Ex. No: 4

Date:

Use the GAE launcher to launch the web applications.

AIM

To create hello world app and other simple web applications using python/java in Google App Engine.

PROCEDURE

Google Cloud SDK is a set of tools that you can use to manage resources and applications hosted on Google Cloud Platform. These include the gcloud, gsutil,and bq command line tools. The gcloud command-line tool is downloaded along with the Cloud SDK; a comprehensive guide to the gcloud CLI can be found in gcloud command-line tool overview. Gcloud CLI reference documents all of the gcloud CLI's functionality. For a quicker introduction, refer to the gcloud command-line tool cheat sheet.

Install the latest Cloud SDK version (303.0.0)

Step 1: Download the Cloud SDK installer.

```
(New-Object
Net.WebClient).DownloadFile("https://dl.google.com/dl/cloudsdk/channels/ra
pid/GoogleCloudSDKInstaller.exe", "$env:Temp\GoogleCloudSDKInstaller.exe")

& $env:Temp\GoogleCloudSDKInstaller.exe
```

Open a PowerShell terminal and run the following PowerShell commands.

Step 2: Launch the installer and follow the prompts. The installer is signed by Google LLC.Cloud SDK requires Python. The installer will install all necessary dependencies, including the needed Python version. While Cloud SDK currently uses Python 2 by default, you can use an existing Python installation if necessary by unchecking the option to 'Install Bundled Python'.

Step 3: After installation has completed, accept the following options:Start Cloud SDK

Shell

Run gcloud init

The installer starts a terminal window and runs the gcloud init command.

Step 4: The default installation does not include the App Engine extensions required to deployan application using gcloud commands. These components can be installed using the Cloud SDKcomponent manager.

Optional: Enable accessibility features

For a more streamlined screen reader experience, the gcloud command-line tool comes with an accessibility/screen_reader property.

```
gcloud config set accessibility/screen_reader
true
```

In order to provide a more streamlined screen reader experience, the gcloud command-line tool comes with an *accessibility/screen_reader* property. When this property is set to true, the following behaviour is enabled:

Status trackers instead of unicode spinners: The phrase 'working' will be displayed on stderr while gcloud is performing tasks.

Percentage progress bars: Progress will be displayed as a percentage, outputted to stderr. **Boxed tables drawn with ASCII characters:** Boxed tables are the default output of many listcommands. Instead of being drawn with Unicode, they will be rendered using ascii characters.Also, consider using the --*format* flag to define your own format.

To enable these accessibility features, run:

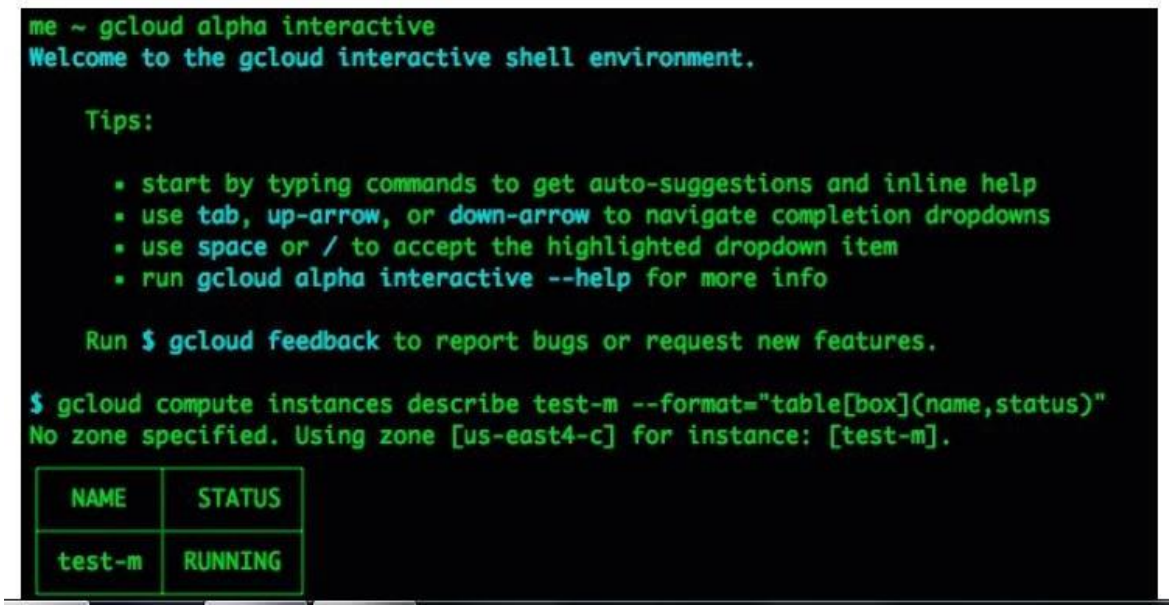
```
$ gcloud config set accessibility/screen_reader
true
```

PROGRAM:

Write down simple hello world python program.

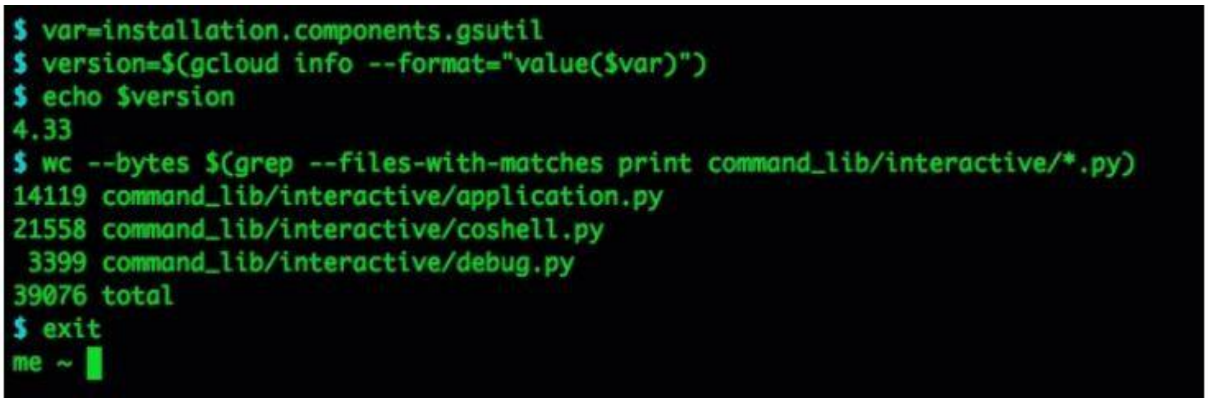
```
import webapp2
        class MainPage(webapp2.RequestHandler)def get(self):
            self.response.write("Hello World")

        app = webapp2.WSGIApplications([('/',MainPage)], debug=True)

        Create app.yaml which has url linkruntime: python3 api_version: 1
        threadsafe: true

        handlers:
```

```
-url: /
 script: test.app
```



```
me ~ gcloud alpha interactive
Welcome to the gcloud interactive shell environment.

   Tips:

     • start by typing commands to get auto-suggestions and inline help
     • use tab, up-arrow, or down-arrow to navigate completion dropdowns
     • use space or / to accept the highlighted dropdown item
     • run gcloud alpha interactive --help for more info

   Run $ gcloud feedback to report bugs or request new features.

$ gcloud compute instances describe test-m --format="table[box](name,status)"
No zone specified. Using zone [us-east4-c] for instance: [test-m].
┌────────┬─────────┐
│  NAME  │  STATUS │
├────────┼─────────┤
│ test-m │ RUNNING │
└────────┴─────────┘
```

- Run the gcloud prompt and give Yes option to proceed further.
- Install gcloud components using the command



```
$ var=installation.components.gsutil
$ version=$(gcloud info --format="value($var)")
$ echo $version
4.33
$ wc --bytes $(grep --files-with-matches print command_lib/interactive/*.py)
14119 command_lib/interactive/application.py
21558 command_lib/interactive/coshell.py
 3399 command_lib/interactive/debug.py
39076 total
$ exit
me ~ █
```

```
gcloud components install cloud-datastore-emulator
```

RESULT:

Thus the execution of the simple hello world application using python was done and output was verified successfully.

Ex. No: 5

Date:

Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.

AIM

To simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.

PROCEDURE

Basics of Scheduling

In computers, Scheduling is a process of arranging the submitted jobs/task into a very specific sequence of execution, handled by a very special program known as a scheduler.

Scheduler's main objective is to keep the underlined hardware resources (primarily processor) to be used effectively as well as efficient. In general, the scheduler may prefer to haveany of the following scheduling approaches:

- Space-shared: Requested resources are allocated dedicatedly to requesting workload for execution & will be released only on completion. Space-shared is also known as a batch process scheduling.
- Time-shared: Requested resources would be shared among more than one workload(task). The sharing is done based on time-sliced allocation where each workloadis allocated with a required resource for a defined time(e.g., 200 milliseconds). Once the defined time slice is over, the current workload execution paused, and the resource is released. The released resource gets allocated to the next workload for the same defined time slice, and this cycle goes on till the time all the workloads execution is over. Time- shared is also known as round-robin scheduling.

Scheduling in Cloud

As cloud computing is the virtualized operating environment, and the virtual machines are primary computing component which is responsible for the execution of the workloads. The virtual machine(s) are powered by a physical server host machine (i.e.) hardware. Depending on the requirement of the Virtual Machine(VM) there could be "one to one" or "many to one" mapping between the VM and host machine. That means in cloud computing the scheduling is done at both the mapping levels that are:

- Virtual Machine to Host Machines
- Tasks to Virtual Machines

Both of VM to Host as well as Workload(task) to VM mappings may utilize space-shareor time-shared or any other specialized scheduling algorithm.

Scheduling in Cloudsim

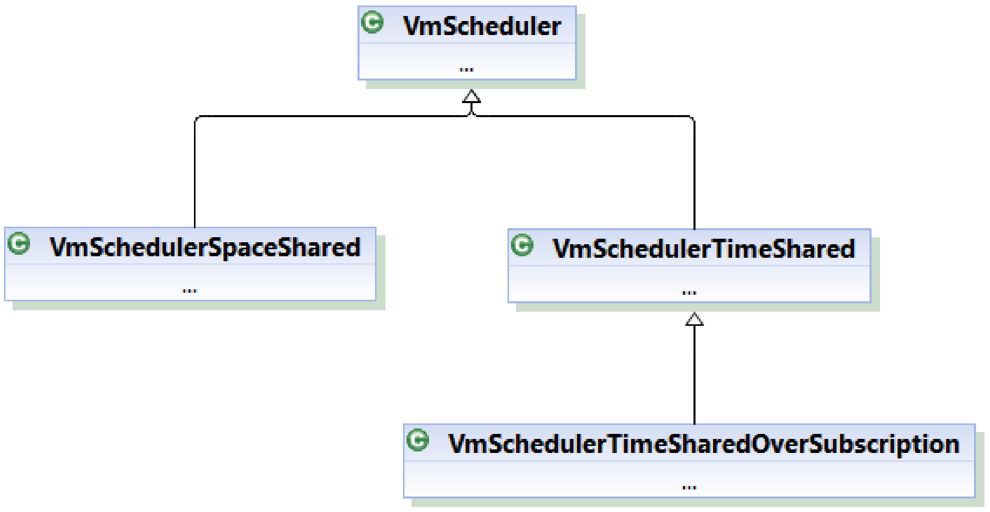
The Cloudsim simulation toolkit framework has effectively addressed the Scheduling scenario and implemented it as a set of the programmable class hierarchies with parent class as:

1. VmScheduler
2. CloudletScheduler

Also, Virtual Machine (VM) and Task (Cloudlet) scheduling are one of the most important and the popular use case to be simulated by researchers using the CloudSim simulation toolkit.

Cloudsim Virtual Machine Scheduling

The VmScheduler is an abstract class that defines and implements the policy used toshare processing power among virtual machines running on a specified host. The hierarchy of thecloudsim virtual machine scheduler classes is as:



These classes can be located in “org.cloudbus.cloudsim” package of cloudsim.

- VmSchedulerTimeShared:
    - This class implements the VM scheduling policy that allocatesone or more processing elements to a single Virtual machine and allows the sharing of processing elements by multiple virtual machines with a specified time slice. This class also considers the overhead of VM allocation switching(similar to context switching) in policy definition. Here, the VM allocation will fail if the number of processing elements requested is not available. for example, if the VM request for quad-core processor whereas the allocated host has an only dual-core the allocation will fail.

- VmSchedulerSpaceShared:
    - This class implements the VM scheduling policy that allocates one or more processing elements to a single virtual machine, but this policy implementation does not support sharing of processing elements (i.e.) all the requested resources will be used by the allocated VM till the time the VM is not destroyed. Also, Under this allocation policy, if

any virtual machine requests a processing element and is not available at that time, the allocation fails.

- VmSchedulerTimeSharedOverSubscription:
    - This is an extended implementation of VMSchedulerTimeShared VM scheduling policy, which allows over-subscription of processing elements by the virtual machine(s) (i.e.) the scheduler still allows the allocation of VMs that require more CPU capacity that is available. And this oversubscription results in performance degradation.
    - The application of the VmScheduler classes is while instantiating the host model.

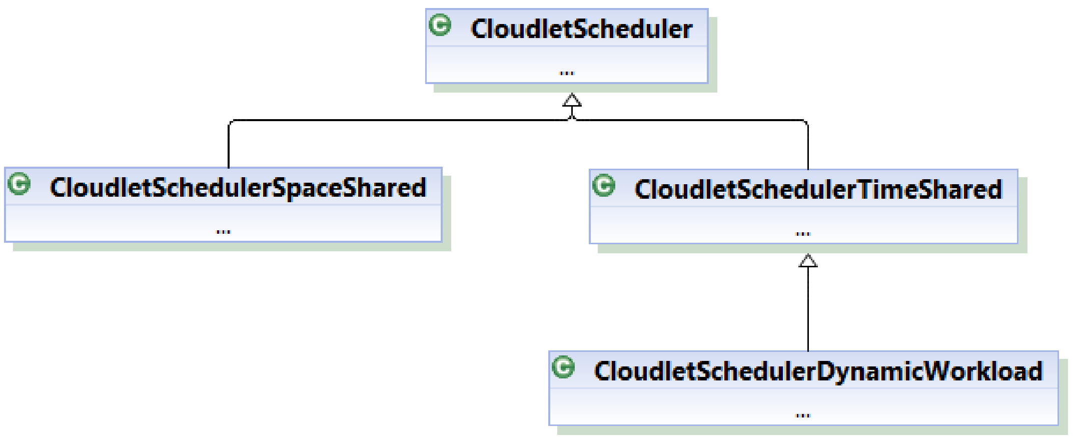
Following is the code snippet used in CloudsimExample1.java from line number 160 to 174:

```
int hostId = 0;

int ram = 2048; // host memory (MB)
long storage = 1000000; // host storage
int bw = 10000;


hostList.add(

                new Host(
                 hostId,

                 new RamProvisionerSimple(ram),
                 new BwProvisionerSimple(bw),
                 storage,
```

This is where the processing element list is passed as a parameter to the VmSchedulerTimeShared() class call and during the simulation, the cloudsim will simulate the timeshare behavior for the virtual machines. Also, in case you want to test other VmScheduler you may replace it with VmSchedulerTimeShared() call with appropriate parameters, this includes your own designed custom virtual machine scheduler.

Cloudsim Cloudlet Scheduling

The “CloudletScheduler” is an abstract class that defines the basic skeleton to implement the policy to be used for cloudlet scheduling to be performed by a virtual machine. The hierarchyof the cloudsim Cloudlet scheduler classes is as:


CloudletScheduler
...
CloudletSchedulerSpaceShared
...
CloudletSchedulerTimeShared
...
CloudletSchedulerDynamicWorkload
...

These classes again exist in “org.cloudbus.cloudsim” package of cloudsim. The definitionof this abstract class is extended as the following types of policies implemented as three individual classes in cloudsim:

- **CloudletSchedulerSpaceShared:** This class implements a policy of scheduling for Virtual machine to execute cloudlet(s) in space shared environment (i.e.) only one cloudlet will be executed on a virtual machine at a time. It means cloudlets share the same queue and requests are processed one at a time per computing core. Space-sharing is similar to batch processing.

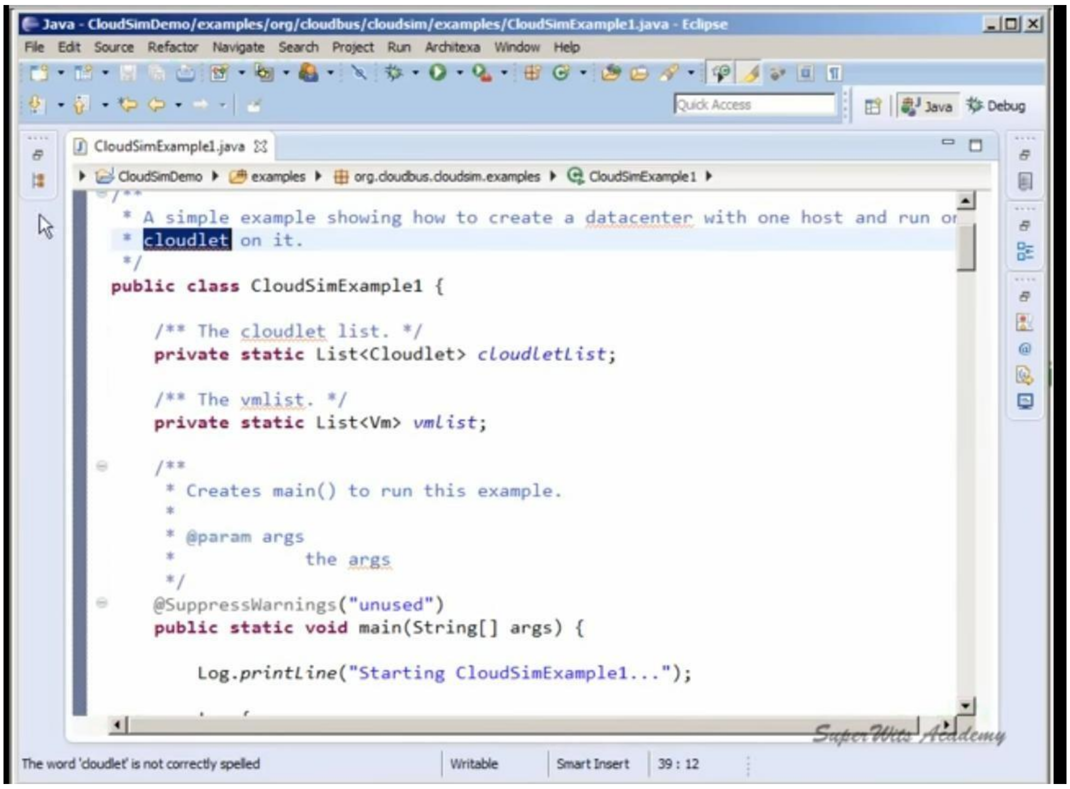
- **CloudletSchedulerTimeShared:** This class implements a policy of cloudlet scheduling for Virtual machines to execute cloudlets in a time-shared environment (i.e.) more than one cloudlet will be submitted to the virtual machine and each will get its specified share of time. It means several requests (cloudlets) are processed at once but they must share the computing power of that virtual machine(by simulating context switching), so they will affect each other‟s processing time. It basically influences the completion time of a cloudlet in CloudSim. Time-sharing is probably referring to the concept of sharingexecuting power (such as CPU, logical processor, GPU) and is commonly known as the round-robin scheduling.

- **CloudletSchedulerDynamicWorkload:** This implements a special policy of scheduling for virtual machine assuming that there is just one cloudlet which is working as an online service with a different requirement of workload as per the need of peak/offpeak user load at a specified period of time.

  - The application of the CloudletScheduler classes is while instantiating the Vm model.
  - Following is the code snippet used in CloudsimExample1.java from line number 82 to 91:

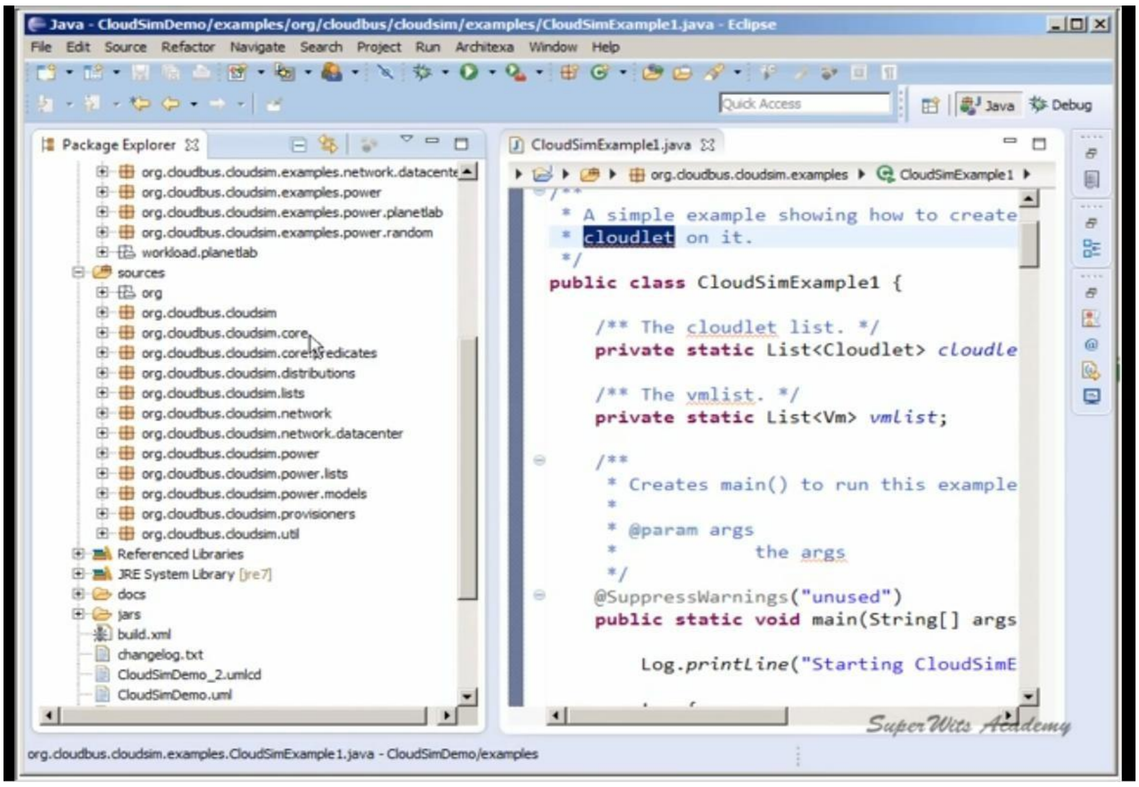
```
int vmid = 0;
int mips = 1000;

long size = 10000; // image size (MB)
int ram = 512; // vm memory (MB)

long bw = 1000;

int pesNumber = 1; // number of cpus
String vmm = "Xen"; // VMM name
```

- By instantiating the CloudletSchedulerTimeShared() class, the Virtual machine is decidedto follow the timeshare(round-robin) approach while simulation for scheduling & executing the Cloudlets. Also, in case you want to test other CloudletScheduler you may replace it with CloudletSchedulerTimeShared() call with appropriate parameters, this includes your own designed custom cloudlet scheduler.


Java - CloudSimDemo/examples/org/cloudbus/cloudsim/examples/CloudSimExample1.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Architexa Window Help
Quick Access
Java
Debug
CloudSimExample1.java
CloudSimDemo ▸ examples ▸ org.cloudbus.cloudsim.examples ▸ CloudSimExample1 ▸
SuperWits Academy
Writable
Smart Insert
39 : 12

```
/**
 * A simple example showing how to create a datacenter with one host and run or
 * cloudlet on it.
 */
public class CloudSimExample1 {

    /** The cloudlet list. */
    private static List<Cloudlet> cloudletList;

    /** The vmlist. */
    private static List<Vm> vmlist;

    /**
     * Creates main() to run this example.
     *
     * @param args
     *            the args
     */
    @SuppressWarnings("unused")
    public static void main(String[] args) {

        Log.printLine("Starting CloudSimExample1...");
```

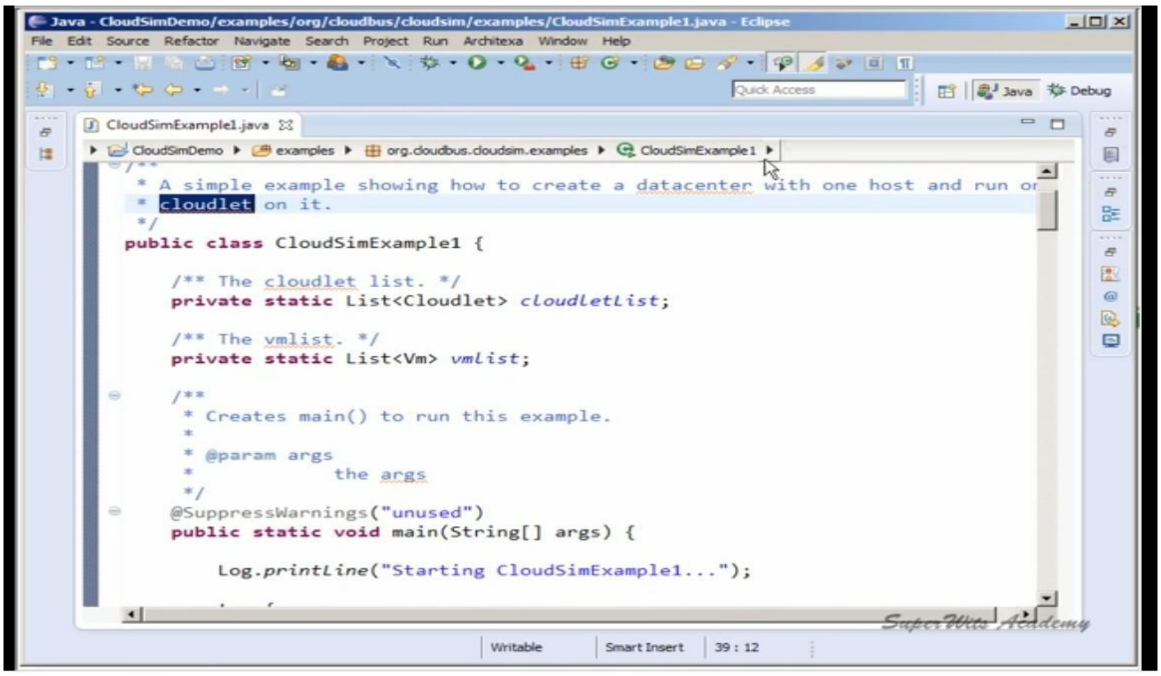
The word 'cloudlet' is not correctly spelled


Java - CloudSimDemo/examples/org/cloudbus/cloudsim/examples/CloudSimExample1.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Architexa Window Help
Quick Access
Java
Debug
Package Explorer
CloudSimExample1.java
SuperWits Academy

```
org.cloudbus.cloudsim.examples.network.datacente
org.cloudbus.cloudsim.examples.power
org.cloudbus.cloudsim.examples.power.planetlab
org.cloudbus.cloudsim.examples.power.random
workload.planetlab
sources
  org
  org.cloudbus.cloudsim
  org.cloudbus.cloudsim.core
  org.cloudbus.cloudsim.core.predicates
  org.cloudbus.cloudsim.distributions
  org.cloudbus.cloudsim.lists
  org.cloudbus.cloudsim.network
  org.cloudbus.cloudsim.network.datacenter
  org.cloudbus.cloudsim.power
  org.cloudbus.cloudsim.power.lists
  org.cloudbus.cloudsim.power.models
  org.cloudbus.cloudsim.provisioners
  org.cloudbus.cloudsim.util
Referenced Libraries
JRE System Library [jre7]
docs
jars
build.xml
changelog.txt
CloudSimDemo_2.umlcd
CloudSimDemo.uml
```

```java
/**
 * A simple example showing how to create
 * cloudlet on it.
 */
public class CloudSimExample1 {

    /** The cloudlet list. */
    private static List<Cloudlet> cloudle

    /** The vmlist. */
    private static List<Vm> vmlist;

    /**
     * Creates main() to run this example
     *
     * @param args
     *            the args
     */
    @SuppressWarnings("unused")
    public static void main(String[] args

        Log.printLine("Starting CloudSimE
```

org.cloudbus.cloudsim.examples.CloudSimExample1.java - CloudSimDemo/examples


Java - CloudSimDemo/examples/org/cloudbus/cloudsim/examples/CloudSimExample1.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Architexa Window Help
Quick Access
Java
Debug
CloudSimExample1.java
CloudSimDemo
examples
org.cloudbus.cloudsim.examples
CloudSimExample1
SuperWits Academy
Writable
Smart Insert
39 : 12

```java
/**
 * A simple example showing how to create a datacenter with one host and run o
 * cloudlet on it.
 */
public class CloudSimExample1 {

    /** The cloudlet list. */
    private static List<Cloudlet> cloudletList;

    /** The vmlist. */
    private static List<Vm> vmlist;

    /**
     * Creates main() to run this example.
     *
     * @param args
     *            the args
     */
    @SuppressWarnings("unused")
    public static void main(String[] args) {

        Log.printLine("Starting CloudSimExample1...");
```

RESULT:

Thus the round robin scheduling algorithm exists in CloudletSchedulerTimeShared() class has been executed using cloudsim and output was verified.