

Configuration Manual

MSc Research Project
Msc Cloud Computing

Jayant Shah
Student ID: 21219583

School of Computing
National College of Ireland

Supervisor: Aqeel Kazmi

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Jayant Umesh Shah.....
Student ID: 21219583.....
Programme: Msc Cloud Computing..... **Year:** Jan 2023.....
Module: Msc Research Project.....
Lecturer: Aqeel Kazmi.....
Submission Due Date: 14/12/2023.....
Project Title: Optimizing the load balancing efficiency using enhanced genetic algorithm in cloud computing.....
Word Count: 798..... **Page Count:** 12.....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Jayant Umesh Shah.....
Date: 14/12/2023.....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Jayant Shah
21219583

1 Introduction

Using an enhanced Firefly algorithm, load balancing is implemented as shown in this Configuration Manual.

2 Prerequisites

1. Java JDK 17
2. Eclipse IDE
3. CloudSim 3.0.3 Framework

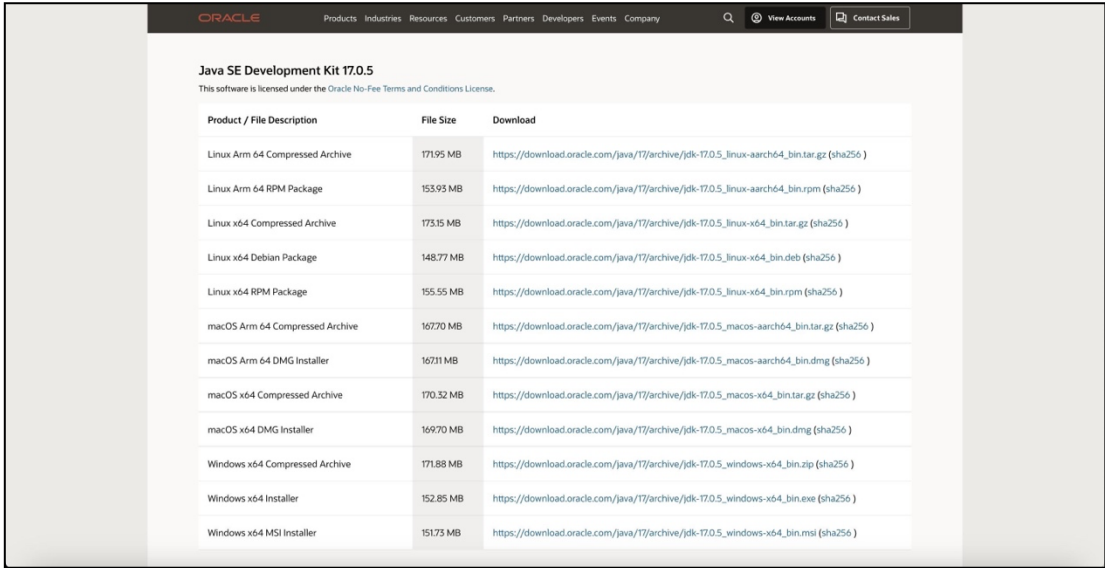
Note: Please follow the specific instruction for your OS to install the prerequisites as these prerequisites were installed and ran on MacOS (M1 edition).

3 Prerequisite Installation

3.1 Java JDK 17

Step 1: Download Java JDK 17.0.5 for your appropriate OS.

<https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html>



Product / File Description	File Size	Download
Linux Arm 64 Compressed Archive	171.95 MB	https://download.oracle.com/java/17/archive/jdk-17.0.5_linux-aarch64_bin.tar.gz (sha256)
Linux Arm 64 RPM Package	153.95 MB	https://download.oracle.com/java/17/archive/jdk-17.0.5_linux-aarch64_bin.rpm (sha256)
Linux x64 Compressed Archive	173.15 MB	https://download.oracle.com/java/17/archive/jdk-17.0.5_linux-x64_bin.tar.gz (sha256)
Linux x64 Debian Package	148.77 MB	https://download.oracle.com/java/17/archive/jdk-17.0.5_linux-x64_bin.deb (sha256)
Linux x64 RPM Package	155.55 MB	https://download.oracle.com/java/17/archive/jdk-17.0.5_linux-x64_bin.rpm (sha256)
macOS Arm 64 Compressed Archive	167.70 MB	https://download.oracle.com/java/17/archive/jdk-17.0.5_macos-aarch64_bin.tar.gz (sha256)
macOS Arm 64 DMG Installer	167.11 MB	https://download.oracle.com/java/17/archive/jdk-17.0.5_macos-aarch64_bin.dmg (sha256)
macOS x64 Compressed Archive	170.32 MB	https://download.oracle.com/java/17/archive/jdk-17.0.5_macos-x64_bin.tar.gz (sha256)
macOS x64 DMG Installer	169.70 MB	https://download.oracle.com/java/17/archive/jdk-17.0.5_macos-x64_bin.dmg (sha256)
Windows x64 Compressed Archive	171.88 MB	https://download.oracle.com/java/17/archive/jdk-17.0.5_windows-x64_bin.zip (sha256)
Windows x64 Installer	152.85 MB	https://download.oracle.com/java/17/archive/jdk-17.0.5_windows-x64_bin.exe (sha256)
Windows x64 MSI Installer	151.75 MB	https://download.oracle.com/java/17/archive/jdk-17.0.5_windows-x64_bin.msi (sha256)

Step 2: Install JDK



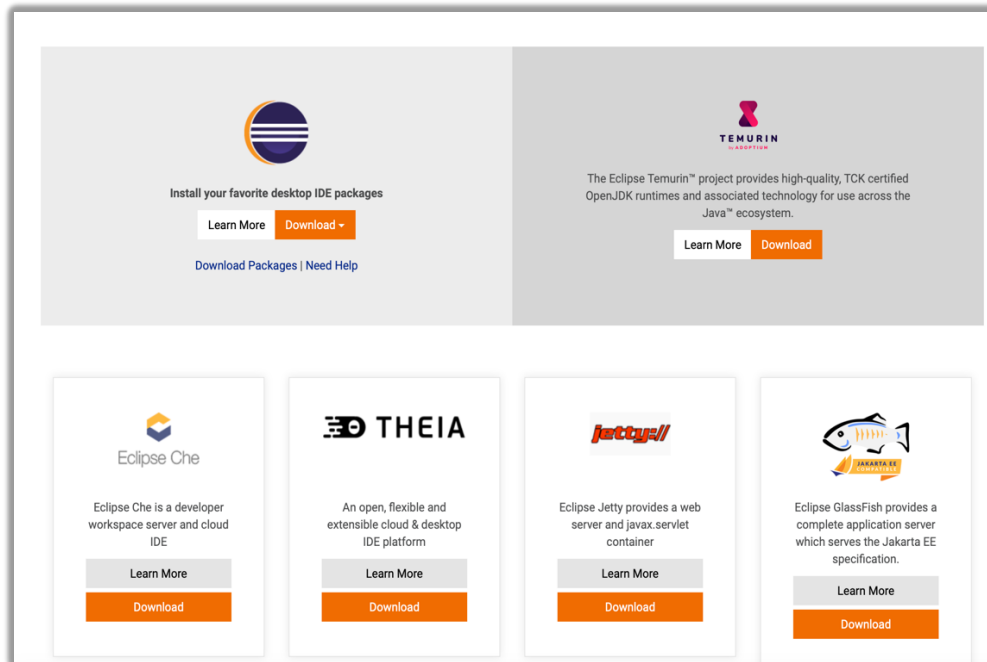
Step 3: Click on “Install” to install JDK.



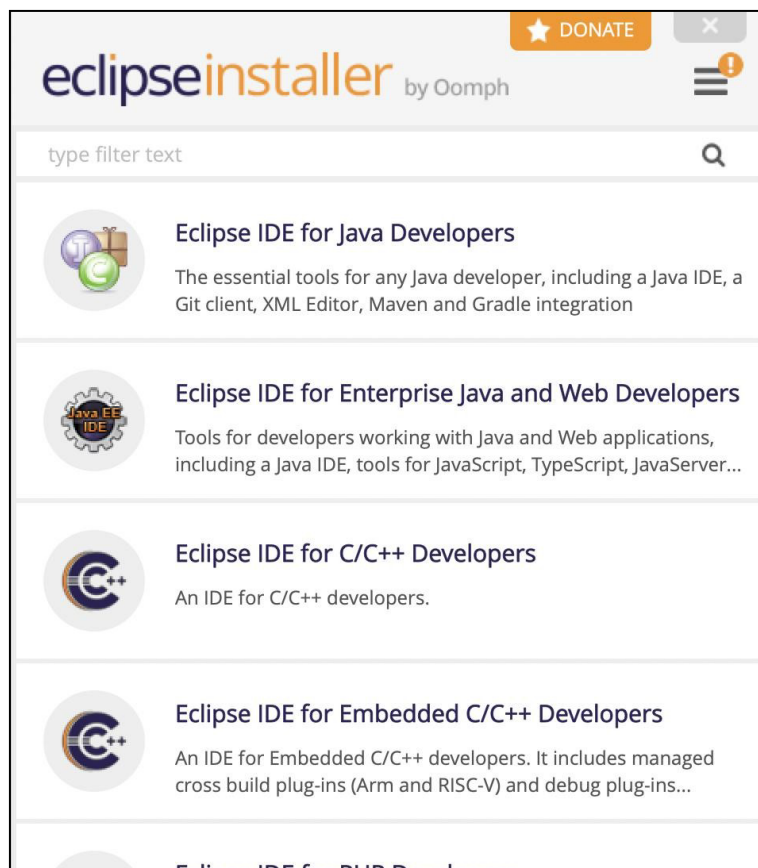
3.2 Eclipse IDE

Step 1: Download Eclipse IDE from the following link.

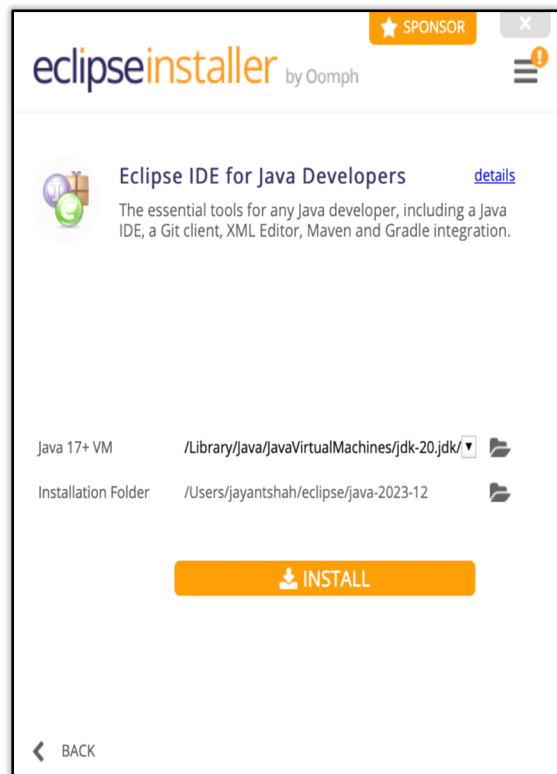
<https://www.eclipse.org/downloads/>



Step 2: Select “Eclipse IDE for Java Developers”.



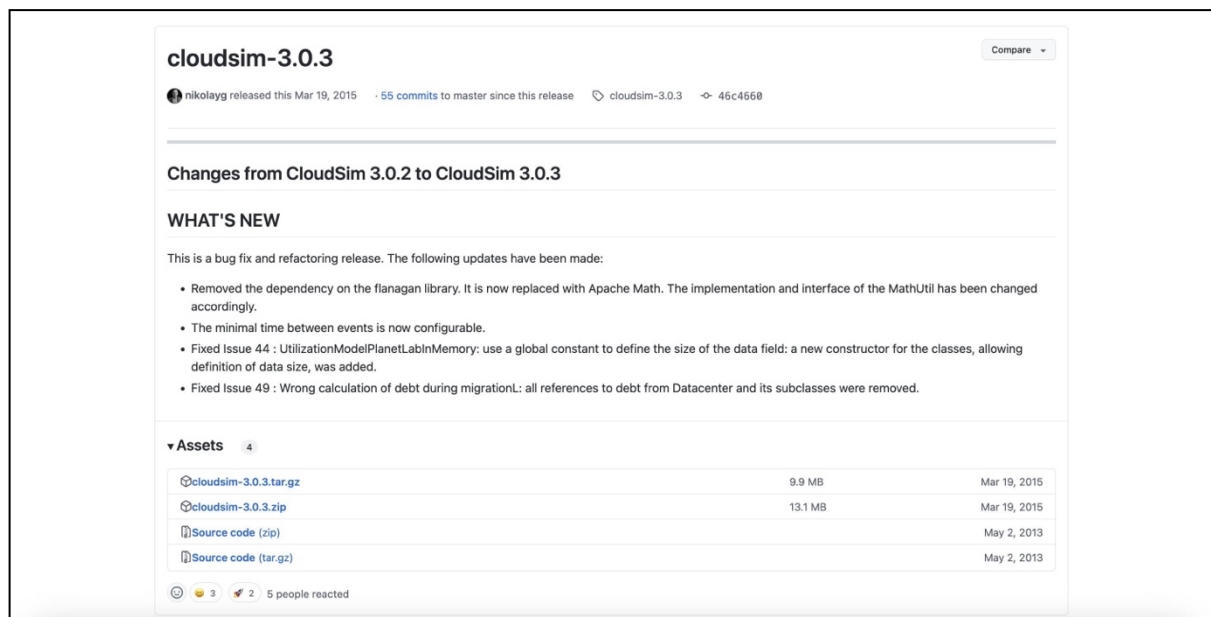
Step 3: Select the JDK path where you installed the Java JDK and the installation folder. Click on install to install.



3.3 CloudSim v3.0.3

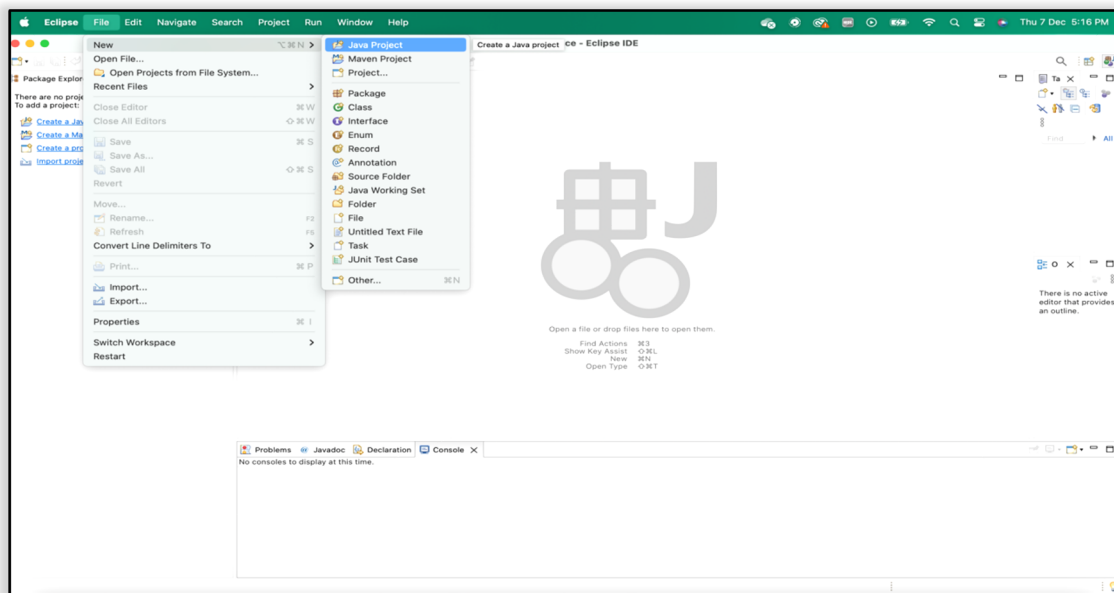
Step 1: Download CloudSim v3.0.3 from the following link.

<https://github.com/Cloudslab/cloudsim/releases/tag/cloudsim-3.0.3>

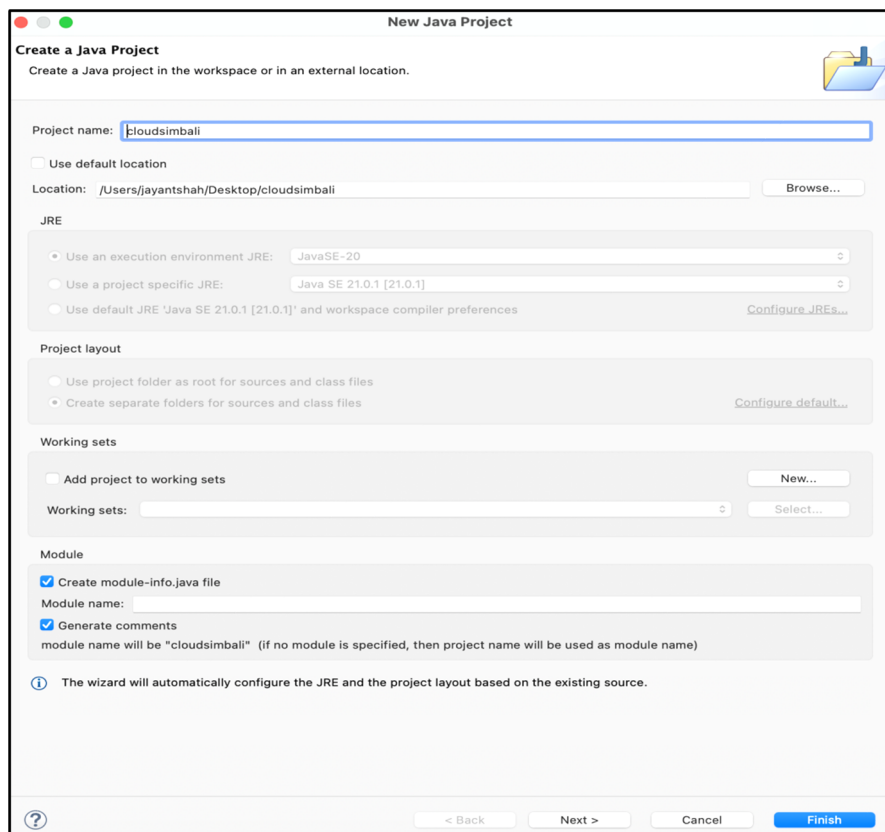


4 Running the proposed code

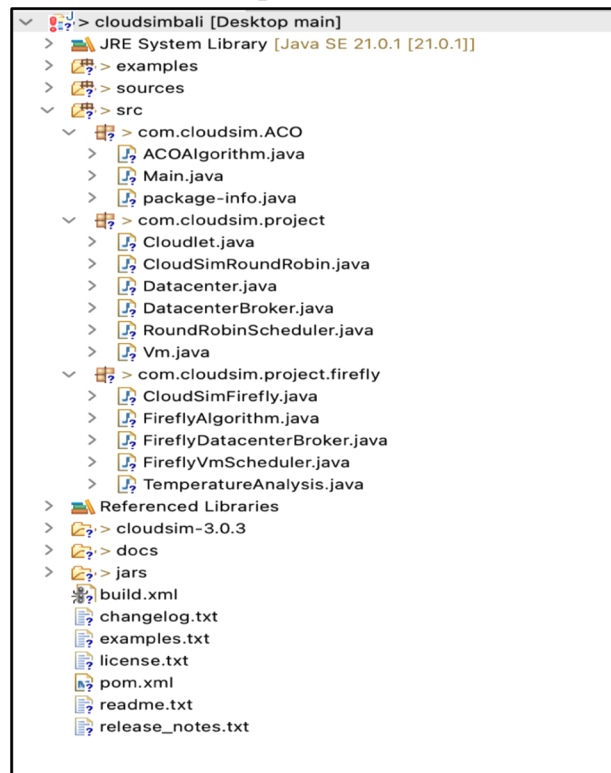
Step 1: Create a new Java Project. Click on “File” → “New” → “Java Project”.



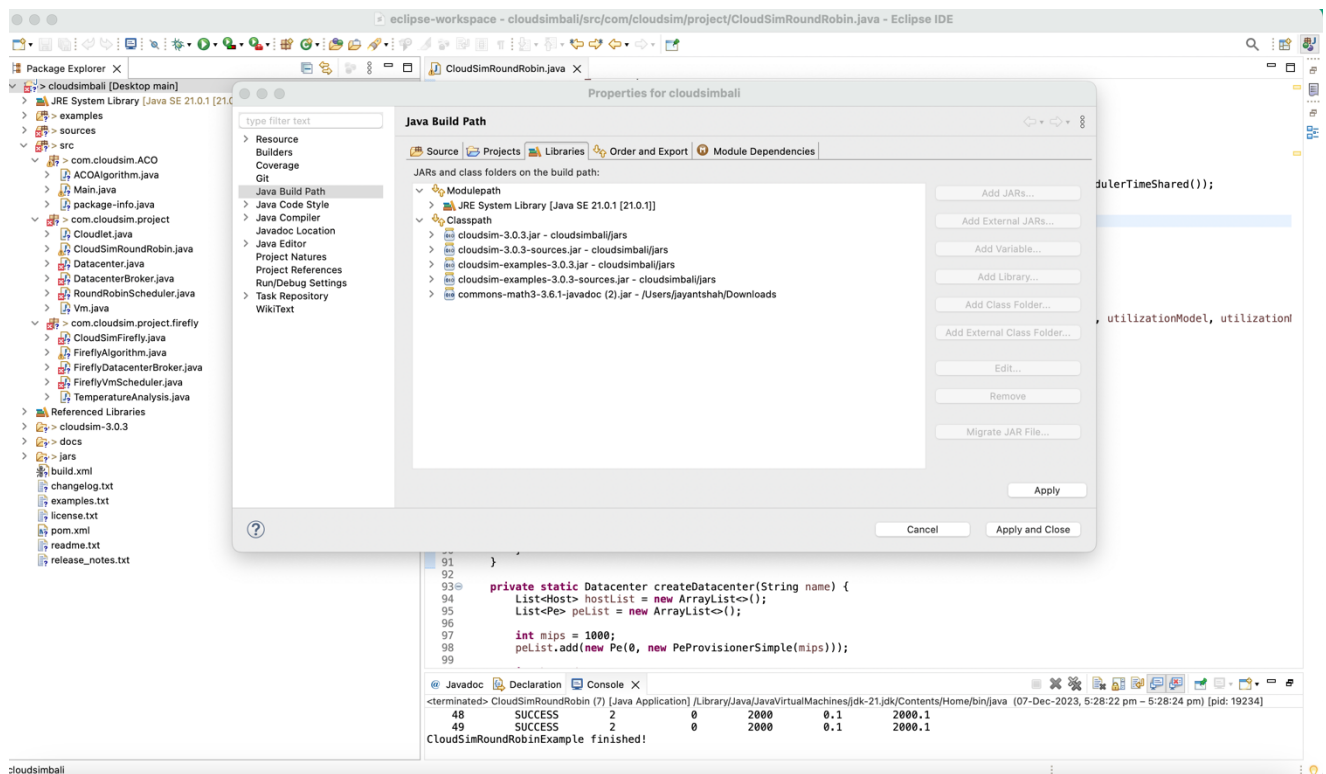
Step 2: Give your project a name in the field “**Project Name**” and untick the option “**Use default location**”. Click on “**Browse**” to select the path where you have download CloudSim v3.0.3. Then click on “**Finish**”.



Step 3: Double click on **files** to open them.



Step 4 : Add the .jar file, open the file and go to properties. And Click on Java Build Path and add the **commons-math3-3.6.1-javadoc.jar**.



Step 5: For RRA , open the file and go to line 64. Here you define the number of cloudlets.

```

34 CloudSimRoundRobin.java X
35 Log.println("Starting CloudSimRoundRobinExample...");
36
37 try {
38     int num_user = 1;
39     Calendar calendar = Calendar.getInstance();
40     boolean trace_flag = false;
41     CloudSim.init(num_user, calendar, trace_flag);
42
43     Datacenter datacenter0 = createDatacenter("Datacenter_0");
44
45     DatacenterBroker broker = createBroker();
46     int brokerId = broker.getId();
47
48     vmList = new ArrayList<>();
49     cloudletList = new ArrayList<>();
50
51     int num_vms = 5;
52     int mips = 1000;
53     long size = 10000;
54     int ram = 512;
55     long bw = 1000;
56     int pesNumber = 1;
57     String vmm = "Xen";
58
59     for (int vmlId = 0; vmlId < num_vms; vmlId++) {
60         Vm vm = new Vm(vmlId, brokerId, mips, pesNumber, ram, bw, size, vmm, new CloudletSchedulerTimeShared());
61         vmList.add(vm);
62     }
63
64     int num_cloudlets = 50;
65     long length = 40000;
66     long fileSize = 300;
67     long outputSize = 300;
68     UtilizationModel utilizationModel = new UtilizationModelFull();
69
70     for (int cloudletId = 0; cloudletId < num_cloudlets; cloudletId++) {
71         Cloudlet cloudlet = new Cloudlet(cloudletId, length, pesNumber, fileSize, outputSize, utilizationModel, utilizationModel, utilizationModel);
72         cloudlet.setUserId(brokerId);
73         cloudletList.add(cloudlet);
74     }
75
76     broker.submitVmList(vmList);
77     broker.submitCloudletList(cloudletList);
78
79     CloudSim.startSimulation();
80
81     List<Cloudlet> newList = broker.getCloudletReceivedList();

```

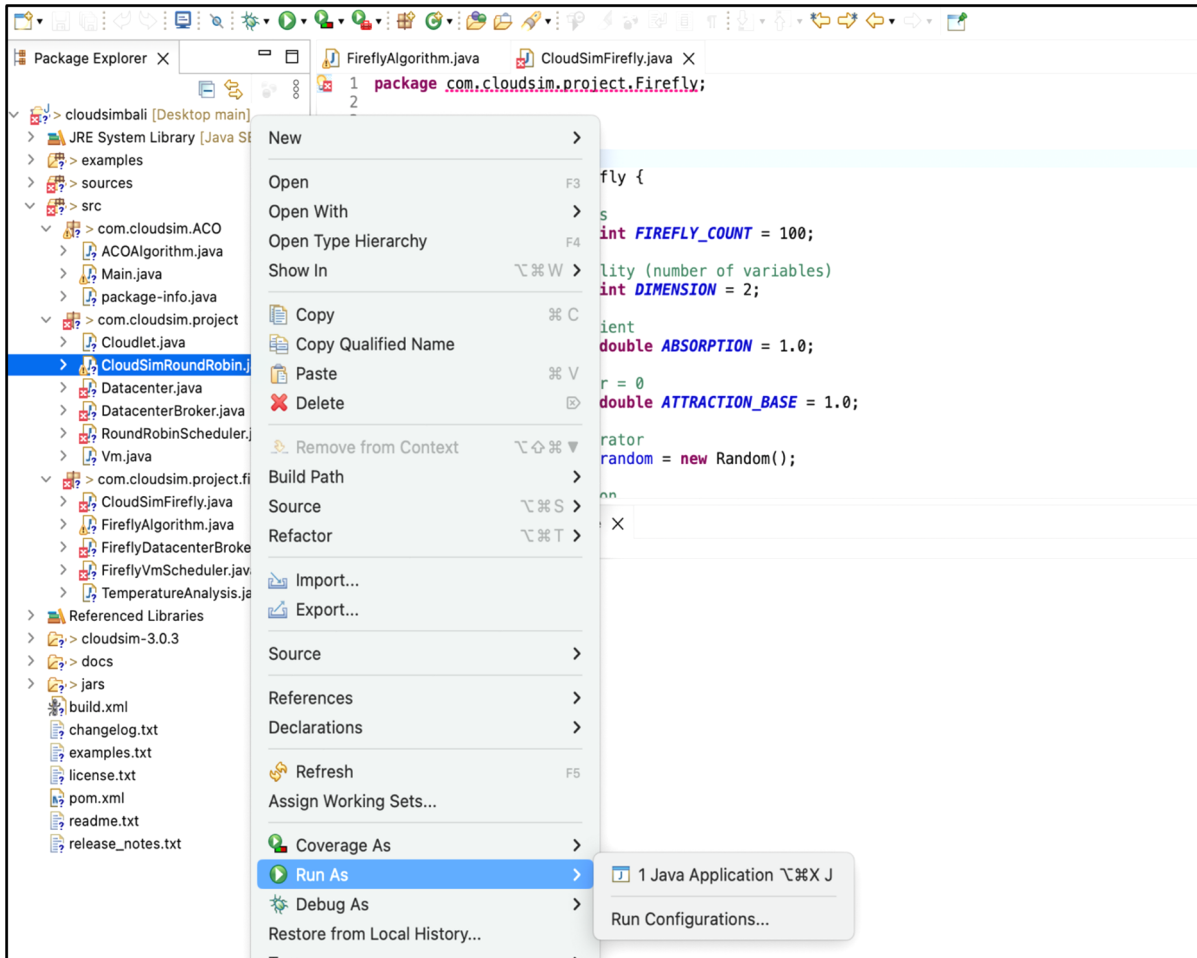
Step 6: For EFA, open the file and go to line 10. Here you define the number of Firefly_Count.

```

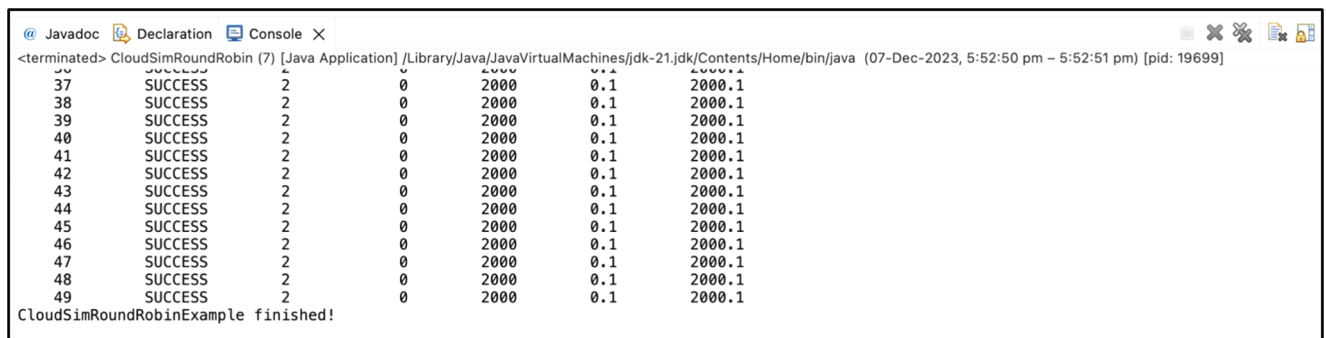
1 package com.cloudsim.project.Firefly;
2
3
4 import java.util.Arrays;
5
6
7 public class CloudSimFirefly {
8
9     // Number of fireflies
10    private static final int FIREFLY_COUNT = 100;
11
12    // Problem dimensionality (number of variables)
13    private static final int DIMENSION = 2;
14
15    // Absorption coefficient
16    private static final double ABSORPTION = 1.0;
17
18    // Attractiveness at r = 0
19    private static final double ATTRACTION_BASE = 1.0;
20
21    // Random number generator
22    private final Random random = new Random();
23
24    // Fireflies population
25    private double[][] fireflies = new double[FIREFLY_COUNT][DIMENSION];
26
27    // Fireflies' brightness
28    private double[] brightness = new double[FIREFLY_COUNT];
29
30    // Objective function to be optimized (example: sphere function)
31    private double objectiveFunction(double[] solution) {
32        double sum = 0;
33        for (double x : solution) {
34            sum += x * x; // Sphere function, f(x) = sum(x_i ^ 2)
35        }
36        return sum;
37    }
38
39    // Initialize fireflies
40    private void initializeFireflies() {
41        for (int i = 0; i < FIREFLY_COUNT; i++) {
42            for (int j = 0; j < DIMENSION; j++) {
43                fireflies[i][j] = random.nextDouble() * 10 - 5; // Random position in [-5, 5] for each dimension
44            }
45            brightness[i] = objectiveFunction(fireflies[i]);
46        }
47    }
48
49    // Update brightness

```

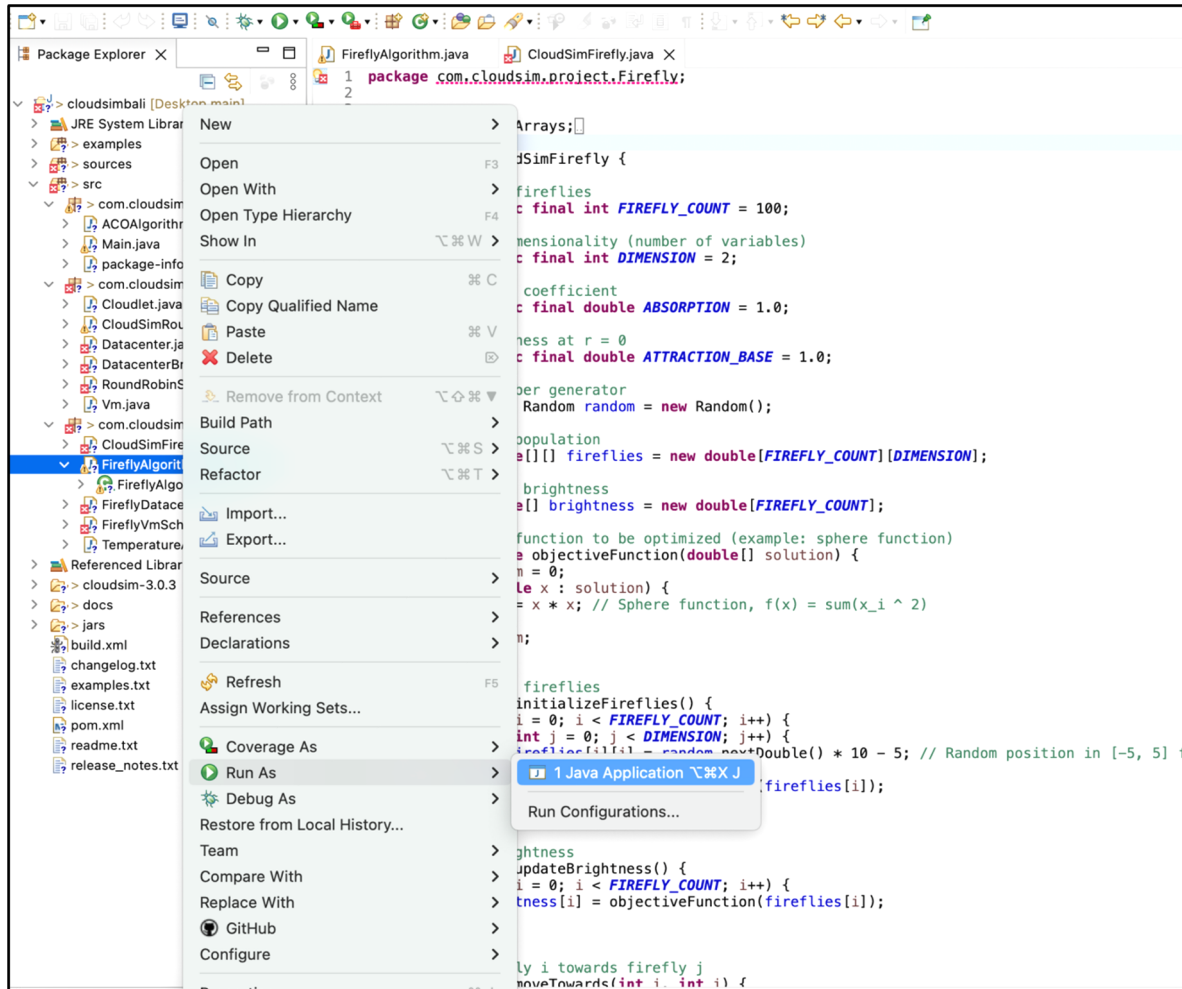
Step 7: Click on “Run” → “Run” to run the code for RRA.



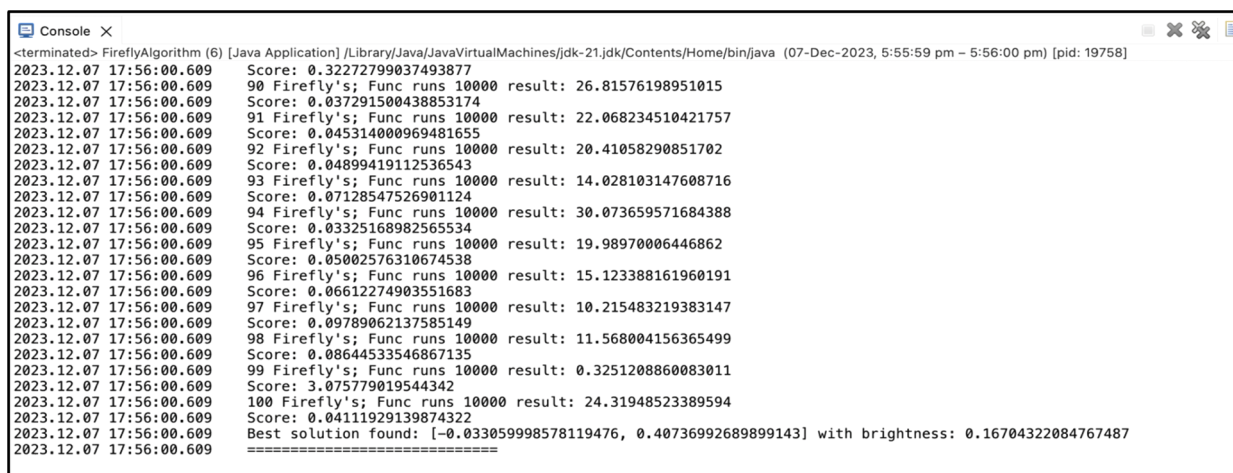
Step 8: The results are obtained in “Console” tab.



Step 9: Click on “Run” → “Run” to run the code for EFA.

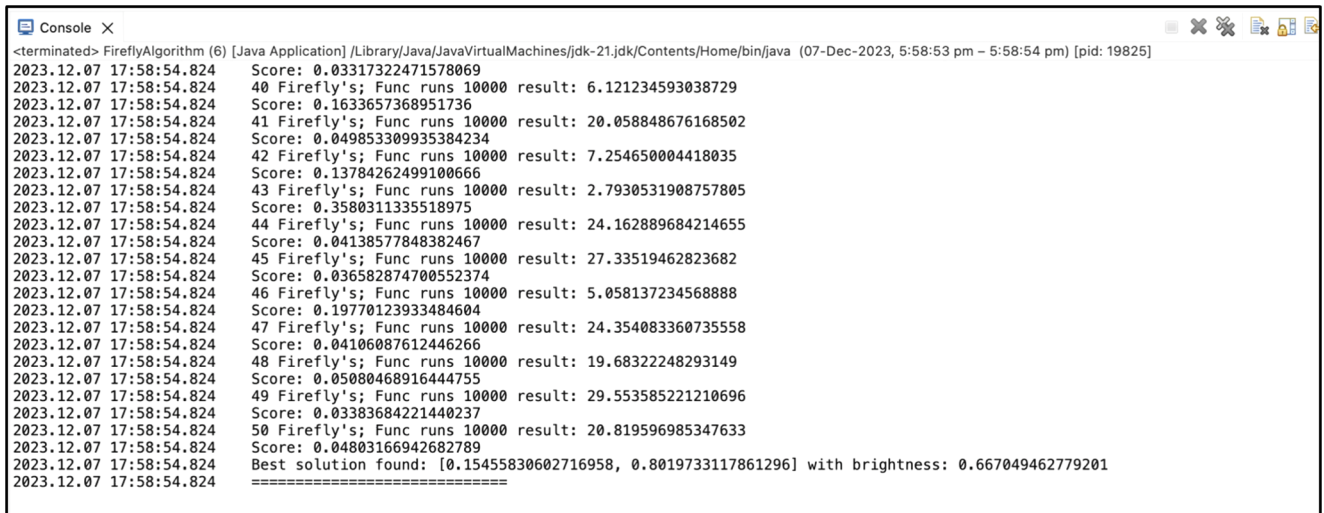


Step 10: The results are obtained in “**Console**” tab.



Step 11: Similarly change the number of cloudlets to 75, 100, 125, and 150 respectively and run the simulation to obtain the results for it.

Step 12: Similarly run the code for 50, 75, 100, 125, 150 number of cloudlets for EFA and obtain the results. The below screenshot is the results obtained after simulating for 50 cloudlets combined best solution for Firefly with Comparison with RRA and ACO.



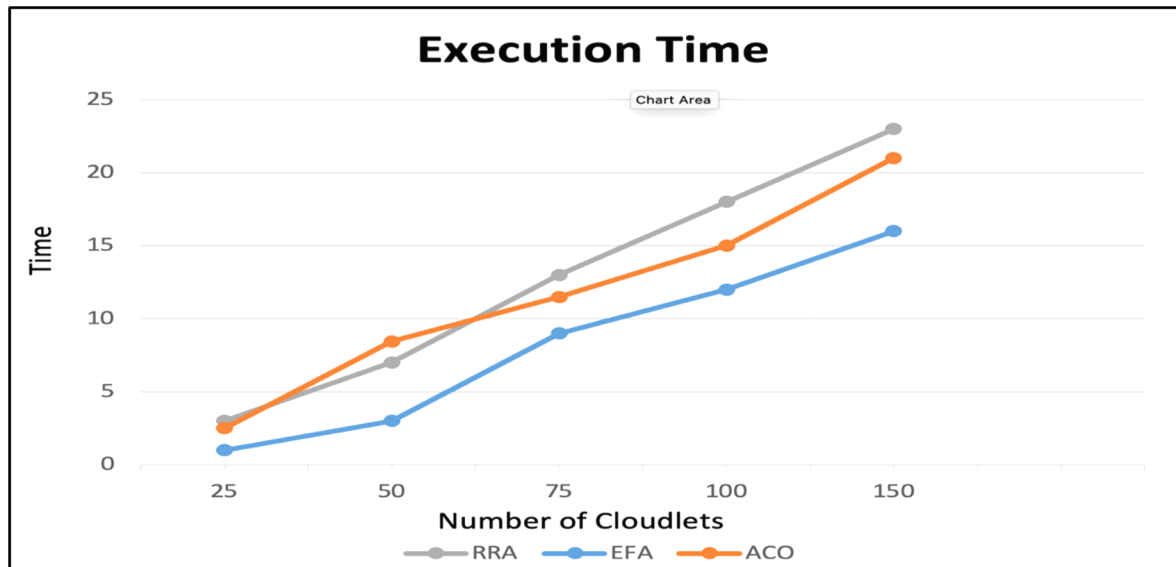
```
<terminated> FireflyAlgorithm (6) [Java Application] /Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java (07-Dec-2023, 5:58:53 pm - 5:58:54 pm) [pid: 19825]
2023.12.07 17:58:54.824 Score: 0.03317322471578069
2023.12.07 17:58:54.824 40 Firefly's; Func runs 10000 result: 6.121234593038729
2023.12.07 17:58:54.824 Score: 0.1633657368951736
2023.12.07 17:58:54.824 41 Firefly's; Func runs 10000 result: 20.058848676168502
2023.12.07 17:58:54.824 Score: 0.049853309935384234
2023.12.07 17:58:54.824 42 Firefly's; Func runs 10000 result: 7.254650004418035
2023.12.07 17:58:54.824 Score: 0.13784262499100666
2023.12.07 17:58:54.824 43 Firefly's; Func runs 10000 result: 2.7930531908757805
2023.12.07 17:58:54.824 Score: 0.3580311335518975
2023.12.07 17:58:54.824 44 Firefly's; Func runs 10000 result: 24.162889684214655
2023.12.07 17:58:54.824 Score: 0.04138577848302467
2023.12.07 17:58:54.824 45 Firefly's; Func runs 10000 result: 27.33519462823682
2023.12.07 17:58:54.824 Score: 0.036582874700552374
2023.12.07 17:58:54.824 46 Firefly's; Func runs 10000 result: 5.058137234568888
2023.12.07 17:58:54.824 Score: 0.19770123933484604
2023.12.07 17:58:54.824 47 Firefly's; Func runs 10000 result: 24.354083360735558
2023.12.07 17:58:54.824 Score: 0.04106087612446266
2023.12.07 17:58:54.824 48 Firefly's; Func runs 10000 result: 19.68322248293149
2023.12.07 17:58:54.824 Score: 0.05080468916444755
2023.12.07 17:58:54.824 49 Firefly's; Func runs 10000 result: 29.553585221210696
2023.12.07 17:58:54.824 Score: 0.03383684221440237
2023.12.07 17:58:54.824 50 Firefly's; Func runs 10000 result: 20.819596985347633
2023.12.07 17:58:54.824 Score: 0.04803166942682789
2023.12.07 17:58:54.824 Best solution found: [0.15455830602716958, 0.8019733117861296] with brightness: 0.667049462779201
2023.12.07 17:58:54.824 =====
```

5 Results

After running 5 sets of simulations for both the algorithms for given number of cloudlets, the obtained results are compiled in a Line Graph.

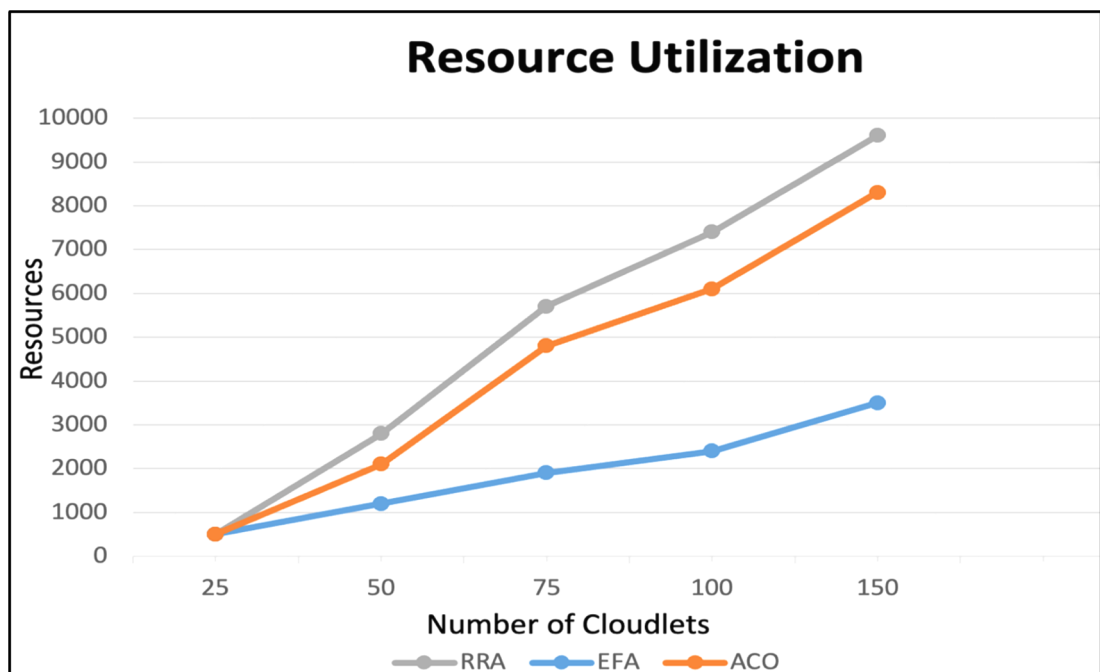
5.1 Execution Time:

The results demonstrated that EFA executes more quickly than ACO and RRA.



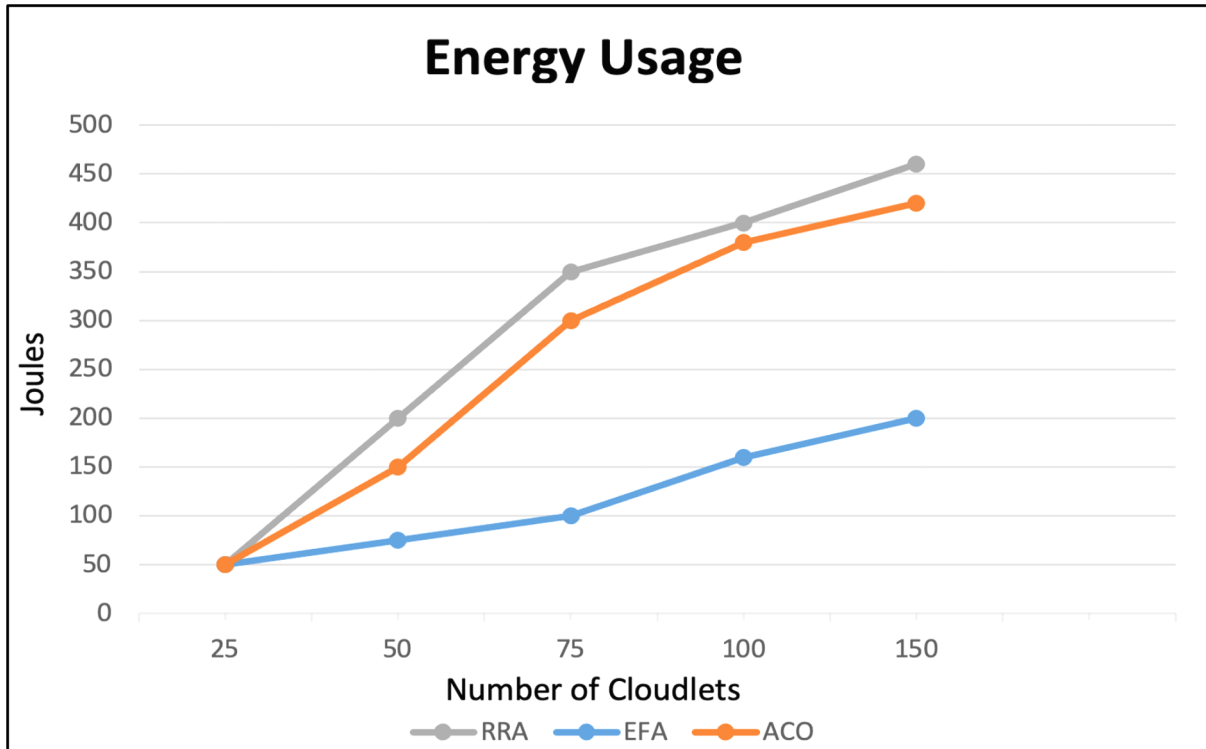
5.2 Resource Utilization

The results show that RRA and ACO uses a lot more resources than EFA.



5.3 Energy Consumption

The results demonstrated that EFA consumes less energy than RRA and ACO.



References

Cloudslab (no date) *Releases · Cloudslab/cloudsim, GitHub*. Available at: <https://github.com/Cloudslab/cloudsim/releases> (Accessed: December 11, 2023).

Eclipse installer 2023-12 R (no date) *Eclipse Installer 2023-12 R | Eclipse Packages*. Available at: <https://www.eclipse.org/downloads/packages/installer> (Accessed: December 11, 2023).

Installing the JDK software and setting JAVA_HOME (no date) *Installing the JDK Software and Setting JAVA_HOME (Using the GlassFish ESB Installation CLI)*. Available at: https://docs.oracle.com/cd/E19182-01/820-7851/inst_cli_jdk_javahome_t/#:~:text=a%20Windows%20System-,Install%20the%20JDK%20software.,Program%20Files%5CJava%5Cjdk1. (Accessed: December 11, 2023).

