid resource");

        return gridRes;
    }
}
```

Output:

```
Starting example of how to create one Grid resource
Initializing GridSim package
Initialising...
Starting to create one Grid resource with 3 Machines ...
Creates a Machine list
Creates the 1st Machine that has 4 PEs and stores it into the Machine list
Creates the 2nd Machine that has 4 PEs and stores it into the Machine list
Creates the 3rd Machine that has 2 PEs and stores it into the Machine list
```