San Francisco State University

Department of Computer Science

Culminating Experience Report

**Title**: Analysis of Accuracy of Queuing Models

**Author**: Ping Xiao

**Date**: 4/15/2014

**Abstract**:

The primary goal of this project is to develop a simulator of four types of queuing systems: single server model, multiple servers model, feedback queuing model and interactive system model with general distribution of interarrival time and general service time distributions. In the case of exponential distributions the work of each simulator can be validated using analytic models. In cases that use a general distribution our goal is to evaluate the robustness of exponential queuing models as predictors of performance indicators of such systems. The simulators produce numerical results and histograms for various distributions such as interarrival time, inter-departure time, response time and so on. This technique is used to demonstrate how feedback in queuing networks creates non-exponential flows of service requests even in the case of Poisson arrival process. The results obtained using simulation are compared with analytic models and used for systematic analysis of errors and a study of robustness of queuing models of open and closed queuing networks.

**Keywords**: Simulator, Single server model, Multiple servers model, Feedback queuing model, Interactive system, Robustness of queuing models.

**TABLE OF CONTENTS**

[1. Introduction 1](#_Toc393238373)

[1.1. Analysis of queuing system 1](#_Toc393238374)

[1.2. Contribution of this project 3](#_Toc393238375)

[1.3. Report outline 3](#_Toc393238376)

[2. Design concept of this project 5](#_Toc393238377)

[2.1. Discrete –event simulator 5](#_Toc393238378)

[2.2. Graphical User Interface design 6](#_Toc393238379)

[3. Simulator of single server systems 9](#_Toc393238380)

[3.1. Single server model 9](#_Toc393238381)

[3.1.1. Distributions of M/M/1 Model 10](#_Toc393238382)

[3.1.2. Distributions of M/D/1 Model 12](#_Toc393238383)

[3.1.3. Distributions of M/G/1 Model 14](#_Toc393238384)

[3.1.4. Distributions of G/G/1 Model 16](#_Toc393238385)

[3.2. Analysis of accuracy of simulation models 19](#_Toc393238386)

[3.2.1. Distributions of M/M/1 Model 19](#_Toc393238387)

[3.2.2. Distributions of M/D/1 Model 21](#_Toc393238388)

[3.2.3. Distributions of M/G/1 Model 23](#_Toc393238389)

[3.2.4. Distributions of G/G/1 Model 25](#_Toc393238390)

[4. Simulator of multiple servers model 29](#_Toc393238391)

[4.1. Multiple servers model 29](#_Toc393238392)

[4.2. Analysis of accuracy of simulation models 34](#_Toc393238393)

[5. Simulator of feedback system 37](#_Toc393238394)

[5.1. Feedback queuing model 37](#_Toc393238395)

[5.2. Analysis of accuracy of simulation models 46](#_Toc393238396)

[6. Simulator of interactive system 50](#_Toc393238397)

[6.1. Interactive System 50](#_Toc393238398)

[6.2. Analysis of accuracy of simulator models 53](#_Toc393238399)

[7. Conclusions and future work 57](#_Toc393238400)

[7.1. Conclusions 57](#_Toc393238401)

[7.2. Future work 58](#_Toc393238402)

# Introduction

## Analysis of queuing system

A Queuing system consists of customers (tasks), servers (processors or resources), and a queue. Customers arrive at the system at a rate with defined statistical properties. A queue forms because arriving customers may have to wait other customers for service. Customers leave the system after service. [1] There are a lot of examples that we are familiar with can be described as queuing system, such as supermarket checkout, airport check-in, customer service in a bank and so on.

Based on the above three important components, queuing system can also be described as these characteristics: interarrival distribution, service distribution, interdeparture distribution, response time distribution and so on. Interdeparture time distribution and response time distribution depend on the interarrival distribution and service distribution.

There are there basic distributions for interarrival time and service time:

**Exponential distribution (M)**, it is a probability distribution that describes the time between events in a passion process. This distribution is the most important one in queuing theory because of the following reasons:

* Assuming it leads to reasonable expression for the queuing parameters;
* It is a good approximation to many distributions that are otherwise too different to handle;
* Assuming it is usually (but not always) pessimistic;
* Assuming this distribution is equivalent to saying that arrival customers individually and collectively behave as if they are not aware of each other’s existence.

**General distribution (GI)**, in this project, we use uniform distribution, it is another important distribution characterized by the mean and the variance.

**Deterministic distribution (D)**, the interarrival or service time is a constant.

**Terminology and notation:**

S = mean service time with coefficient of variation

a = mean interarrival time with coefficient of variation

X = 1/ a = throughput (input arrival and output departure rate)

U = SX = S/a = server utilization

R = response time

W = R – S = wait time

## Contribution of this project

This project is comprised of following major contributions:

1. Develop a graphical user interface in Java so that the user can specify distributions of interarrival and service times and then see on the screen the corresponding distributions of whatever result the user wants to see (primarily response time, inter-arrival and inter-departure times at any point the user selects)
2. Use that tool for detailed analysis of accuracy of analytic approximations of queuing models.
3. Develop the same solution for both the G/G/1 and feedback model.
4. Deliver the tool to users over the Internet.
5. Provide the users with the possibility to get accurate results if they want to wait to have 1 million or more jobs processed.
6. Errors analysis of G/G/1 formula.
7. Non exponential distribution in feedback network with exponential arrivals.
8. Differences between non-exponential results and exponential analytical results for cases different from G/G/1.

## Report outline

The reminder of this report is organized in the following manner.

Chapter 2 describes discrete-event simulation method for this project and an introduction to the design of GUI in the main display windows.

Chapter 3 presents the distributions and experimental results for M/M/1, M/D/1, M/G/1, G/G/1 model, with comparison of the simulation results and analytical results to verify the accuracy of the single server simulator.

Chapter 4 presents the distributions and experimental results for multiple servers model, with comparison of the simulation results and analytical results to verify the accuracy of multiple servers model simulator.

Chapter 5 presents the distributions and experimental results for feedback queuing model, with comparison of the simulation results and analytical results to verify the accuracy of feedback queuing model simulator.

Chapter 6 presents the distributions and experimental results for interactive system, with comparison of the simulation results and analytical results to verify the accuracy of interactive system simulator.

Chapter 7 includes the conclusion and suggestions for future work.

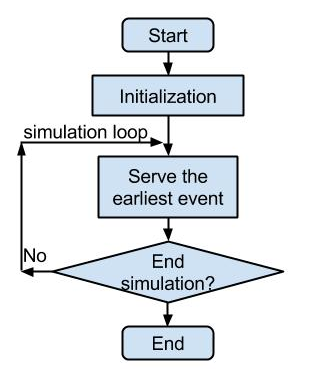
# Design concept of this project

In this chapter, we describe the details approaches to implement the simulators. Also, we briefly discuss the design of the Graphical User Interface (GUI) of the simulators.

## Discrete –event simulator

The major approach to implement the simulators is discrete-event simulator (DES) which models the operation of a system as a discrete sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system. Discrete-event simulators are simulation programs which simulate the dynamic behavior of a given class of discrete-event dynamic systems. [2] For example, in a single server model, there are two discrete events, job arrival and job departure. Hence the primary task of every discrete-event simulator is to maintain one or more lists of expected discrete events. The events are usually sorted by the anticipated time of occurrence. The simulator program first selects the earliest event and then performs all activities related to the select events. Some of these activities may cause expected events and such events are to be properly included in corresponding lists of expected events. For example, in a single server model, a job arrival event will cause a job departure event.

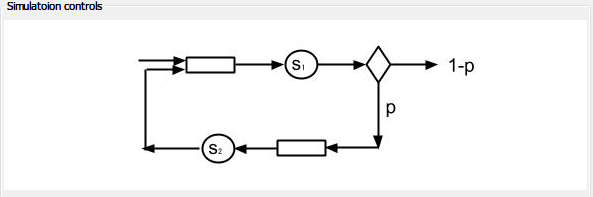
Figure 2-1 shows the basic simulator structure design.



*Figure 2-1: Basic simulator design*

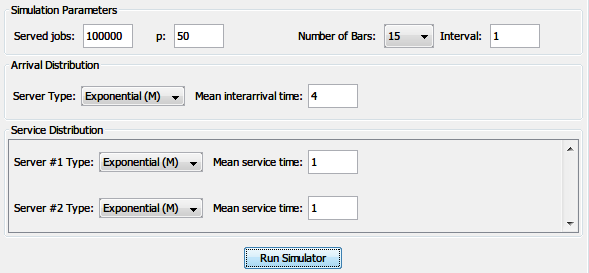
## Graphical User Interface design

There are four parts for each simulator, the first part (Figure 2-2) is an image shows which simulator you are using.



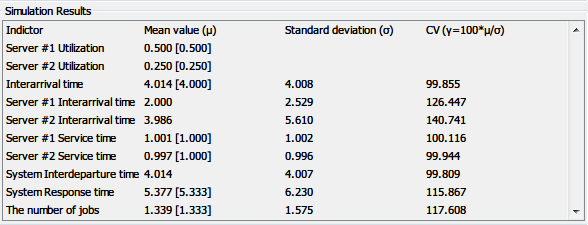
*Figure 2-2*

The second part (Figure 2-3) includes all the parameters to run a simulator. The user can change the number of jobs, mean interarrival time, mean service time and so on. After the parameters are all set, the user clicks “Run Simulator”, the numerical result and charts will be displayed.



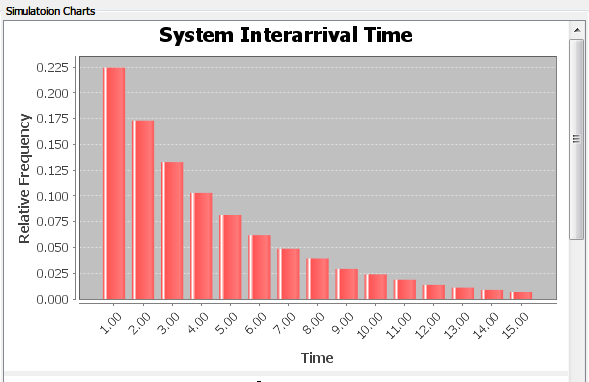
*Figure 2-3*

The third part (Figure 2-4) shows the numerical result.



*Figure 2-4*

The fourth part (Figure 2-5) shows the distribution of interarrival time, response time, inter-departure time and the number of customer in the system.

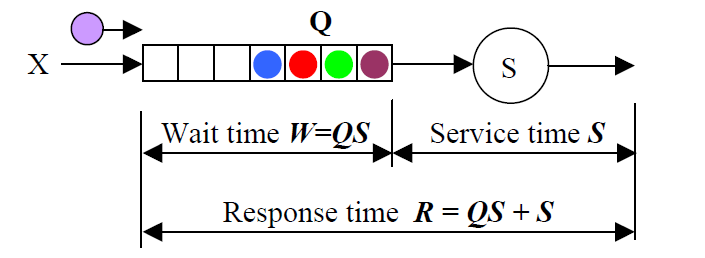


*Figure 2-5*

# Simulator of single server systems

## Single server model

In essence, all complex queuing systems are comprised of individual single server queuing systems which consist of a single queue and a single server (as shown below).



*Figure 3-1: Single server queuing model*

In order to make a single server simulator, we need information relating to: arrival rate, service rate and queue length.

With different arrival rate and different service rate, there are different models.

**M/M/1 Model**: the interarrival time distribution is exponential distribution and the service time distribution is exponential distribution.

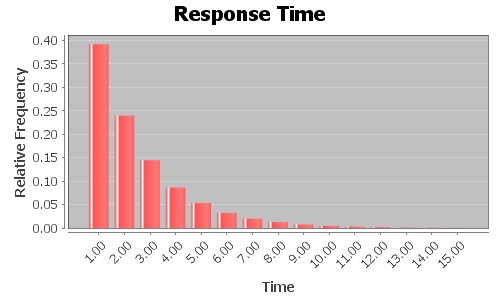
**M/G/1 Model**: the interarrival time distribution is exponential distribution and the service time distribution is uniform distribution.

**M/D/1 Model**: the interarrival time distribution is exponential distribution and the service time distribution is deterministic distribution.

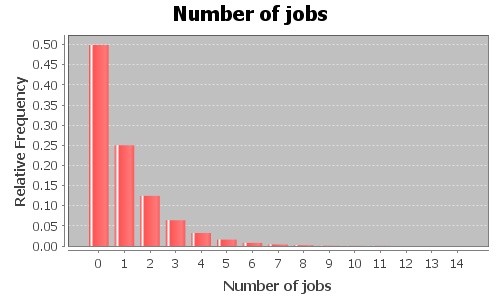
**GI/G/1 Model**: the interarrival time distribution is uniform distribution and the service time distribution is uniform distribution...

### Distributions of M/M/1 Model

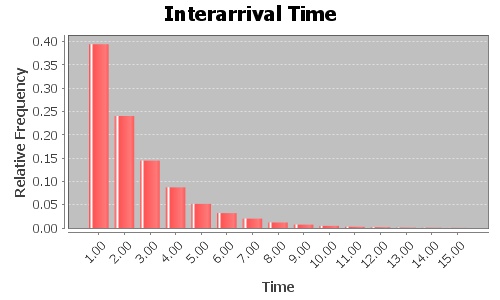
In M/M/1 Model, Figure 3-2 to Figure 3-5 show response time distribution, the distribution of average number of jobs in the system, interarrival time distribution and inter departure time distribution with parameters a = 2, S = 1.



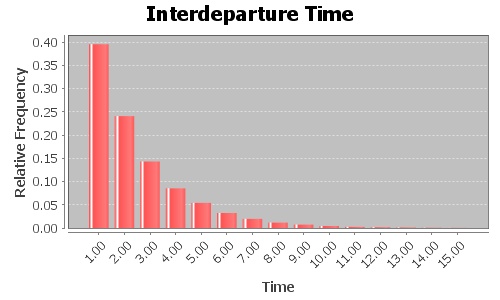
*Figure 3-2: Response Time distribution*



*Figure 3-3: Distribution of average number of jobs in the system*



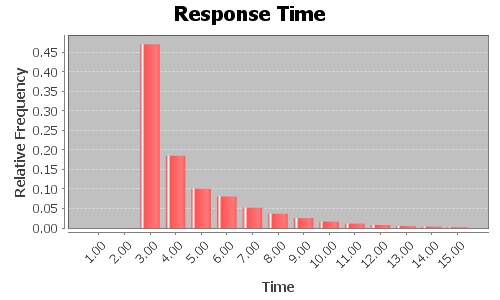
*Figure 3-4: interarrival time distribution*

**

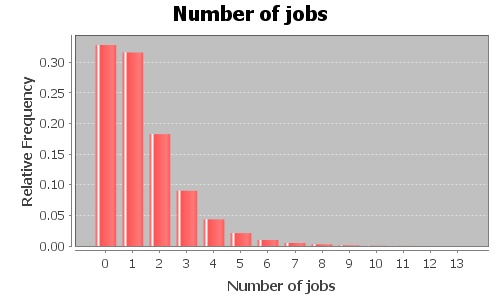
*Figure 3-5: inter departure time distribution*

### Distributions of M/D/1 Model

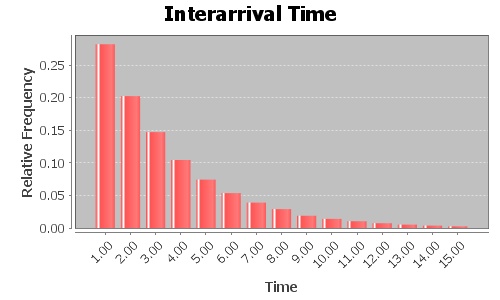
In M/D/1 Model, Figure 3-6 to Figure 3-9 show response time distribution, the distribution of average number of jobs in the system, interarrival time distribution and inter departure time distribution with parameters a = 3, S = 2.



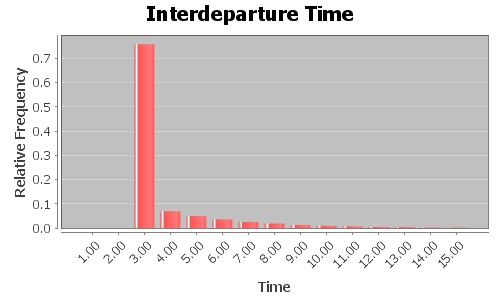
*Figure 3-6: Response time distribution*

**

*Figure 3-7: Distribution of average number of jobs in the system*



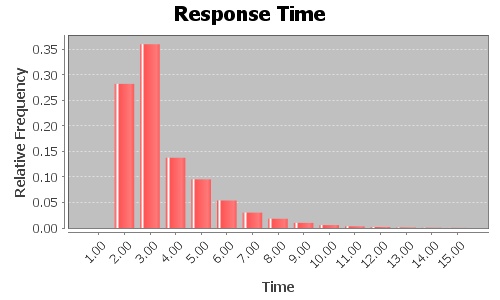
*Figure 3-8: Interarrival time distribution*

**

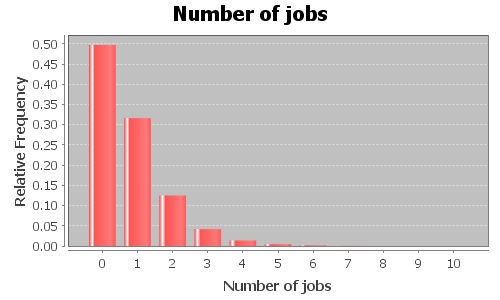
*Figure 3-9: Inter departure time distribution*

### Distributions of M/G/1 Model

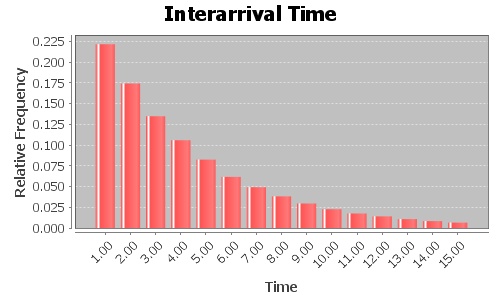
In M/G/1 Model, Figure 3-10 to Figure 3-13 show response time distribution, the distribution of average number of jobs in the system, interarrival time distribution and inter departure time distribution with parameters a = 4, S = 2 (from 1 to 3).



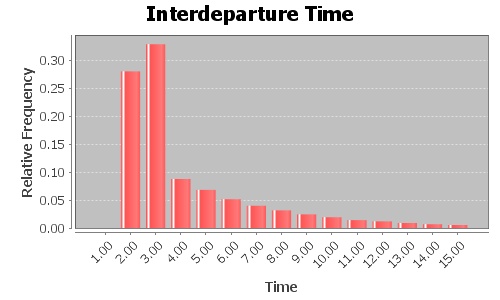
*Figure 3-10: Response time distribution*



*Figure 3-11: Distribution of average number of jobs in the system*



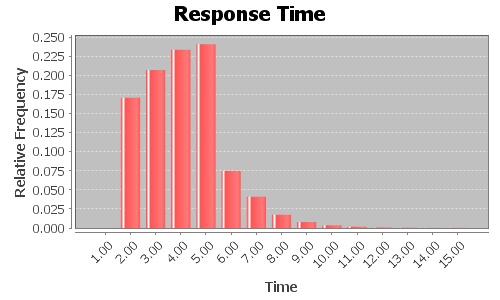
*Figure 3-12: Interarrival time distribution*



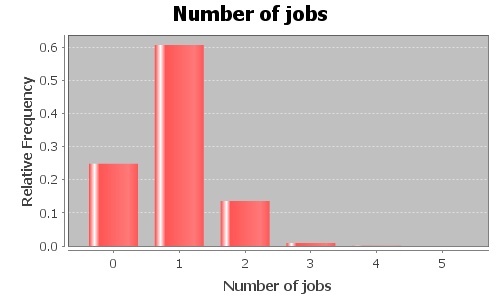
*Figure 3-13: Inter departure time distribution*

### Distributions of G/G/1 Model

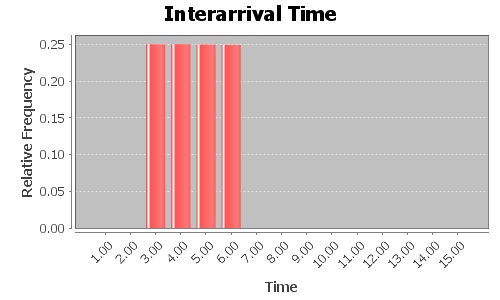
In G/G/1 Model, Figure 3-14 to Figure 3-17 show response time distribution, the distribution of average number of jobs in the system, interarrival time distribution and inter departure time distribution with parameters a = 4 (from 2 to 6), S = 3 (from 1 to 5).



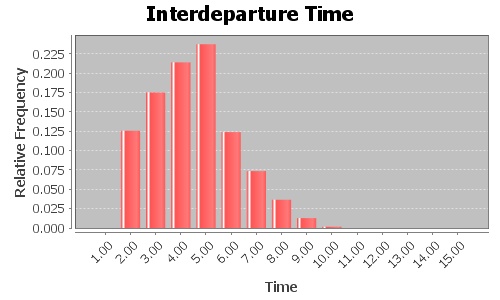
*Figure 3-14: Response time distribution*



*Figure 3-15: Distribution of average number of jobs in the system*



*Figure 3-16: Interarrival time distribution*



*Figure 3-17: Inter departure time distribution*

## Analysis of accuracy of simulation models

We will use a group of experiments with different parameters to verify the accuracy of simulation models.

### Distributions of M/M/1 Model

The formulas to verify M/M/1 Model are listed as below:

U = S / a;

R = S / (1 - U);

Q = U / (1 - U);

Table 3-1 shows experimental result with different parameters.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | S | AU | SU | E [%] | AR | SR | E [%] | AQ | SQ | E [%] |
| 2 | 0.1 | 0.05 | 0.05 | 0.171 | 0.105 | 0.106 | 0.383 | 0.053 | 0.053 | 0.421 |
| 2 | 0.3 | 0.15 | 0.15 | 0.297 | 0.353 | 0.353 | 0.049 | 0.176 | 0.177 | 0.409 |
| 2 | 0.5 | 0.25 | 0.249 | -0.394 | 0.667 | 0.664 | -0.42 | 0.333 | 0.332 | -0.513 |
| 2 | 0.7 | 0.35 | 0.349 | -0.299 | 1.077 | 1.073 | -0.399 | 0.538 | 0.535 | -0.62 |
| 2 | 0.9 | 0.45 | 0.453 | 0.567 | 1.636 | 1.668 | 1.908 | 0.818 | 0.837 | 2.288 |
| 2 | 1.1 | 0.55 | 0.555 | 0.902 | 2.444 | 2.494 | 2.047 | 1.222 | 1.256 | 2.712 |
| 2 | 1.3 | 0.65 | 0.651 | 0.084 | 3.714 | 3.71 | -0.109 | 1.857 | 1.854 | -0.171 |
| 2 | 1.5 | 0.75 | 0.752 | 0.314 | 6 | 6.064 | 1.06 | 3 | 3.044 | 1.433 |
| 2 | 1.7 | 0.85 | 0.846 | -0.493 | 11.333 | 10.731 | -5.317 | 5.667 | 5.359 | -5.734 |
| 2 | 1.9 | 0.95 | 0.952 | 0.263 | 38 | 37.956 | -0.116 | 19 | 18.928 | -0.381 |

*Table 3-1: M/M/1 experimental result*

*Figure* 3-18 and Figure 3-19 show the comparison between analytical result and simulation result based on the table 3-1.

*Figure 3-18: M/M/1 Utilization with data in table 3.6.1*

*Figure 3-19: M/M/1 Response time with data in table 3.6.1*

### Distributions of M/D/1 Model

The formulas to verify M/D/1 Model are listed as below:

U = S / a;

R = S \* (2 - U) / (2 \* (1 - U));

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | S | AU | SU | E [%] | AR | SR | E [%] | AQ | SQ | E [%] |
| 4 | 0.3 | 0.075 | 0.075 | -0.543 | 0.312 | 0.312 | -0.033 | 0.078 | 0.078 | -0.579 |
| 4 | 0.7 | 0.175 | 0.175 | 0.274 | 0.774 | 0.775 | 0.076 | 0.194 | 0.194 | 0.348 |
| 4 | 1.1 | 0.275 | 0.276 | 0.219 | 1.309 | 1.309 | 0.014 | 0.327 | 0.328 | 0.233 |
| 4 | 1.5 | 0.375 | 0.375 | -0.05 | 1.95 | 1.946 | -0.222 | 0.488 | 0.486 | -0.274 |
| 4 | 1.9 | 0.475 | 0.472 | -0.565 | 2.76 | 2.74 | -0.723 | 0.69 | 0.681 | -1.298 |
| 4 | 2.3 | 0.575 | 0.575 | 0.082 | 3.856 | 3.844 | -0.312 | 0.964 | 0.962 | -0.23 |
| 4 | 2.7 | 0.675 | 0.677 | 0.345 | 5.504 | 5.524 | 0.371 | 1.376 | 1.386 | 0.712 |
| 4 | 3.1 | 0.775 | 0.775 | -0.021 | 8.439 | 8.361 | -0.923 | 2.11 | 2.09 | -0.952 |
| 4 | 3.5 | 0.875 | 0.874 | -0.122 | 15.75 | 15.689 | -0.387 | 3.937 | 3.918 | -0.508 |
| 4 | 3.9 | 0.975 | 0.976 | 0.053 | 79.95 | 74.693 | -6.576 | 19.987 | 18.687 | -6.961 |

Q = U \* U / (2 \* (1 - U)) + U;

Table 3-2 shows experimental result with different parameters.

*Table 3-2: M/D/1 experimental result*

*Figure* 3-20 and Figure 3-21 show the comparison between analytical result and simulation result based on the table 3-2.

*Figure 3-20: M/D/1 Utilization*

*Figure 3-21: M/D/1 Response time*

### Distributions of M/G/1 Model

The formulas to verify M/D/1 Model are listed as below:

S = (sMin + sMax) / 2;

U = S / a;

R = (S / (1 - U)) \* (1 – 0.5 \* U \* (1 – vs \* vs));

Q = U + (U \* U + vs \* (1 / a) \* (1 / a)) / (2 \* (1 - U));

Table 3-3 shows the experimental result with different parameters.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | S | AU | SU | E [%] | AR | SR | E [%] | AQ | SQ | E [%] |
| 4 | (1, 2.8) | 0.475 | 0.477 | 0.327 | 2.824 | 2.814 | -0.351 | 0.706 | 0.705 | -0.113 |
| 4 | (1, 3.2) | 0.525 | 0.524 | -0.209 | 3.367 | 3.358 | -0.254 | 0.835 | 0.839 | 0.469 |
| 4 | (1, 3.6) | 0.575 | 0.574 | -0.215 | 4.022 | 3.992 | -0.738 | 0.988 | 0.997 | 0.893 |
| 4 | (1, 4.0) | 0.625 | 0.626 | 0.128 | 4.833 | 4.786 | -0.985 | 1.175 | 1.199 | 2.018 |
| 4 | (1, 4.4) | 0.675 | 0.674 | -0.158 | 5.874 | 5.893 | 0.314 | 1.411 | 1.472 | 4.145 |
| 4 | (1, 4.8) | 0.725 | 0.727 | 0.208 | 7.27 | 7.232 | -0.515 | 1.724 | 1.812 | 4.88 |
| 4 | (1, 5.2) | 0.775 | 0.776 | 0.186 | 9.256 | 9.161 | -1.022 | 2.164 | 2.294 | 5.661 |
| 4 | (1, 5.6) | 0.825 | 0.822 | -0.357 | 12.338 | 12.2 | -1.121 | 2.841 | 3.048 | 6.768 |
| 4 | (1, 6.0) | 0.875 | 0.873 | -0.198 | 17.833 | 17.713 | -0.674 | 4.041 | 4.428 | 8.754 |
| 4 | (1, 6.4) | 0.925 | 0.92 | -0.533 | 30.567 | 29.898 | -2.187 | 6.805 | 7.441 | 8.554 |

*Table 3-3 M/G/1 experimental result*

Figure 3-22 and Figure 3-23 show the comparison between analytical result and simulation result based on the table 3.6.3.

*Figure 3-22: M/G/1 Utilization*

*Figure 3-23: M/G/1 Response time*

### Distributions of G/G/1 Model

The formulas to verify M/D/1 Model are listed as below:

aMin = minimum interarrival time;

aMax = maximum interarrival time;

sMin = minimum service time;

sMax = maximum service time;

a = (aMin + aMax) / 2;

S = (sMin + sMax) / 2;

U = S / a;

R (S / (1 - U)) \* (1 – 0.5 \* U \* (1 – vs \* vs – ((vs \* vs + 1) \* (va \* va - 1)/ (U \* U \* vs \* vs + 1))));

Table 3-4 shows the experimental result with different parameters.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | S | AU | SU | E [%] | AR | SR | E [%] |
| (1, 14) | (2, 3) | 0.333 | 0.333 | 0.058 | 2.659 | 2.605 | 2.093 |
| (1, 14) | (2, 4) | 0.4 | 0.4 | 0.077 | 3.264 | 3.22 | 1.389 |
| (1, 14) | (2, 5) | 0.467 | 0.467 | -0.069 | 3.923 | 3.892 | 0.789 |
| (1, 14) | (2, 6) | 0.533 | 0.534 | -0.149 | 4.663 | 4.643 | 0.422 |
| (1, 14) | (2, 7) | 0.6 | 0.598 | 0.404 | 5.532 | 5.485 | 0.848 |
| (1, 14) | (2, 8) | 0.667 | 0.667 | -0.012 | 6.615 | 6.581 | 0.512 |
| (1, 14) | (2, 9) | 0.733 | 0.732 | 0.242 | 8.084 | 8.035 | 0.611 |
| (1, 14) | (2, 10) | 0.8 | 0.795 | 0.595 | 10.344 | 10.116 | 2.258 |
| (1, 14) | (2, 11) | 0.867 | 0.869 | -0.25 | 14.602 | 14.744 | -0.966 |
| (1, 14) | (2, 12) | 0.933 | 0.934 | -0.082 | 26.9 | 27.619 | -2.603 |

*Table 3-4: Experimental result*

Figure 3-24 and Figure 3-25 show the comparison between analytical result and simulation result based on the table 3-4.

*Figure 3-24: G/G/1 Utilization*

*Figure 3-25: G/G/1 Response time*

Next, we varied the parameters for G/G/1 model. This time we make aMin = 0, sMin = 0 and R = sMax / aMax. Table 3-5 shows the average error rate for different R.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | R | | | | | | | | |
| E [%] | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| aMax = 1 | 1.085 | 2.049 | 2.761 | 3.149 | 3.279 | 2.931 | 2.436 | 1.866 | 2.635 |
| aMax = 10 | 1.076 | 2.058 | 2.802 | 3.155 | 3.245 | 2.994 | 2.435 | 2.022 | 2.714 |
| aMax = 100 | 1.098 | 2.055 | 2.792 | 3.193 | 3.246 | 2.921 | 2.521 | 2.006 | 2.742 |

*Table 3-5: Error rate based on different R*

Figure 3-26 shows the comparison between analytical result and simulation result based on the table 3-5.

*Figure 3-26: Error of G/G/1 formula*

Based on the experimental result and the utilization and response time figure of M/M/1, M/D/1, M/G/1, we can see that after 1,000,000 events, the analytical results are very close to the simulation results, with the error rate less than 5%. Because all the analytical result are correctly, we can conclude that the simulator for the single server model is numerically correct.

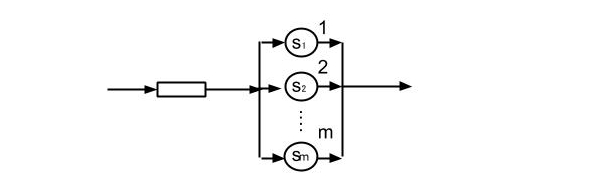
For the G/G/1 Model, the analytical formula for response time is approximate. Since the simulator is numerically correct. We can use simulator to verify the accuracy the analytical formula for response time. We can see from table 3-4, with 10 experiments, the error rate for response time is less than 3%, so we can also conclude the approximate analytical formula for response time is numerically correct.

In table 3-5 and figure 3-36, we can see that with different R, the error rate is very stable. When R < 0.5 and 0.8 < R < 1, the error rate is going up, when R > 0.5 and R < 0.8, the error rate is going down. So the worst case happens when R around 0.5 where error rate is about 3.5 percent. So the approximate analytical formula is good.

# Simulator of multiple servers model

## Multiple servers model

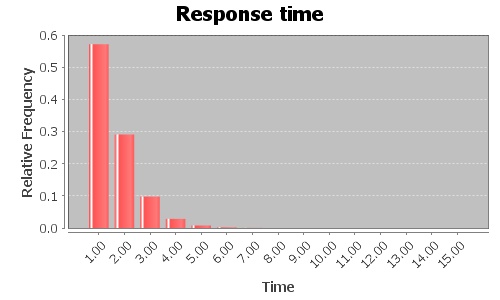
In the case of exponential interarrival and service times a single server model M/M/1 has the response time R = S / (1 - SX). If the arrival / departure rate X is constant, then it is limited by X < 1 / S, and the only way to increase the performance is to have a faster sever. Of course, the server speed is always limited, either by the financial limitations, of by the current technology limits. The only way to go beyond these limits is to use multiple servers (Figure 4-1).



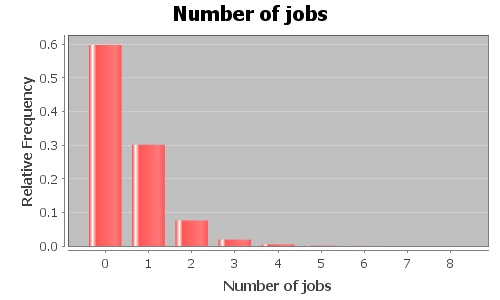
*Figure 4-1: Single queue multiple server model*

We assume that the servers are equivalent and that the flow of service requests is perfectly balanced. If the number of servers is m, the system called M/M/m model.

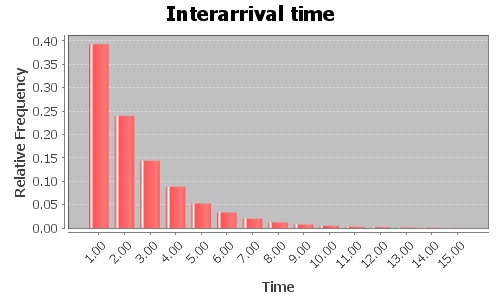
Figure 4-2 to figure 4-5 show the distribution of response time, customers in the system, interarrival time and inter departure time with parameters m = 2 (m is number of servers), a = 2, S = 1.



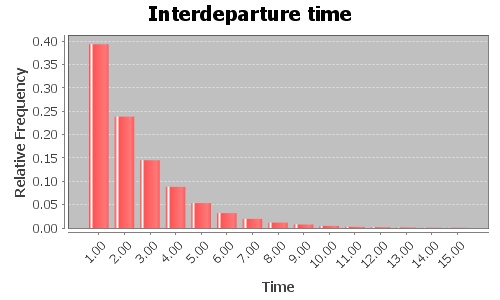
*Figure 4-2: M/M/2 Response time*

**

*Figure 4-3: M/M/2 average number of jobs in the system*

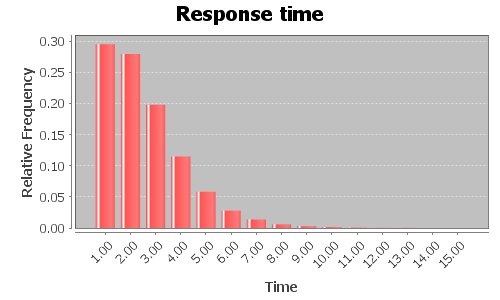


*Figure 4-3: M/M/2 Interarrival time*

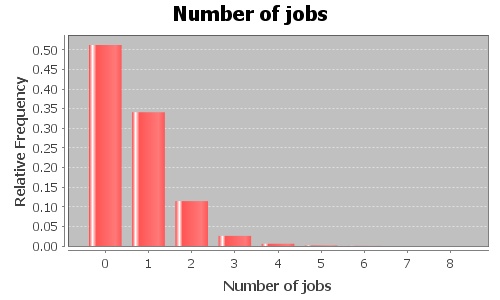
**

*Figure 4-4: M/M/2 Inter departure time*

Figure 4-6 to figure 4-9 show the distribution of response time, customers in the system, interarrival time and inter departure time with parameters m = 3, a = 3, S = 2.

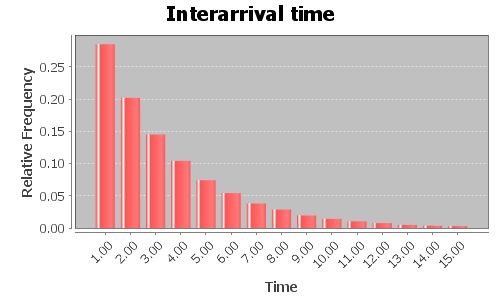


*Figure 4-6: M/M/3 Response time*

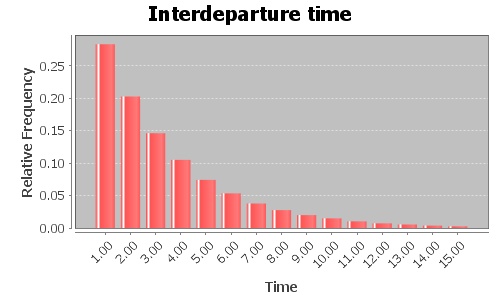
**

*Figure 4-7: M/M/3 Average jobs in the system*

*\*

**

*Figure 4-8: M/M/3 Interarrival time*

**

*Figure 4-9: M/M/3 Inter departure time*

## Analysis of accuracy of simulation models

In the case of k equivalent parallel servers (k > 0) and exponential interarrival and service times (M/M/k model), the formulas for the multiple servers models are listed as follows:

= S\*X;

= S\*X / k = / k;

= S \*

= S \* (1 + ) = S +

Here denotes the time customers spend in queue waiting for service.

Table 4-1 shows the experimental result with different parameters, 1,000,000 jobs were processed in these experiments.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m | a | s | AU | SU | E[%] | AR | SR | E[%] | AQ | SQ | E[%] |
| 4 | 1 | 0.1 | 0.025 | 0.025 | 0.02 | 0.1 | 0.1 | -0.047 | 0.1 | 0.1 | 0.021 |
| 4 | 1 | 0.5 | 0.125 | 0.125 | 0 | 0.5 | 0.501 | 0.112 | 0.5 | 0.5 | 0.001 |
| 4 | 1 | 0.9 | 0.225 | 0.225 | -0.027 | 0.904 | 0.905 | 0.066 | 0.904 | 0.904 | -0.038 |
| 4 | 1 | 1.3 | 0.325 | 0.325 | -0.002 | 1.323 | 1.323 | 0.011 | 1.323 | 1.322 | -0.046 |
| 4 | 1 | 1.7 | 0.425 | 0.425 | -0.051 | 1.78 | 1.782 | 0.1 | 1.78 | 1.779 | -0.063 |
| 4 | 1 | 2.1 | 0.525 | 0.525 | 0.011 | 2.32 | 2.323 | 0.101 | 2.32 | 2.321 | 0.026 |
| 4 | 1 | 2.5 | 0.625 | 0.626 | 0.185 | 3.033 | 3.036 | 0.111 | 3.033 | 3.038 | 0.176 |
| 4 | 1 | 2.9 | 0.725 | 0.725 | 0.002 | 4.134 | 4.155 | 0.483 | 4.134 | 4.151 | 0.39 |
| 4 | 1 | 3.3 | 0.825 | 0.824 | -0.111 | 6.327 | 6.297 | -0.486 | 6.327 | 6.293 | -0.538 |
| 4 | 1 | 3.7 | 0.925 | 0.927 | 0.187 | 14.047 | 14.143 | 0.676 | 14.047 | 14.149 | 0.719 |

*Table 4-1: Multiple servers model experimental results*

*Figure* 4-10 and Figure 4-11 show the comparison between analytical result and simulation result based on the table 4-1.

*Figure 4-10: Multiple servers model: Server utilization*

*Figure 4-11: Multiple servers model: Response time*

From the data in table 5-1 and figure 4-10, 4-11, we can see that after 1,000,000 events, the experimental results are very close to analytical results, with an error rate less than 2%. Therefore, we can conclude that the simulator for multiple servers model is numerically correct.

Next, we would like to verify if the formula is still good for non-exponential model, Table 5-2 shows the experimental results after we change interarrival time distribution and service time distribution to uniform with different parameters. The number of servers m = 2;

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | s | AU | SU | E[%] | AR | SR | E[%] | AQ | SQ | E[%] |
| (0, 2) | (0, 0.5) | 0.125 | 0.125 | 0.118 | 0.251 | 0.254 | 1.267 | 0.251 | 0.254 | 1.343 |
| (0, 2) | (0, 1) | 0.250 | 0.250 | 0.054 | 0.507 | 0.533 | 4.885 | 0.507 | 0.533 | 4.915 |
| (0, 2) | (0, 1.5) | 0.375 | 0.375 | 0.008 | 0.779 | 0.873 | 10.789 | 0.779 | 0.873 | 10.776 |
| (0, 2) | (0, 2) | 0.500 | 0.500 | 0.068 | 1.083 | 1.333 | 18.772 | 1.083 | 1.333 | 18.810 |
| (0, 2) | (0, 2.5) | 0.625 | 0.625 | -0.050 | 1.469 | 2.051 | 28.396 | 1.469 | 2.051 | 28.390 |
| (0, 2) | (0, 3) | 0.750 | 0.750 | -0.750 | 2.068 | 3.429 | 39.682 | 2.068 | 3.429 | 39.653 |
| (0, 2) | (0, 3.5) | 0.876 | 0.875 | -0.085 | 3.577 | 7.467 | 52.088 | 3.578 | 7.467 | 52.080 |

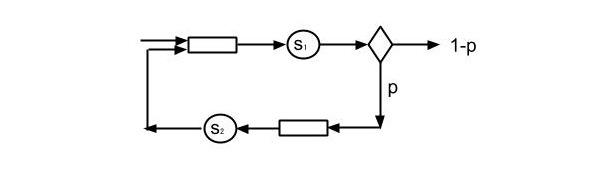
*Table 5-2: Error rate for uniform distribution*

From table 5-2, we can see that when server utilization is less than 0.25, the error is less than 5% which is good. However, when the utilization increases, the error also increases. In the worst case, the error rate is more than 52 percent. So we cannot use exponential formula to the case when interarriva time and server time are uniform distributed.

# Simulator of feedback system

## Feedback queuing model

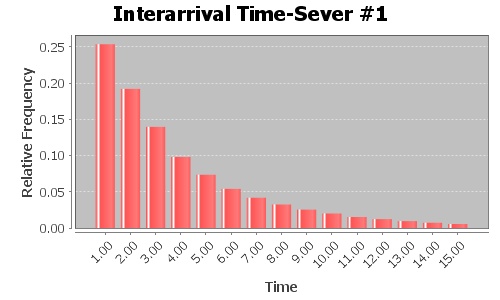
A feedback queuing model (Figure 5-1) consists two servers, a customer comes into the system has percentage p to recycle, and percentage 1-p to leave the system. When p = 0, the system becomes a single server model.



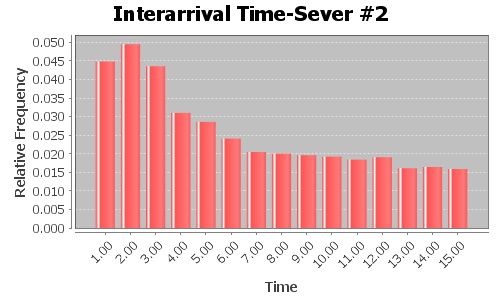
*Figure 5-1: Feedback queuing model*

In feedback queuing model, we only focus on the exponential case: the interarrival time and the service time for two severs are exponential distributed. As the system interarrival time and service time change, the interarrival time distributions for sever one and server two also change and they are not always exponential distribution.

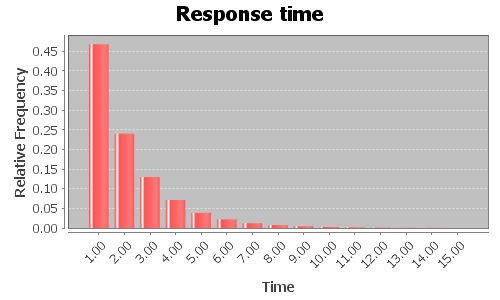
The flowing figure 5-2 to figure 5-5 show the distribution of interarrival time for server one and server two, response time and customers in the system with a = 4 (mean interarrival time), s1 = 1 (mean service time for the server 1), s2 = 1 (mean service time for server 2) and p = 10.

**

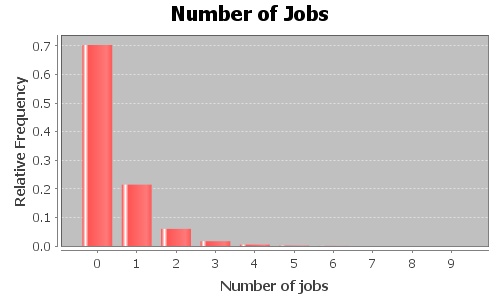
*Figure 5-2 Interarrival time for sever 1*

**

*Figure 5-3 Interarrival time for sever 2*

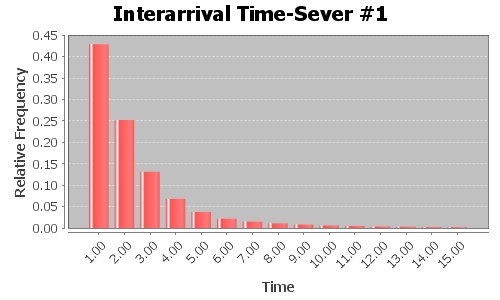
**

*Figure 5-4 Response time*

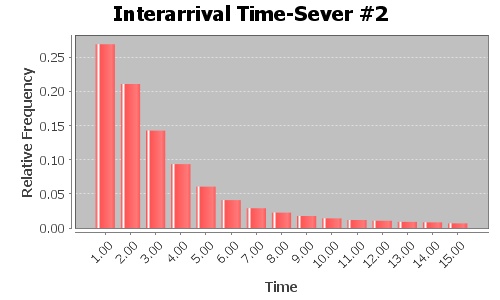
**

*Figure 5-5 The number of jobs in the system.*

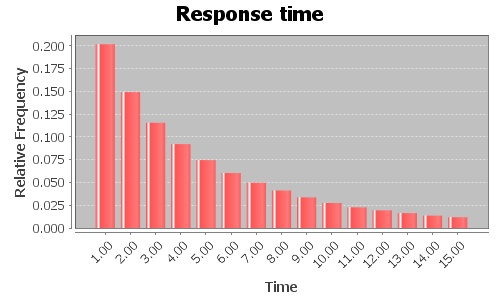
Figure 5-6 to figure 5-9 show distribution of interarrival time for server one and server two, response time and customers in the system with a = 4, s1 = 1, s2 = 1 and p = 50.



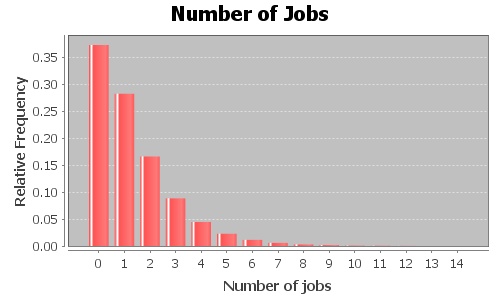
*Figure 5-6: Interarrival time for server 1*

**

*Figure 5-7: Interarrival time for server 2*

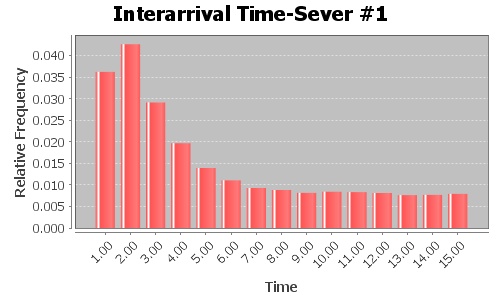
**

*Figure 5-8: Response time*

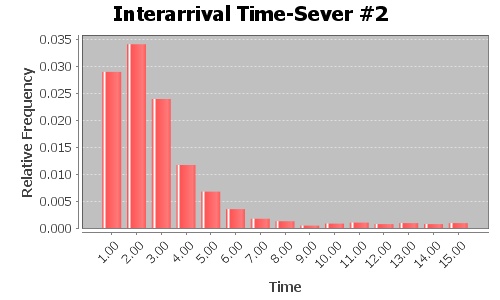
**

*Figure 5-9: Average number of jobs in the system*

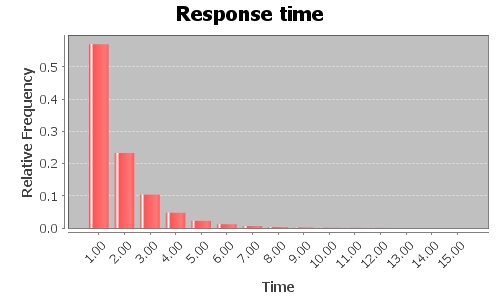
Figure 5-10 to figure 5-13 show distribution of interarrival time for server one and server two, response time and customers in the system with a = 100, s1 = 1, s2 = 1 and p = 10.

**

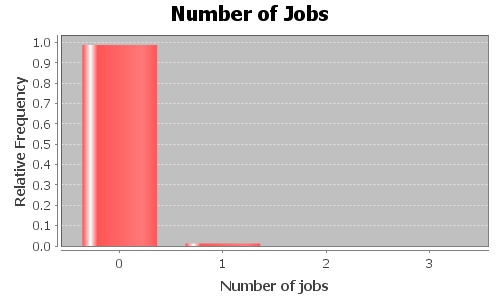
*Figure 5-10: Interarrival time for server 1*

**

*Figure 5-11: Interarrival time for server 2*

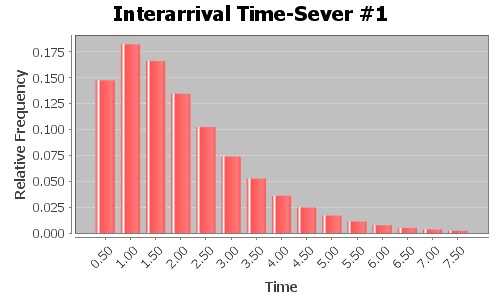


*Figure 5-13: Response time*

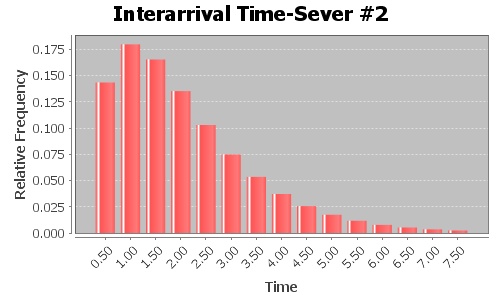
**

*Figure 5-13: Average number of jobs in the system*

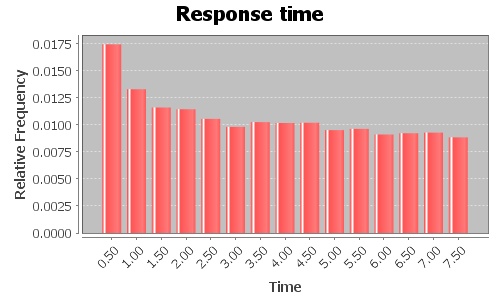
Figure 5-14 to figure 5-17 show distribution of interarrival time for server one and server two, response time and customers in the system with a = 100, s1 = 1, s2 = 1 and p = 95.



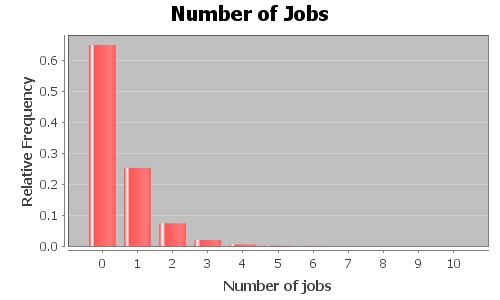
*Figure 5-14: Interarriva time for server 1*



*Figure 5-15 Interarrival time for server 2*



*Figure 5-16: Response time*



*Figure 5-17: Average number of jobs in the system*

From figure 5-10, 5-11, 5-14, 5-15, we can see that non-exponential distributions can be created when the interarrival time and server time are exponential distributed.

## Analysis of accuracy of simulation models

The formulas to verify feedback queuing model are listed as below:

U1 = S1 / (a \* (1 - p))

U2 = p \* S2 / (a \* (1 - p))

Q = S1 / (a \* (1 - p) – S1) + p \* S2 / (a \* (1 - p) – S2);

R = a \* Q;

Table 5.2.1 shows the experimental result with different parameters.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | S1 | S2 | p | AU1 | SU1 | E [%] | AU2 | SU2 | E [%] |
| 4 | 1 | 1 | 0 | 0.25 | 0.249 | -0.247 | 0 | 0 | 0 |
| 4 | 1 | 1 | 0.1 | 0.278 | 0.276 | -0.671 | 0.028 | 0.028 | -0.906 |
| 4 | 1 | 1 | 0.2 | 0.312 | 0.31 | -0.682 | 0.062 | 0.063 | 0.097 |
| 4 | 1 | 1 | 0.3 | 0.357 | 0.358 | 0.318 | 0.107 | 0.108 | 0.506 |
| 4 | 1 | 1 | 0.4 | 0.417 | 0.417 | 0.115 | 0.167 | 0.167 | -0.001 |
| 4 | 1 | 1 | 0.5 | 0.5 | 0.5 | 0.027 | 0.25 | 0.25 | -0.001 |
| 4 | 1 | 1 | 0.6 | 0.625 | 0.63 | 0.788 | 0.375 | 0.38 | 1.37 |
| 4 | 1 | 1 | 0.7 | 0.833 | 0.832 | -0.117 | 0.583 | 0.581 | -0.384 |

*Table 5.2.1 part 1*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | S1 | S2 | p | AR | SR | E [%] | AQ | SQ | E [%] |
| 4 | 1 | 1 | 0 | 0.333 | 0.333 | -0.141 | 1.333 | 1.333 | -0.04 |
| 4 | 1 | 1 | 0.1 | 0.413 | 0.41 | -0.759 | 1.653 | 1.643 | -0.597 |
| 4 | 1 | 1 | 0.2 | 0.521 | 0.516 | -0.909 | 2.085 | 2.076 | -0.41 |
| 4 | 1 | 1 | 0.3 | 0.676 | 0.681 | 0.858 | 2.702 | 2.725 | 0.859 |
| 4 | 1 | 1 | 0.4 | 0.914 | 0.918 | 0.385 | 3.657 | 3.661 | 0.114 |
| 4 | 1 | 1 | 0.5 | 1.333 | 1.337 | 0.262 | 5.333 | 5.338 | 0.08 |
| 4 | 1 | 1 | 0.6 | 2.267 | 2.316 | 2.158 | 9.067 | 9.199 | 1.461 |
| 4 | 1 | 1 | 0.7 | 6.4 | 6.316 | -1.316 | 25.6 | 25.21 | -1.524 |

*Table 5.2.1 part 2*

Figure 5.2.1 to Figure 5.2.3 show the comparison between analytical result and simulation result based on the table 5.2.1.

*Figure 5.2.1 Server #1 utilization*

*Figure 5.2.2 Server #2 utilization*

*Figure 5.2.3 Feedback queuing model response time*

From the data in table 5.2.1 and figure 5.2.1 to figure 5.2.3, we can see that after 100,000 events, the experimental results are very close to analytical results, with an error rate less than 2%. Therefore, we can conclude that the simulator for feedback queuing model is numerically correct.

Next, we would like to verify if the formula is still good for non-exponential model, Table 5-2 and table 5-3 show the experimental results after we change interarrival time distribution and service time distribution to uniform with different parameters.

In table 5-2, p = 50, s1 = 1 (from 0 to 2), s2 = 1 (from 0 to 2), aMin = 0.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| aMax | AU1 | SU1 | E [%] | AU2 | SU2 | E [%] | AR | SR | E [%] | AQ | SQ | E [%] |
| 5 | 0.8 | 0.798 | 0.272 | 0.4 | 0.398 | 0.543 | 11.667 | 7.059 | 39.496 | 4.667 | 2.82 | 39.568 |
| 6 | 0.667 | 0.665 | 0.247 | 0.333 | 0.33 | 0.861 | 7.5 | 5.001 | 33.317 | 2.5 | 1.669 | 33.23 |
| 7 | 0.571 | 0.573 | -0.322 | 0.286 | 0.286 | -0.159 | 6.067 | 4.384 | 27.737 | 1.733 | 1.257 | 27.49 |
| 8 | 0.5 | 0.5 | -0.09 | 0.25 | 0.251 | -0.361 | 5.333 | 4.016 | 24.697 | 1.333 | 1.003 | 24.75 |
| 9 | 0.444 | 0.444 | 0.136 | 0.222 | 0.222 | 0.171 | 4.886 | 3.794 | 22.352 | 1.086 | 0.843 | 22.386 |

*Table 5-2: Experimental result for different aMax*

In table 5-3, a = 50 (from 0 to 100), s1 = 1 (from 0 to 2), s2 = 1 (from 0 to 2).

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| p | AU1 | SU1 | E [%] | AU2 | SU2 | E [%] | AR | SR | E [%] | AQ | SQ | E [%] |
| 10 | 0.022 | 0.022 | -0.251 | 0.002 | 0.002 | -0.24 | 1.248 | 1.231 | 1.307 | 0.025 | 0.025 | 1.029 |
| 20 | 0.025 | 0.025 | -0.46 | 0.005 | 0.005 | -0.582 | 1.533 | 1.517 | 1.042 | 0.031 | 0.03 | 0.933 |
| 30 | 0.029 | 0.029 | 0.034 | 0.009 | 0.009 | -0.188 | 1.903 | 1.869 | 1.781 | 0.038 | 0.037 | 1.597 |
| 40 | 0.033 | 0.033 | -0.17 | 0.013 | 0.013 | 0.042 | 2.4 | 2.353 | 1.934 | 0.048 | 0.047 | 1.689 |
| 50 | 0.04 | 0.04 | 0.353 | 0.02 | 0.02 | 0.8 | 3.104 | 3.019 | 2.727 | 0.062 | 0.06 | 2.677 |
| 60 | 0.05 | 0.05 | 0.322 | 0.03 | 0.03 | 0.496 | 4.178 | 4.055 | 2.937 | 0.084 | 0.081 | 3.137 |
| 70 | 0.067 | 0.067 | 0.162 | 0.047 | 0.047 | 0.351 | 6.019 | 5.761 | 4.279 | 0.12 | 0.115 | 4.055 |
| 80 | 0.1 | 0.1 | -0.052 | 0.08 | 0.08 | -0.061 | 9.903 | 9.339 | 5.701 | 0.198 | 0.187 | 5.839 |
| 90 | 0.2 | 0.2 | 0.248 | 0.18 | 0.18 | 0.268 | 23.476 | 20.667 | 11.966 | 0.47 | 0.413 | 12.117 |

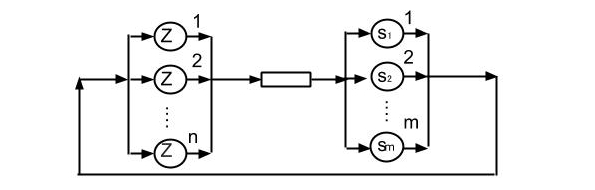
*Table 5-3: Experimental results for different p*

From table 5-2, we can see that in the worst case, the error reaches 40 percent, so the formula cannot be used as predictor as non-exponential models. However, from table 5-3, we can see that when mean interarrival time is much higher than mean service time and p less than 80, the error is very small. So in this case, the formula can be used as an approximate predictor for non-exponential models.

# Simulator of interactive system

## Interactive System

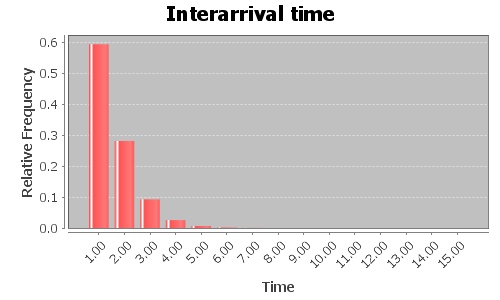
An interactive system (Figure 6-1) consists several workstations and several servers, each workstations has one customer. Each customer will go back to a workstation after being served.



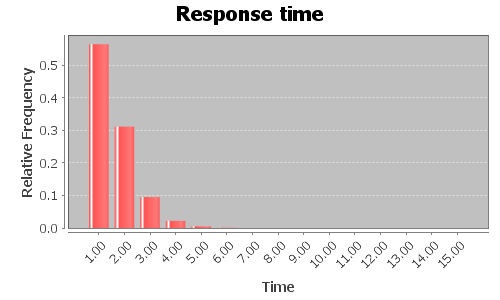
*Figure 6-1 Interactive system*

In Interactive system, we also focus on the exponential case: the think time of the workstations and the service time for two severs are exponential distributed.

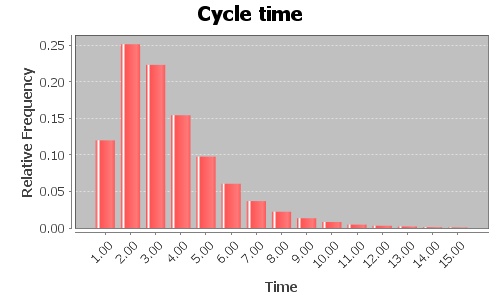
The flowing figure 6-2 to figure 6-6 show the distribution of interarrival time, response time, cycle time and customers in the system with mean think time t = 4, number of workstations n = 3, number of servers m = 2, mean think time t = 2, mean service time s = 1.



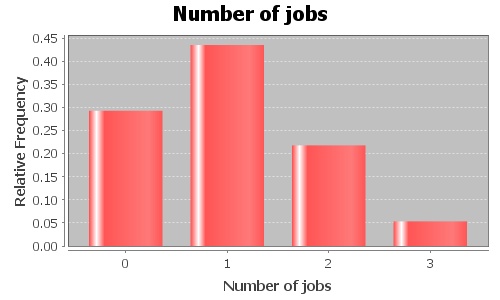
*Figure 6-2: Interarrival time in the queue*



*Figure 6-3: Response time*



*Figure 6-4: Cycle time*



*Figure 6-5: Number of jobs*

## Analysis of accuracy of simulator models

The formulas to verify interactive system model are listed as below:

n = number of work stations;

m = number of servers;

Z = average think time for work station;

S = average service time;

R = average response time = – Z = S, n = 1

R = average response time = – Z = , n >>1

Table 6-1 shows the experimental results with different parameters.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Z | S | n | m | AU | SU | E [%] | AR | SR | E [%] |
| 5 | 1 | 5 | 1 | 0.715 | 0.713 | -0.359 | 6.992 | 7.004 | 0.178 |
| 5 | 2 | 5 | 1 | 0.93 | 0.931 | 0.072 | 10.75 | 10.755 | 0.051 |
| 5 | 3 | 5 | 1 | 0.98 | 0.979 | -0.039 | 15.312 | 15.292 | -0.13 |
| 5 | 4 | 5 | 1 | 0.993 | 0.993 | 0.038 | 20.147 | 20.148 | 0.002 |
| 5 | 5 | 5 | 1 | 0.997 | 0.997 | 0.008 | 25.077 | 25.146 | 0.276 |
| 5 | 6 | 5 | 1 | 0.999 | 0.999 | 0.006 | 30.044 | 30.158 | 0.38 |
| 5 | 7 | 5 | 1 | 0.999 | 0.999 | -0.003 | 35.027 | 35.061 | 0.097 |
| 5 | 8 | 5 | 1 | 1 | 1 | 0.005 | 40.017 | 40.143 | 0.315 |
| 5 | 9 | 5 | 1 | 1 | 1 | 0.003 | 45.011 | 45.05 | 0.086 |

*Table 6-1 Part 1*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Z | S | n | m | AC | SC | E [%] | AQ | SQ | E[%] |
| 5 | 1 | 5 | 1 | 1.424 | 1.413 | -0.827 | 1.992 | 1.979 | -0.647 |
| 5 | 2 | 5 | 1 | 2.674 | 2.681 | 0.262 | 5.75 | 5.768 | 0.314 |
| 5 | 3 | 5 | 1 | 3.367 | 3.371 | 0.103 | 10.312 | 10.309 | -0.027 |
| 5 | 4 | 5 | 1 | 3.759 | 3.762 | 0.082 | 15.147 | 15.16 | 0.084 |
| 5 | 5 | 5 | 1 | 4.003 | 4.004 | 0.025 | 20.077 | 20.137 | 0.301 |
| 5 | 6 | 5 | 1 | 4.168 | 4.166 | -0.035 | 25.044 | 25.13 | 0.345 |
| 5 | 7 | 5 | 1 | 4.286 | 4.283 | -0.07 | 30.027 | 30.035 | 0.028 |
| 5 | 8 | 5 | 1 | 4.375 | 4.376 | 0.021 | 35.017 | 35.135 | 0.336 |
| 5 | 9 | 5 | 1 | 4.445 | 4.444 | -0.008 | 40.011 | 40.042 | 0.078 |

*Table 6-1 Part 2*

Figure 6-1 to Figure 6-3 show the comparison between analytical result and simulation result based on the table 6-1.

*Figure 6-1: Server utilization*

*Figure 6-2: Cycle time*

*Figure 6-3: Response Time*

From the data in table 6-1 and figure 6-1 to figure 6-3, we can see that after 1,000,000 events, the experimental results are very close to analytical results, with an error rate less than 2%. Therefore, we can conclude that the simulator for feedback queuing model is numerically correct.

Next, we would like to verify if the formula is still good for non-exponential model, Table 6-2 shows the experimental results after we change interarrival time distribution and service time distribution to uniform with different parameters.

In table 6-2, n =3, m = 1, aMin = 0, aMax = 3, sMin = 0;

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| sMin | AU | SU | E [%] | AR | SR | E [%] | AC | SC | E [%] | AQ | SQ | E[%] |
| 1 | 0.654 | 0.677 | -3.27 | 0.794 | 0.721 | 9.231 | 2.294 | 2.22 | 3.214 | 1.038 | 0.974 | 6.197 |
| 2 | 0.866 | 0.903 | -4.306 | 1.966 | 1.822 | 7.326 | 3.466 | 3.32 | 4.199 | 1.701 | 1.646 | 3.326 |
| 3 | 0.938 | 0.965 | -2.954 | 3.3 | 3.159 | 4.279 | 4.8 | 4.66 | 2.92 | 2.062 | 2.034 | 1.4 |
| 4 | 0.967 | 0.984 | -1.849 | 4.708 | 4.597 | 2.358 | 6.208 | 6.096 | 1.796 | 2.275 | 2.262 | 0.572 |
| 5 | 0.98 | 0.992 | -1.171 | 6.152 | 6.063 | 1.449 | 7.652 | 7.563 | 1.162 | 2.412 | 2.405 | 0.29 |
| 6 | 0.987 | 0.995 | -0.792 | 7.615 | 7.549 | 0.874 | 9.115 | 9.049 | 0.726 | 2.506 | 2.503 | 0.149 |
| 7 | 0.991 | 0.997 | -0.552 | 9.091 | 9.04 | 0.558 | 10.591 | 10.539 | 0.488 | 2.575 | 2.573 | 0.071 |

*Table 6-2: Experimental result for different sMin*

From table 6-2, we can see the worst case for error is 9.231%, when other parameters are fixed, as the mean service time increases, the error decreases. So in interactive system model, the exponential formula can be used as a predictor for the case when think time and service time are uniform distributed.

# Conclusions and future work

## Conclusions

A Java program for simulator of queuing models has been presented. This project is based on the discrete event driven model with GUI features of Java. The project provides four web based simulators of queuing models. With these simulators, students can study the accuracy of queuing models. So these simulators can also be used as educational tool for computer science students.

Experiments were run with different parameters for each 4 models. Measurements from the experiments have been verified to be reasonably accurate. Because this project provides numerical result and various distribution charts, it can help researchers to design and analyze their research projects involving queuing models, and help them to understand the system.

In short, this project has achieved the goals of providing numerically accurate tool for single server model, multiple server model, feedback model and interactive system. Moreover, as the user increases the number of served jobs, the simulator will provide more accurate result. Also, the histogram for feedback model proves that non-exponential flows can be created even in the case of Poisson arrivals process.

## Future work

Based on the problems and challenges in developing four simulators in this project, the future work should be including following two aspects:

First of all, make the code for simulator more extendable. Current simulators are not very extendable; it is very hard to modify the code to design a new simulator. Future work can include more extendable code and interface, so that when a new simulator is required, it will be more efficient to design a new simulator.

Also, it should be necessary to provide simulators for completed models. Current simulators only include 4 simple models: single server model, multiple servers model, feedback queuing model with single server and interactive system. So if more completed models can be added into this 4 simulators, it will be very helpful for the users.

**References**

[1] Beizer Boris. “Micro-analysis of computer system performance”, 1934, pp 207-236

[2] Jozo Dojmovic. “Course Reader for CSC841”, 1998, pp 341-345

Appendix: Source Code

/\*\*

\* Calculator.java - Calculator for analytical models

\* **@author** Xiao Ping

\*/

package gg1simulator**;**

public abstract class Calculator **{**

public abstract double getArrivalMean**();**

public abstract double getServiceMean**();**

public abstract double getUtilization**();**

public abstract double getResponseTime**();**

public abstract double getQueueLength**();**

**}**

/\*\*

\* GG1Calculator.java - Calculator for G/G/1 Model

\* **@author** Xiao Ping

\*/

package gg1simulator**;**

public class GG1Calculator **extends** Calculator **{**

double aMin**,** aMax**,** sMin**,** sMax**;**

public GG1Calculator**(**double aMin**,** double aMax**,** double sMin**,** double sMax**)** **{**

**this.**aMin **=** aMin**;**

**this.**aMax **=** aMax**;**

**this.**sMin **=** sMin**;**

**this.**sMax **=** sMax**;**

**}**

@Override

public double getUtilization**()** **{**

double meanArrivalTime **=** **(**aMin **+** aMax**)/**2**;**

double meanServiceTime **=** **(**sMin **+** sMax**)/**2**;**

**return** meanServiceTime**/**meanArrivalTime**;**

**}**

@Override

public double getResponseTime**()** **{**

double S **=** **(**sMin **+** sMax**)/**2**;**

double va **=** **(**aMax **-** aMin**)** **/** **((**aMax **+** aMin**)** **\*** Math**.**sqrt**(**3.**));**

double vs **=** **(**sMax **-** sMin**)** **/** **((**sMax **+** sMin**)** **\*** Math**.**sqrt**(**3.**));**

double U **=** getUtilization**();**

double R **=** **(**S **/** **(**1. **-** U**))** **\*** **(**1 **-** 0.5 **\*** U **\*** **(**1. **-** vs **\*** vs **-** **((**vs **\*** vs **+** 1.**)** **\*** **(**va **\*** va **-** 1.**)** **/** **(**U **\*** U **\*** vs **\*** vs **+** 1**))));**

**return** R**;**

**}**

@Override

public double getQueueLength**()** **{**

**return** 0**;**

**}**

@Override

public double getArrivalMean**()** **{**

**return** **(**aMin **+** aMax**)/**2**;**

**}**

@Override

public double getServiceMean**()** **{**

**return** **(**sMin **+** sMax**)/**2**;**

**}**

**}**

**/\***

**\* MM1Calculator.java - Calculator for M/M/1 Model**

**\*/**

package gg1simulator**;**

public class MM1Calculator **extends** Calculator **{**

double meanArrivalTime**,** meanServiceTime**;**

public MM1Calculator**(**double meanArrivalTime**,** double meanServiceTime**)** **{**

**this.**meanArrivalTime **=** meanArrivalTime**;**

**this.**meanServiceTime **=** meanServiceTime**;**

**}**

@Override

public double getUtilization**()** **{**

**return** meanServiceTime **/** meanArrivalTime**;**

**}**

@Override

public double getResponseTime**()** **{**

**return** meanServiceTime **/** **(**1 **-** getUtilization**());**

**}**

@Override

public double getQueueLength**()** **{**

double u **=** getUtilization**();**

**return** u **/** **(**1 **-** u**);**

**}**

@Override

public double getArrivalMean**()** **{**

**return** meanArrivalTime**;**

**}**

@Override

public double getServiceMean**()** **{**

**return** meanServiceTime**;**

**}**

**}**

/\*

\* MD1Calculator.java - Calculator for M/D/1 Model

\*/

package gg1simulator**;**

public class MD1Calculator **extends** Calculator **{**

double meanArrivalTime**,** meanServiceTime**;**

public MD1Calculator**(**double meanArrivalTime**,** double meanServiceTime**)** **{**

**this.**meanArrivalTime **=** meanArrivalTime**;**

**this.**meanServiceTime **=** meanServiceTime**;**

**}**

@Override

public double getUtilization**()** **{**

**return** meanServiceTime **/** meanArrivalTime**;**

**}**

@Override

public double getResponseTime**()** **{**

**return** meanServiceTime **\*** **(**2 **-** getUtilization**())** **/** **(**2 **\*** **(**1 **-** getUtilization**()));**

**}**

@Override

public double getQueueLength**()** **{**

double u **=** getUtilization**();**

**return** u **\*** u **/** **(**2 **\*** **(**1 **-** u**))** **+** u**;**

**}**

@Override

public double getArrivalMean**()** **{**

**return** meanArrivalTime**;**

**}**

@Override

public double getServiceMean**()** **{**

**return** meanServiceTime**;**

**}**

**}**

/\*

\* MG1Calculator.java - Calculator for M/G/1 model

\*/

package gg1simulator**;**

public class MG1Calculator **extends** Calculator **{**

double a**,** sMin**,** sMax**;**

public double getMin**(){**

**return** sMin**;**

**}**

public double getMax**(){**

**return** sMax**;**

**}**

public MG1Calculator**(**double a**,** double sMin**,** double sMax**)** **{**

**this.**a **=** a**;**

**this.**sMin **=** sMin**;**

**this.**sMax **=** sMax**;**

**}**

@Override

public double getUtilization**()** **{**

double s **=** **(**sMin **+** sMax**)** **/** 2**;**

**return** s **/** a**;**

**}**

@Override

public double getResponseTime**()** **{**

double S **=** **(**sMin **+** sMax**)** **/** 2**;**

double vs **=** **(**sMax **-** sMin**)** **/** **((**sMax **+** sMin**)** **\*** Math**.**sqrt**(**3.**));**

double U **=** getUtilization**();**

double R **=** **(**S **/** **(**1. **-** U**))** **\*** **(**1 **-** 0.5 **\*** U **\*** **(**1. **-** vs **\*** vs**));**

**return** R**;**

**}**

@Override

public double getQueueLength**()** **{**

double vs **=** **(**sMax **-** sMin**)** **/** **((**sMax **+** sMin**)** **\*** Math**.**sqrt**(**3.**));**

double U **=** getUtilization**();**

double aRate **=** 1.0 **/** a**;**

**return** U **+** **(**U **\*** U **+** aRate **\*** aRate **\*** vs**)** **/** **(**2 **\*** **(**1 **-** U**));**

**}**

@Override

public double getArrivalMean**()** **{**

**return** a**;**

**}**

@Override

public double getServiceMean**()** **{**

**return** **(**sMin **+** sMax**)** **/** 2**;**

**}**

**}**

/\*

\* MultipleServersCalculator.java - Calculator for multiple servers model

\*/

package gg1simulator**;**

public class MultipleServersCalculator **extends** Calculator **{**

private int numOfServers**;**

private double avgArrivalTime**,** avgServiceTime**;**

private double waitTime**,** responseTime**,** jobs**;**

public MultipleServersCalculator**(**int numOfServers**,** double avgArrivalTime**,** double avgServiceTime**){**

**this.**numOfServers **=** numOfServers**;**

**this.**avgArrivalTime **=** avgArrivalTime**;**

**this.**avgServiceTime **=** avgServiceTime**;**

**}**

@Override

public double getArrivalMean**()** **{**

**return** avgArrivalTime**;**

**}**

@Override

public double getServiceMean**()** **{**

**return** avgServiceTime**;**

**}**

@Override

public double getUtilization**()** **{**

**return** avgServiceTime **/(**numOfServers **\*** avgArrivalTime**);**

**}**

public void Calculate**(){**

double u **=** avgServiceTime **/** avgArrivalTime**;**

double sum **=** 1**;**

**for(**int i **=** 2**;** i **<=** numOfServers**;** i**++){**

sum **+=** Math**.**pow**(**u**,** i **-** 1**)** **/** factorial**(**i **-** 1**);**

**}**

sum **+=** Math**.**pow**(**u**,** numOfServers**)** **/** **(**factorial**(**numOfServers**)** **\*** **(**1 **-** u **/** numOfServers**));**

sum **\*=** factorial**(**numOfServers**);**

sum **\*=** numOfServers**;**

sum **\*=** Math**.**pow**(**1 **-** u **/** numOfServers**,** 2**);**

waitTime **=** avgServiceTime **\*** Math**.**pow**(**u**,** numOfServers**)** **/** sum**;**

responseTime **=** avgServiceTime **\*** **(**1 **+** waitTime**);**

jobs **=** responseTime **/** avgArrivalTime**;**

**}**

public static int factorial**(**int x**)** **{**

**if** **(**x **<** 0**)** **{**

**throw** **new** IllegalArgumentException**(**"x must be>=0"**);**

**}**

int fact **=** 1**;**

**for** **(**int i **=** 2**;** i **<=** x**;** i**++)** **{**

fact **\*=** i**;**

**}**

**return** fact**;**

**}**

public double getWaitTime**(){**

**return** waitTime**;**

**}**

@Override

public double getResponseTime**()** **{**

**return** responseTime**;**

**}**

@Override

public double getQueueLength**()** **{**

**return** jobs**;**

**}**

public static void main**(**String args**[]){**

MultipleServersCalculator m **=** **new** MultipleServersCalculator**(**4**,** 2**,** 1**);**

m**.**Calculate**();**

System**.**out**.**println**(**m**.**getUtilization**());**

System**.**out**.**println**(**m**.**responseTime**);**

System**.**out**.**println**(**m**.**jobs**);**

**}**

**}**

/\*

\* FeedbackQueueCalculator.java - Calculator for feedback queuing model

\*/

package gg1simulator**;**

public class FeedbackQueueCalculator **{**

double arrivalTime**;**

double firstServiceTime**;**

double secondServiceTime**;**

double p**;**

public double getMeanArrivalTime**(){**

**return** arrivalTime**;**

**}**

public double getMeanFirstServiceTime**(){**

**return** firstServiceTime**;**

**}**

public double getMeanSecondServiceTime**(){**

**return** secondServiceTime**;**

**}**

public FeedbackQueueCalculator**(**double arrivalTime**,** double firstServiceTime**,** double secondServiceTime**,** double p**)** **{**

**this.**arrivalTime **=** arrivalTime**;**

**this.**firstServiceTime **=** firstServiceTime**;**

**this.**secondServiceTime **=** secondServiceTime**;**

**this.**p **=** p**;**

**}**

public double getFirstUtilization**()** **{**

**return** firstServiceTime **/** **(**arrivalTime **\*** **(**1 **-** p**));**

**}**

public double getSecondUtilization**()** **{**

**return** p **\*** secondServiceTime **/** **(**arrivalTime **\*** **(**1 **-** p**));**

**}**

public double getJobsInTheSystem**()** **{**

double j1 **=** firstServiceTime **/** **(**arrivalTime **\*** **(**1 **-** p**)** **-** firstServiceTime**);**

double j2 **=** p **\*** secondServiceTime **/** **(**arrivalTime **\*** **(**1 **-** p**)** **-** p **\*** secondServiceTime**);**

**return** j1 **+** j2**;**

**}**

public double getResponseTime**()** **{**

double j1 **=** firstServiceTime **/** **(**arrivalTime **\*** **(**1 **-** p**)** **-** firstServiceTime**);**

double j2 **=** p **\*** secondServiceTime **/** **(**arrivalTime **\*** **(**1 **-** p**)** **-** p **\*** secondServiceTime**);**

**return** **(**j1 **+** j2**)** **\*** arrivalTime**;**

**}**

**}**

/\*

\* InteractiveSystemCalculator.java - Calculator for interactive systems

\*/

package gg1simulator**;**

public class InteractiveSystemCalculator **{**

private int n**,** m**;**

private double t**,** s**;**

public double u**,** r**,** x**,** l**;**

public InteractiveSystemCalculator**(**double t**,** double s**,** int n**,** int m**)** **{**

**this.**t **=** t**;**

**this.**s **=** s**;**

**this.**n **=** n**;**

**this.**m **=** m**;**

**}**

public void Calculate**()** **{**

double**[]** f **=** **new** double**[**500**],** p **=** **new** double**[**500**];**

//critn = nproc \* (serve + think) / serve;

double rho **=** s **/** t**;**

f**[**0**]** **=** 1.0**;**

double sum **=** 1.0**;**

int j **=** 1**;**

**while** **(**j **<=** n**)** **{**

f**[**j**]** **=** f**[**j **-** 1**]** **\*** **(**n **+** 1. **-** j**)** **\*** rho **/** Math**.**min**(**m**,** j**);**

sum **+=** f**[**j**];**

j**++;**

**}**

j **=** 0**;**

**while** **(**j **<=** m**)** **{**

p**[**j**]** **=** f**[**j**]** **/** sum**;**

j**++;**

**}**

double pidle **=** 0.**;**

j **=** 0**;**

**while** **(**j **<** n**)** **{**

pidle **+=** p**[**j**]** **\*** **(**float**)** **(**m **-** j**)** **/** **(**float**)** **(**m**);**

j**++;**

**}**

u **=** 1. **-** pidle**;**

r **=** n **\*** s **/** **(**u **\*** m**)** **-** t**;**

x **=** n **/** **(**t **+** r**);**

l **=** r **\*** x**;**

**}**

public double getUtilization**()** **{**

**return** u**;**

**}**

public double getResponseTime**()** **{**

**return** r**;**

**}**

public double getThroughput**()** **{**

**return** x**;**

**}**

public double getCycleTime**()** **{**

**return** t **+** r**;**

**}**

public double getQueueLength**()** **{**

**return** l**;**

**}**

**}**

/\*

\* Distribution.java

\*/

package gg1simulator**;**

public abstract class Distribution **{**

public abstract double generateRandomTimes**();**

public abstract double getMean**();**

**}**

package gg1simulator**;**

/\*

\* ExponentialDistribution.java

\*/

**import** java**.**util**.\*;**

public class ExponentialDistribution **extends** Distribution **{**

double lambda**;**

Random rand **=** **new** Random**();**

public ExponentialDistribution**(**double lambda**)** **{**

**this.**lambda **=** lambda**;**

**}**

@Override

public double generateRandomTimes**()** **{**

**return** **-** lambda **\*** Math**.**log**(**rand**.**nextDouble**());**

**}**

@Override

public double getMean**()** **{**

**return** lambda**;**

**}**

**}**

/\*

\* UniformDistribution.java

\*/

package gg1simulator**;**

**import** java**.**util**.**Random**;**

public class UniformDistribution **extends** Distribution **{**

double a**,** b**;**

Random generator **=** **new** Random**();**

public UniformDistribution**(**double a**,** double b**)** **{**

**this.**a **=** Math**.**min**(**a**,** b**);**

**this.**b **=** Math**.**max**(**a**,** b**);**

**}**

@Override

public double generateRandomTimes**()** **{**

**return** a **+** generator**.**nextDouble**()** **\*** **(**b **-** a**);**

**}**

@Override

public double getMean**()** **{**

**return** **(**a **+** b**)** **/**2**;**

**}**

public double getMin**(){**

**return** Math**.**min**(**a**,** b**);**

**}**

public double getMax**(){**

**return** Math**.**max**(**a**,** b**);**

**}**

**}**

/\*

\* DeterministicDistribution.java

\*/

package gg1simulator**;**

public class DeterministicDistribution **extends** Distribution **{**

private double constrant**;**

public DeterministicDistribution**(**double constrant**){**

**this.**constrant **=** constrant**;**

**}**

@Override

public double generateRandomTimes**()** **{**

**return** constrant**;**

**}**

@Override

public double getMean**()** **{**

**return** constrant**;**

**}**

**}**

/\*\*

\* Event.java

\*

\*/

package gg1simulator**;**

public class Event **implements** Comparable**<**Event**>** **{**

public static final int ARRIVAL **=** 0**;**

public static final int DEPARTURE **=** 1**;**

public static final int MIDARRIVAL **=** 2**;**

public static final int MIDDEPARTURE **=** 3**;**

public static final int DEPARTURE2 **=** 4**;**

public static final int FIRSTBRANCHARRIVAL **=** 5**;**

public static final int SECONDBRANCHARRIVAL **=** 6**;**

protected double startsTime**;**

protected double endsTime**;** // next arrival or departure time

protected int code**;**

protected int branch**;**

protected int id**;**

public Event**(**double startsTime**,** double endsTime**,** int code**,** int branch**)** **{**

**this.**startsTime **=** startsTime**;**

**this.**endsTime **=** endsTime**;**

**this.**code **=** code**;**

**this.**branch **=** branch**;**

**}**

public Event**(**int id**,** double startsTime**,** double endsTime**,** int code**,** int branch**)** **{**

**this.**id **=** id**;**

**this.**startsTime **=** startsTime**;**

**this.**endsTime **=** endsTime**;**

**this.**code **=** code**;**

**this.**branch **=** branch**;**

**}**

public double getStartsTime**()** **{**

**return** startsTime**;**

**}**

public double getEndsTime**()** **{**

**return** endsTime**;**

**}**

public int getCode**()** **{**

**return** code**;**

**}**

public int getBranch**()** **{**

**return** branch**;**

**}**

@Override

public int compareTo**(**Event e**)** **{**

**return** Double**.**compare**(**endsTime**,** e**.**endsTime**);**

**}**

**}**

/\*

\* Jobs.java

\*/

package gg1simulator**;**

public class Jobs **implements** Comparable**<**Jobs**>** **{**

private int jobId**;**

private double startTime**;**

private double arrivalTime**;**

private double lastArrivalTime**;**

private double startServiceTime**;**

private double midArrivalTime**;**

private double midDepartureTime**;**

private double departureTime**;**

public Jobs**(**int jobId**,** double arrivalTime**,** double lastArrivalTime**)** **{**

**this.**jobId **=** jobId**;**

**this.**arrivalTime **=** arrivalTime**;**

**this.**lastArrivalTime **=** lastArrivalTime**;**

**}**

public Jobs**(**int jobId**,** double startTime**,** double lastArrivalTime**,** double departureTime**){**

**this.**jobId **=** jobId**;**

**this.**startTime **=** startTime**;**

**this.**lastArrivalTime **=** lastArrivalTime**;**

**this.**departureTime **=** departureTime**;**

**}**

public int getJobId**()** **{**

**return** jobId**;**

**}**

public void setJobId**(**int jobId**)** **{**

**this.**jobId **=** jobId**;**

**}**

public double getStartTime**(){**

**return** startTime**;**

**}**

public double getArrivalTime**()** **{**

**return** arrivalTime**;**

**}**

public double getMidArrivalTime**()** **{**

**return** midArrivalTime**;**

**}**

public double getLastArrivalTime**()** **{**

**return** lastArrivalTime**;**

**}**

public void setLastArrivaltime**(**double lastArrivalTime**)** **{**

**this.**lastArrivalTime **=** lastArrivalTime**;**

**}**

public void setMidArrivalTime**(**double midArrivalTime**)** **{**

**this.**midArrivalTime **=** midArrivalTime**;**

**}**

public double getMidDepartureTime**()** **{**

**return** midDepartureTime**;**

**}**

public void setMidDepartureTime**(**double midDepartureTime**)** **{**

**this.**midDepartureTime **=** midDepartureTime**;**

**}**

public double getDepartureTime**()** **{**

**return** departureTime**;**

**}**

public void setDepartureTime**(**double departureTime**)** **{**

**this.**departureTime **=** departureTime**;**

**}**

public double getStartServiceTime**()** **{**

**return** startServiceTime**;**

**}**

public void setStartServiceTime**(**double startServiceTime**)** **{**

**this.**startServiceTime **=** startServiceTime**;**

**}**

@Override

public int compareTo**(**Jobs jobs**)** **{**

**return** Double**.**compare**(**lastArrivalTime**,** jobs**.**lastArrivalTime**);**

**}**

**}**

/\*

\* PlotGenerator.java

\*/

package gg1simulator**;**

**import** java**.**awt**.**BasicStroke**;**

**import** java**.**awt**.**Color**;**

**import** java**.**io**.**FileOutputStream**;**

**import** java**.**io**.**OutputStream**;**

**import** java**.**text**.**DecimalFormat**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Collections**;**

**import** java**.**util**.**HashMap**;**

**import** java**.**util**.**Map**.**Entry**;**

**import** org**.**jfree**.**chart**.**ChartFactory**;**

**import** org**.**jfree**.**chart**.**ChartUtilities**;**

**import** org**.**jfree**.**chart**.**JFreeChart**;**

**import** org**.**jfree**.**chart**.**axis**.**CategoryAxis**;**

**import** org**.**jfree**.**chart**.**axis**.**CategoryLabelPositions**;**

**import** org**.**jfree**.**chart**.**axis**.**NumberAxis**;**

**import** org**.**jfree**.**chart**.**axis**.**NumberTickUnit**;**

**import** org**.**jfree**.**chart**.**plot**.**CategoryPlot**;**

**import** org**.**jfree**.**chart**.**plot**.**PlotOrientation**;**

**import** org**.**jfree**.**chart**.**plot**.**XYPlot**;**

**import** org**.**jfree**.**chart**.**renderer**.**category**.**BarRenderer**;**

**import** org**.**jfree**.**chart**.**renderer**.**xy**.**XYSplineRenderer**;**

**import** org**.**jfree**.**data**.**category**.**DefaultCategoryDataset**;**

**import** org**.**jfree**.**data**.**statistics**.**HistogramDataset**;**

**import** org**.**jfree**.**data**.**statistics**.**HistogramType**;**

**import** org**.**jfree**.**data**.**xy**.**XYSeries**;**

**import** org**.**jfree**.**data**.**xy**.**XYSeriesCollection**;**

/\*\*

\*

\* **@author** Administrator

\*/

public class PlotGenerator **{**

public static JFreeChart generateHistogram**(**ArrayList**<**Double**>** list**,** String name**)** **{**

HistogramDataset dataset **=** **new** HistogramDataset**();**

dataset**.**setType**(**HistogramType**.**RELATIVE\_FREQUENCY**);**

double**[]** values **=** **new** double**[**list**.**size**()];**

**for** **(**int i **=** 0**;** i **<** values**.**length**;** i**++)** **{**

values**[**i**]** **=** list**.**get**(**i**);**

**}**

dataset**.**addSeries**(**"name"**,** values**,** 20**);**

JFreeChart chart **=** ChartFactory**.**createHistogram**(**name**,** "Time (s)"**,** "Relative Frequency"**,** dataset**,** PlotOrientation**.**VERTICAL**,** **false,** **false,** **false);**

**return** chart**;**

**}**

public static JFreeChart generateHistogram**(**HashMap**<**Integer**,** Double**>** map**,** String name**,** int bar**,** int type**)** **{**

DefaultCategoryDataset d **=** **new** DefaultCategoryDataset**();**

double max **=** **-**1**;**

**for** **(**Entry**<**Integer**,** Double**>** e **:** map**.**entrySet**())** **{**

max **=** Math**.**max**(**max**,** e**.**getKey**());**

**}**

double**[]** values **=** **new** double**[(**int**)** max **+** 1**];**

**for** **(**Entry**<**Integer**,** Double**>** e **:** map**.**entrySet**())** **{**

values**[**e**.**getKey**()]** **+=** e**.**getValue**();**

**}**

int length**;**

**if** **(**values**.**length **<** bar**)** **{**

length **=** values**.**length**;**

**}** **else** **{**

length **=** bar**;**

**}**

double sum **=** Utility**.**getSum**(**map**);**

**for** **(**int i **=** 0**;** i **<** length**;** i**++)** **{**

d**.**setValue**(**values**[**i**]/**sum**,** "Relative Frequency"**,** "" **+** i**);**

**}**

JFreeChart chart **=** ChartFactory**.**createBarChart**(**name**,** "Number of jobs"**,** "Relative Frequency"**,** d**,** PlotOrientation**.**VERTICAL**,** **false,** **false,** **false);**

CategoryPlot plot **=** **(**CategoryPlot**)** chart**.**getPlot**();**

**if** **(**type **==** 0**)** **{**

BarRenderer barRenderer **=** **(**BarRenderer**)** plot**.**getRenderer**();**

barRenderer**.**setSeriesPaint**(**0**,** Color**.**BLUE**);**

**}**

**try** **{**

OutputStream os **=** **new** FileOutputStream**(**"F:\\CSCClasses\\CSC 895\\Plot\\" **+** name **+** ".jpg"**);**

ChartUtilities**.**writeChartAsJPEG**(**os**,** chart**,** 500**,** 300**);**

**}** **catch** **(**Exception e**)** **{**

System**.**out**.**println**(**e**.**getMessage**());**

**}**

**return** chart**;**

**}**

public static JFreeChart generateHistogramChart**(**ArrayList**<**Double**>** list**,** int bars**,** String name**)** **{**

DefaultCategoryDataset d **=** **new** DefaultCategoryDataset**();**

double max **=** Collections**.**max**(**list**);**

System**.**out**.**println**(**"length: " **+** list**.**size**()** **+** name **+** ": " **+** max**);**

double width **=** max **/** bars**;**

int size **=** list**.**size**();**

int index**;**

int counterList**[]** **=** **new** int**[**bars**];**

**for** **(**int i **=** 0**;** i **<** bars**;** i**++)** **{**

counterList**[**i**]** **=** 0**;**

**}**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)** **{**

index **=** **(**int**)** **(**list**.**get**(**i**)** **/** width**);**

**if** **(**index **>=** bars**)** **{**

**continue;**

**}**

counterList**[**index**]++;**

**}**

DecimalFormat df **=** **new** DecimalFormat**(**"0.00"**);**

**for** **(**int i **=** 0**;** i **<** bars**;** i**++)** **{**

d**.**setValue**((**double**)** counterList**[**i**]** **/** size**,** "Relative Frequency"**,** df**.**format**((**i **+** 1**)** **\*** width**).**toString**());**

**}**

JFreeChart chart **=** ChartFactory**.**createBarChart**(**name**,** "Time"**,** "Relative Frequency"**,** d**,** PlotOrientation**.**VERTICAL**,** **false,** **false,** **false);**

CategoryPlot plot **=** **(**CategoryPlot**)** chart**.**getPlot**();**

CategoryAxis domainAxis **=** **(**CategoryAxis**)** plot**.**getDomainAxis**();**

domainAxis**.**setCategoryLabelPositions**(**CategoryLabelPositions**.**UP\_45**);**

**return** chart**;**

**}**

public static JFreeChart generateHistogramChart1**(**ArrayList**<**Double**>** list**,** int bars**,** String name**){**

DefaultCategoryDataset d **=** **new** DefaultCategoryDataset**();**

int size **=** list**.**size**();**

int counterList**[]** **=** **new** int**[**bars**];**

**for** **(**int i **=** 0**;** i **<** bars**;** i**++)** **{**

counterList**[**i**]** **=** 0**;**

**}**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)** **{**

**for(**int j **=** 1**;** j **<=** bars**;** j**++){**

**if(**list**.**get**(**i**)** **==** j**){**

counterList**[**j **-** 1**]++;**

**}**

**}**

**}**

DecimalFormat df **=** **new** DecimalFormat**(**"0.00"**);**

**for** **(**int i **=** 0**;** i **<** bars**;** i**++)** **{**

d**.**setValue**((**double**)** counterList**[**i**]** **/** size**,** "Relative Frequency"**,** df**.**format**((**i **+** 1**)).**toString**());**

**}**

JFreeChart chart **=** ChartFactory**.**createBarChart**(**name**,** "Time"**,** "Relative Frequency"**,** d**,** PlotOrientation**.**VERTICAL**,** **false,** **false,** **false);**

CategoryPlot plot **=** **(**CategoryPlot**)** chart**.**getPlot**();**

CategoryAxis domainAxis **=** **(**CategoryAxis**)** plot**.**getDomainAxis**();**

domainAxis**.**setCategoryLabelPositions**(**CategoryLabelPositions**.**UP\_45**);**

**return** chart**;**

**}**

public static JFreeChart generateHistogramChart2**(**ArrayList**<**Double**>** list**,** int bars**,** String name**,** double interval**){**

DefaultCategoryDataset d **=** **new** DefaultCategoryDataset**();**

double max **=** Collections**.**max**(**list**);**

System**.**out**.**println**(**"length: " **+** list**.**size**()** **+** name **+** ": " **+** max**);**

int size **=** list**.**size**();**

int index**;**

int counterList**[]** **=** **new** int**[**bars**];**

**for** **(**int i **=** 0**;** i **<** bars**;** i**++)** **{**

counterList**[**i**]** **=** 0**;**

**}**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)** **{**

index **=** **(**int**)** **(**list**.**get**(**i**)** **/** interval**);**

**if** **(**index **>=** bars**)** **{**

**continue;**

**}**

counterList**[**index**]++;**

**}**

DecimalFormat df **=** **new** DecimalFormat**(**"0.00"**);**

**for** **(**int i **=** 0**;** i **<** bars**;** i**++)** **{**

d**.**setValue**((**double**)** counterList**[**i**]** **/** size**,** "Relative Frequency"**,** df**.**format**((**i **+** 1**)** **\*** interval**).**toString**());**

**}**

JFreeChart chart **=** ChartFactory**.**createBarChart**(**name**,** "Time"**,** "Relative Frequency"**,** d**,** PlotOrientation**.**VERTICAL**,** **false,** **false,** **false);**

CategoryPlot plot **=** **(**CategoryPlot**)** chart**.**getPlot**();**

CategoryAxis domainAxis **=** **(**CategoryAxis**)** plot**.**getDomainAxis**();**

domainAxis**.**setCategoryLabelPositions**(**CategoryLabelPositions**.**UP\_45**);**

**return** chart**;**

**}**

public static JFreeChart generateHistogramChart**(**int**[]** list**,** double max**,** String name**,** int type**)** **{**

DefaultCategoryDataset d **=** **new** DefaultCategoryDataset**();**

double width **=** max **/** list**.**length**;**

int size **=** Utility**.**getSum**(**list**);**

DecimalFormat df **=** **new** DecimalFormat**(**"0.00"**);**

**for** **(**int i **=** 0**;** i **<** list**.**length**;** i**++)** **{**

d**.**setValue**((**double**)** list**[**i**]** **/** size**,** "Relative Frequency"**,** df**.**format**((**i **+** 1**)** **\*** width**).**toString**());**

**}**

JFreeChart chart **=** ChartFactory**.**createBarChart**(**name**,** "Time"**,** "Relative Frequency"**,** d**,** PlotOrientation**.**VERTICAL**,** **false,** **false,** **false);**

CategoryPlot plot **=** **(**CategoryPlot**)** chart**.**getPlot**();**

**if** **(**type **==** 0**)** **{**

BarRenderer barRenderer **=** **(**BarRenderer**)** plot**.**getRenderer**();**

barRenderer**.**setSeriesPaint**(**0**,** Color**.**BLUE**);**

**}**

CategoryAxis domainAxis **=** **(**CategoryAxis**)** plot**.**getDomainAxis**();**

domainAxis**.**setCategoryLabelPositions**(**CategoryLabelPositions**.**UP\_45**);**

**try** **{**

OutputStream os **=** **new** FileOutputStream**(**"F:\\CSCClasses\\CSC 895\\Plot\\" **+** name **+** ".jpg"**);**

ChartUtilities**.**writeChartAsJPEG**(**os**,** chart**,** 500**,** 300**);**

**}** **catch** **(**Exception e**)** **{**

System**.**out**.**println**(**e**.**getMessage**());**

**}**

**return** chart**;**

**}**

public static JFreeChart generateHistogramChart2**(**int**[]** list**,** int total**,** double interval**,** String name**,** int type**)** **{**

DefaultCategoryDataset d **=** **new** DefaultCategoryDataset**();**

DecimalFormat df **=** **new** DecimalFormat**(**"0.00"**);**

**for** **(**int i **=** 0**;** i **<** list**.**length**;** i**++)** **{**

d**.**setValue**((**double**)** list**[**i**]** **/** total**,** "Relative Frequency"**,** df**.**format**((**i **+** 1**)** **\*** interval**).**toString**());**

**}**

JFreeChart chart **=** ChartFactory**.**createBarChart**(**name**,** "Time"**,** "Relative Frequency"**,** d**,** PlotOrientation**.**VERTICAL**,** **false,** **false,** **false);**

CategoryPlot plot **=** **(**CategoryPlot**)** chart**.**getPlot**();**

**if** **(**type **==** 0**)** **{**

BarRenderer barRenderer **=** **(**BarRenderer**)** plot**.**getRenderer**();**

barRenderer**.**setSeriesPaint**(**0**,** Color**.**BLUE**);**

**}**

CategoryAxis domainAxis **=** **(**CategoryAxis**)** plot**.**getDomainAxis**();**

domainAxis**.**setCategoryLabelPositions**(**CategoryLabelPositions**.**UP\_45**);**

**try** **{**

OutputStream os **=** **new** FileOutputStream**(**"F:\\CSCClasses\\CSC 895\\Plot\\" **+** name **+** ".jpg"**);**

ChartUtilities**.**writeChartAsJPEG**(**os**,** chart**,** 500**,** 300**);**

**}** **catch** **(**Exception e**)** **{**

System**.**out**.**println**(**e**.**getMessage**());**

**}**

**return** chart**;**

**}**

**}**

/\*

\* Utility.java

\*/

package gg1simulator**;**

**import** java**.**text**.**DecimalFormat**;**

**import** java**.**util**.\*;**

public class Utility **{**

public static double getSum**(**ArrayList**<**Double**>** list**)** **{**

double sum **=** 0**;**

**for** **(**Double d **:** list**)** **{**

sum **+=** d**;**

**}**

**return** sum**;**

**}**

public static int getSum**(**int**[]** list**)** **{**

int sum **=** 0**;**

**for** **(**int i **:** list**)** **{**

sum **+=** i**;**

**}**

**return** sum**;**

**}**

public static double getSum**(**HashMap**<**Integer**,** Double**>** map**)** **{**

double sum **=** 0**;**

Iterator it **=** map**.**entrySet**().**iterator**();**

**while** **(**it**.**hasNext**())** **{**

Map**.**Entry pairs **=** **(**Map**.**Entry**)** it**.**next**();**

sum **+=** **(**double**)**pairs**.**getValue**();**

it**.**remove**();**

**}**

**return** sum**;**

**}**

/\*\*

\* get mean value of the list

\*

\* **@param** list

\* **@return** double result

\*/

public static double getMean**(**ArrayList**<**Double**>** list**)** **{**

**return** getSum**(**list**)** **/** **(**list**.**size**()** **\*** 1.0**);**

**}**

/\*\*

\* get standard deviation of the list

\*

\* **@param** list

\* **@return** double result

\*/

public static double getSD**(**ArrayList**<**Double**>** list**)** **{**

double mean **=** getMean**(**list**),** sum **=** 0**;**

**for** **(**double d **:** list**)** **{**

sum **+=** Math**.**pow**(**d **-** mean**,** 2**);**

**}**

**return** Math**.**sqrt**(**sum **/** **(**list**.**size**()** **-** 1**));**

**}**

public static double getAverage**(**HashMap**<**Integer**,** Double**>** map**)** **{**

double total **=** 0**;**

double totalTime **=** 0**;**

**for** **(**Map**.**Entry**<**Integer**,** Double**>** e **:** map**.**entrySet**())** **{**

total **+=** e**.**getKey**()** **\*** e**.**getValue**();**

totalTime **+=** e**.**getValue**();**

**}**

**return** total **/** totalTime**;**

**}**

public static double getStandardDeviation**(**HashMap**<**Integer**,** Double**>** map**,** double average**)** **{**

double total **=** 0**;**

double totalTime **=** 0**;**

**for** **(**Map**.**Entry**<**Integer**,** Double**>** e **:** map**.**entrySet**())** **{**

total **+=** e**.**getKey**()** **\*** e**.**getKey**()** **\*** e**.**getValue**();**

totalTime **+=** e**.**getValue**();**

**}**

total **/=** totalTime**;**

total **-=** average **\*** average**;**

**return** Math**.**sqrt**(**total**);**

**}**

public static String formatOutput**(**double d**)** **{**

DecimalFormat df **=** **new** DecimalFormat**(**"##0.000"**);**

**return** df**.**format**(**d**);**

**}**

public static void print**(**String s**,** int length**)** **{**

System**.**out**.**print**(**s**);**

**if** **(**s**.**length**()** **<** length**)** **{**

**for** **(**int i **=** 0**;** i **<** length **-** s**.**length**();** i**++)** **{**

System**.**out**.**print**(**" "**);**

**}**

**}**

**}**

**}**

/\*

\* GG1Simulator.java

\*/

package gg1simulator**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**HashMap**;**

**import** java**.**util**.**LinkedList**;**

**import** java**.**util**.**PriorityQueue**;**

**import** java**.**util**.**Queue**;**

public class GG1Simulator **{**

protected int bars**;**

protected double interval**;**

protected Distribution arrivalDistribution**;**

protected Distribution serviceDistribution**;**

protected double t**,** lastEventTime **=** 0**,** delta**;**

protected PriorityQueue**<**Event**>** eventQueue **=** **new** PriorityQueue**<>();**

protected int customers**;**

protected int midCustomers**;**

private int numOfArrivaledJobs **=** 0**,** numOfServicedJobs **=** 0**;**

private ArrayList**<**Double**>** interArrival **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** interDeparture **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** busyTime **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** responseTime **=** **new** ArrayList**<>();**

private HashMap**<**Integer**,** Double**>** queueMap **=** **new** HashMap**<>();**

private Queue**<**Jobs**>** jobsQueue **=** **new** LinkedList**<>();**

private double maxInterArrivalTime **=** 0**,** maxServiceTime **=** 0**,** maxInterDepartureTime **=** 0**,**

maxResponseTime **=** 0**;**

private double sumInterArrivalTime**,** sumServiceTime**,** sumInterDepartureTime**,**

sumResponseTime**,** sumQueueLength**,** sumBusyTime**;**

private double sdInterArrivalTime**,** sdServiceTime**,** sdInterDepartureTime**,**

sdResponseTime**,** sdQueueLength**;**

private int**[]** interArrivalList**,** serviceList**,** interDepartureList**,** responseTimeList**;**

public GG1Simulator**(**Distribution arrivalDistribution**,** Distribution serviceDistribution**,** int bars**,** double interval**)** **{**

**this.**arrivalDistribution **=** arrivalDistribution**;**

**this.**serviceDistribution **=** serviceDistribution**;**

**this.**bars **=** bars**;**

**this.**interval **=** interval**;**

initArray**();**

**}**

public final void initArray**()** **{**

interArrivalList **=** **new** int**[**bars**];**

serviceList **=** **new** int**[**bars**];**

interDepartureList **=** **new** int**[**bars**];**

responseTimeList **=** **new** int**[**bars**];**

**for** **(**int i **=** 0**;** i **<** bars**;** i**++)** **{**

interArrivalList**[**i**]** **=** 0**;**

serviceList**[**i**]** **=** 0**;**

interDepartureList**[**i**]** **=** 0**;**

responseTimeList**[**i**]** **=** 0**;**

**}**

**}**

public int getLength**()** **{**

**return** customers**;**

**}**

public double getDelta**()** **{**

**return** delta**;**

**}**

public int getBars**()** **{**

**return** bars**;**

**}**

public void run**(**int servicedJobs**)** **{**

double arrivalTime**,** serviceTime**,** rTime**;**

double tempArraival **=** 0**,** tempDeparture **=** 0**;**

int jobArrivalId **=** 1**,** index**;**

arrivalTime **=** arrivalDistribution**.**generateRandomTimes**();**

eventQueue**.**add**(new** Event**(**t**,** t **+** arrivalTime**,** Event**.**ARRIVAL**,** 1**));**

**while** **(**numOfServicedJobs **<** servicedJobs**)** **{**

Event e **=** eventQueue**.**poll**();**

t **=** e**.**endsTime**;** //current simulation time

delta **=** t **-** lastEventTime**;** //

sumQueueLength **+=** delta **\*** customers**;**

sdQueueLength **+=** delta **\*** customers **\*** customers**;**

lastEventTime **=** t**;**

Double length **=** queueMap**.**get**(**customers**);**

**if** **(**length **==** **null)** **{**

length **=** 0.0**;**

**}**

queueMap**.**put**(**customers**,** length **+** delta**);**

**switch** **(**e**.**code**)** **{**

**case** Event**.**ARRIVAL**:**

customers**++;**

numOfArrivaledJobs**++;**

interArrival**.**add**(**t **-** tempArraival**);**

sumInterArrivalTime **+=** t **-** tempArraival**;**

sdInterArrivalTime **+=** **(**t **-** tempArraival**)** **\*** **(**t **-** tempArraival**);**

tempArraival **=** t**;**

Jobs jobA **=** **new** Jobs**(**jobArrivalId**++,** t**,** t**);**

jobsQueue**.**add**(**jobA**);**

arrivalTime **=** arrivalDistribution**.**generateRandomTimes**();**

index **=** **(**int**)** **(**arrivalTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

interArrivalList**[**index**]++;**

**}**

// sumArrivalTime += arrivalTime;

// sdArrivalTime += arrivalTime \* arrivalTime;

eventQueue**.**add**(new** Event**(**t**,** t **+** arrivalTime**,** Event**.**ARRIVAL**,** 1**));**

**if** **(**customers **==** 1**)** **{**

serviceTime **=** serviceDistribution**.**generateRandomTimes**();**

index **=** **(**int**)** **(**serviceTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

serviceList**[**index**]++;**

**}**

sumServiceTime **+=** serviceTime**;**

sdServiceTime **+=** serviceTime **\*** serviceTime**;**

eventQueue**.**add**(new** Event**(**t**,** t **+** serviceTime**,** Event**.**DEPARTURE**,** 1**));**

**}**

**break;**

**case** Event**.**DEPARTURE**:**

customers**--;**

numOfServicedJobs**++;**

//interDeparture.add(t - tempDeparture);

index **=** **(**int**)** **((**t **-** tempDeparture**)** **/** interval**);**

**if** **(**index **<** bars**)** **{**

interDepartureList**[**index**]++;**

**}**

sumInterDepartureTime **+=** t **-** tempDeparture**;**

sdInterDepartureTime **+=** **(**t **-** tempDeparture**)** **\*** **(**t **-** tempDeparture**);**

tempDeparture **=** t**;**

Jobs jobD **=** jobsQueue**.**poll**();**

jobD**.**setDepartureTime**(**t**);**

rTime **=** jobD**.**getDepartureTime**()** **-** jobD**.**getArrivalTime**();**

index **=** **(**int**)** **(**rTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

responseTimeList**[**index**]++;**

**}**

responseTime**.**add**(**rTime**);**

sumResponseTime **+=** rTime**;**

sdResponseTime **+=** rTime **\*** rTime**;**

**if** **(**customers **>** 0**)** **{**

serviceTime **=** serviceDistribution**.**generateRandomTimes**();**

index **=** **(**int**)** **(**serviceTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

serviceList**[**index**]++;**

**}**

sumServiceTime **+=** serviceTime**;**

sdServiceTime **+=** serviceTime **\*** serviceTime**;**

eventQueue**.**add**(new** Event**(**t**,** t **+** serviceTime**,** Event**.**DEPARTURE**,** 1**));**

**}**

**break;**

**}**

**}**

**}**

public double getMaxInterArrivalTime**()** **{**

**return** maxInterArrivalTime**;**

**}**

public double getMaxInterDepartureTime**()** **{**

**return** maxInterDepartureTime**;**

**}**

public double getMaxServiceTime**()** **{**

**return** maxServiceTime**;**

**}**

public double getMaxResponseTime**()** **{**

**return** maxResponseTime**;**

**}**

public int**[]** getInterArrivalList**()** **{**

**return** interArrivalList**;**

**}**

public int**[]** getServiceList**()** **{**

**return** serviceList**;**

**}**

public int**[]** getInterDepartureList**()** **{**

**return** interDepartureList**;**

**}**

public int**[]** getResponseTimeList**()** **{**

**return** responseTimeList**;**

**}**

public ArrayList**<**Double**>** getInterArrival**()** **{**

**return** interArrival**;**

**}**

public ArrayList**<**Double**>** getInterDeparture**()** **{**

**return** interDeparture**;**

**}**

public ArrayList**<**Double**>** getBusyTime**()** **{**

**return** busyTime**;**

**}**

public ArrayList**<**Double**>** getResponseTime**()** **{**

**return** responseTime**;**

**}**

public HashMap**<**Integer**,** Double**>** getQueueMap**()** **{**

**return** queueMap**;**

**}**

public int getArrivals**(){**

**return** numOfArrivaledJobs**;**

**}**

public int getDepartures**(){**

**return** numOfServicedJobs**;**

**}**

public double getUtilization**()** **{**

**return** sumServiceTime **/** t**;**

**}**

public double getAvgInterArrivalTime**()** **{**

**return** sumInterArrivalTime **/** numOfArrivaledJobs**;**

**}**

public double getSDInterArrivalTime**()** **{**

**return** Math**.**sqrt**(**sdInterArrivalTime **/** numOfArrivaledJobs **-** Math**.**pow**(**getAvgInterArrivalTime**(),** 2**));**

**}**

public double getAvgResponseTime**()** **{**

**return** sumResponseTime **/** numOfServicedJobs**;**

**}**

public double getSDResponseTime**()** **{**

**return** Math**.**sqrt**(**sdResponseTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgResponseTime**(),** 2**));**

**}**

public double getAvgServiceTime**()** **{**

**return** sumServiceTime **/** numOfServicedJobs**;**

**}**

public double getSDServiceTime**()** **{**

**return** Math**.**sqrt**(**sdServiceTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgServiceTime**(),** 2**));**

**}**

public double getAvgInterDeparutreTime**()** **{**

**return** sumInterDepartureTime **/** numOfServicedJobs**;**

**}**

public double getSDInterDeparutreTime**()** **{**

**return** Math**.**sqrt**(**sdInterDepartureTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgInterDeparutreTime**(),** 2**));**

**}**

public double getAvgJobsIntheSystem**()** **{**

**return** sumQueueLength **/** t**;**

**}**

public double getSDJobsIntheSystem**()** **{**

**return** Math**.**sqrt**(**sdQueueLength **/** t **-** Math**.**pow**(**getAvgJobsIntheSystem**(),** 2**));**

**}**

**}**

/\*

\* FeedbackQueueSimulator.java

\*/

package gg1simulator**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**HashMap**;**

**import** java**.**util**.**LinkedList**;**

**import** java**.**util**.**Map**.**Entry**;**

**import** java**.**util**.**PriorityQueue**;**

**import** java**.**util**.**Queue**;**

**import** java**.**util**.**Random**;**

public class FeedbackQueueSimulator **{**

protected Distribution arrivalDistribution**;**

protected Distribution serviceDistribution**;**

protected Distribution midServiceDistribution**;**

private double interval**;**

private int bars**;**

protected double p**;**

protected double t**,** lastEventTime **=** 0**,** delta**;**

protected int jobsInTheSystem**;**

protected PriorityQueue**<**Event**>** eventQueue **=** **new** PriorityQueue**<>();**

private PriorityQueue**<**Jobs**>** mainJobsQueue **=** **new** PriorityQueue**<>();**

private Queue**<**Jobs**>** branchJobsQueue **=** **new** LinkedList**<>();**

private Queue**<**Jobs**>** servedJobsQueue **=** **new** LinkedList**<>();**

protected int customers**,** midCustomers**;**

private Random r **=** **new** Random**();**

private ArrayList**<**Double**>** interArrivalTime **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** interTotalArrivalTime **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** interDepartureTime **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** interMidArrivalTime **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** interMidDepartureTime **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** busyTimeFirstServer **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** busyTimeSecondServer **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** responseTime **=** **new** ArrayList**<>();**

private HashMap**<**Integer**,** Double**>** queueLength **=** **new** HashMap**<>();**

private int numOfServicedJobs **=** 0**,** numOfMidArrivaledJobs **=** 0**,** numOfTotalArrivaledJobs **=** 0**,** numOfArrivaledJobs **=** 0**,**

numOfMidServicedJobs **=** 0**,** numOfJobs **=** 0**;**

private double maxInterArrivalTime **=** 0**,** maxTotalInterArrivalTime **=** 0**,** maxInterDepartureTime **=** 0**,**

maxInterMidArrivalTime **=** 0**,** maxInterMidDepartureTime **=** 0**,** maxServiceTime1 **=** 0**,**

maxServiceTime **=** 0**,** maxResponseTime **=** 0**;**

private double sumInterArrivalTime**,** sumTotalInterArrivalTime**,** sumInterDepartureTime**,** sumMidArrivalTime**,**

sumInterMidDepartureTime**,** sumServiceTime1**,** sumServiceTime**,** sumResponseTime**,** sumQueueLength**;**

private double sdInterArrivalTime**,** sdTotalInterArrivalTime**,** sdInterDepartureTime**,** sdMidArrivalTime**,**

sdInterMidDepartureTime**,** sdServiceTime1**,** sdServiceTime**,** sdResponseTime**,** sdQueueLength**;**

private int**[]** interArrivalTimeList**,** totalInterArrivalTimeList**,** interDepartureTimeList**,** midArrivalTimeList**,**

interMidDepartureTimeList**,** serviceTimeList1**,** serviceTimeList**,** responseTimeList**;**

public FeedbackQueueSimulator**(**Distribution arrivalDistribution**,** Distribution serviceDistribution**,** Distribution midServiceDistribution**,** int bars**,** double interval**,** double p**)** **{**

**this.**arrivalDistribution **=** arrivalDistribution**;**

**this.**serviceDistribution **=** serviceDistribution**;**

**this.**midServiceDistribution **=** midServiceDistribution**;**

**this.**bars **=** bars**;**

**this.**interval **=** interval**;**

**this.**p **=** p**;**

initArray**();**

**}**

public final void initArray**()** **{**

interArrivalTimeList **=** **new** int**[**bars**];**

totalInterArrivalTimeList **=** **new** int**[**bars**];**

interDepartureTimeList **=** **new** int**[**bars**];**

midArrivalTimeList **=** **new** int**[**bars**];**

interMidDepartureTimeList **=** **new** int**[**bars**];**

serviceTimeList1 **=** **new** int**[**bars**];**

serviceTimeList **=** **new** int**[**bars**];**

responseTimeList **=** **new** int**[**bars**];**

responseTimeList **=** **new** int**[**bars**];**

**for** **(**int i **=** 0**;** i **<** bars**;** i**++)** **{**

interArrivalTimeList**[**i**]** **=** 0**;**

totalInterArrivalTimeList**[**i**]** **=** 0**;**

interDepartureTimeList**[**i**]** **=** 0**;**

midArrivalTimeList**[**i**]** **=** 0**;**

interMidDepartureTimeList**[**i**]** **=** 0**;**

serviceTimeList1**[**i**]** **=** 0**;**

serviceTimeList**[**i**]** **=** 0**;**

responseTimeList**[**i**]** **=** 0**;**

responseTimeList**[**i**]** **=** 0**;**

**}**

**}**

public int getCustomers**()** **{**

**return** customers**;**

**}**

public int getNumOfMidArrivalJobs**()** **{**

**return** numOfMidArrivaledJobs**;**

**}**

public int getMidCustomers**()** **{**

**return** midCustomers**;**

**}**

public double getDelta**()** **{**

**return** delta**;**

**}**

public Queue**<**Jobs**>** getServerdJobsQueue**()** **{**

**return** servedJobsQueue**;**

**}**

public void run**(**int servicedJobs**)** **{**

double arrivalTime**,** serviceTime**,** midServiceTime**,** rTime**;**

double tempArrivalTime **=** 0**,** tempTotalArrayTime **=** 0**,** tempDepartureTime **=** 0**,** tempMidArraialTime **=** 0**,** tempMidDepartureTime **=** 0**;**

arrivalTime **=** arrivalDistribution**.**generateRandomTimes**();**

eventQueue**.**add**(new** Event**(**t**,** t **+** arrivalTime**,** Event**.**ARRIVAL**,** 1**));**

int jobId **=** 1**,** index**;**

Jobs job**;**

**while** **(**numOfServicedJobs **<** servicedJobs**)** **{**

Event e **=** eventQueue**.**poll**();**

t **=** e**.**endsTime**;** //current simulation time

delta **=** t **-** lastEventTime**;**

lastEventTime **=** t**;**

jobsInTheSystem **=** customers **+** midCustomers**;**

sumQueueLength **+=** delta **\*** jobsInTheSystem**;**

sdQueueLength **+=** delta **\*** jobsInTheSystem **\*** jobsInTheSystem**;**

Double length **=** queueLength**.**get**(**jobsInTheSystem**);**

**if** **(**length **==** **null)** **{**

length **=** 0.0**;**

**}**

queueLength**.**put**(**jobsInTheSystem**,** length **+** delta**);**

**switch** **(**e**.**code**)** **{**

**case** Event**.**ARRIVAL**:** // Next Arrival Time

customers**++;**

numOfTotalArrivaledJobs**++;**

//interTotalArrivalTime.add(t - tempTotalArrayTime);

sumTotalInterArrivalTime **+=** t **-** tempTotalArrayTime**;**

sdTotalInterArrivalTime **+=** **(**t **-** tempTotalArrayTime**)** **\*** **(**t **-** tempTotalArrayTime**);**

index **=** **(**int**)** **((**t **-** tempTotalArrayTime**)** **/** interval**);**

**if** **(**index **<** bars**)** **{**

totalInterArrivalTimeList**[**index**]++;**

**}**

tempTotalArrayTime **=** t**;**

**if** **(**e**.**getBranch**()** **==** 1**)** **{**

numOfArrivaledJobs**++;**

//interArrivalTime.add(t - tempArrivalTime);

sumInterArrivalTime **+=** t **-** tempArrivalTime**;**

sdInterArrivalTime **+=** **(**t **-** tempArrivalTime**)** **\*** **(**t **-** tempArrivalTime**);**

tempArrivalTime **=** t**;**

job **=** **new** Jobs**(**jobId**++,** t**,** t**);**

mainJobsQueue**.**add**(**job**);**

arrivalTime **=** arrivalDistribution**.**generateRandomTimes**();**

//System.out.println("a: " + arrivalTime);

index **=** **(**int**)** **(**arrivalTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

interArrivalTimeList**[**index**]++;**

**}**

eventQueue**.**add**(new** Event**(**t**,** t **+** arrivalTime**,** Event**.**ARRIVAL**,** 1**));**

**}**

**if** **(**e**.**getBranch**()** **==** 2**)** **{**

job **=** branchJobsQueue**.**poll**();**

job**.**setLastArrivaltime**(**t**);**

mainJobsQueue**.**add**(**job**);**

**}**

**if** **(**customers **==** 1**)** **{** // the first customer arrives and will be served ASAP , so the queue is empty

serviceTime **=** serviceDistribution**.**generateRandomTimes**();**

sumServiceTime1 **+=** serviceTime**;**

sdServiceTime1 **+=** serviceTime **\*** serviceTime**;**

index **=** **(**int**)** **(**serviceTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

serviceTimeList1**[**index**]++;**

**}**

//busyTimeFirstServer.add(serviceTime);

eventQueue**.**add**(new** Event**(**t**,** t **+** serviceTime**,** Event**.**DEPARTURE**,** 1**));**

**}**

**break;**

**case** Event**.**DEPARTURE**:**

customers**--;**

numOfJobs**++;**

double temp **=** r**.**nextDouble**();**

**if** **(**temp **>** p**)** **{**

numOfServicedJobs**++;**

sumInterDepartureTime **+=** t **-** tempDepartureTime**;**

sdInterDepartureTime **+=** **(**t **-** tempDepartureTime**)** **\*** **(**t **-** tempDepartureTime**);**

index **=** **(**int**)((**t **-** tempDepartureTime**)** **/** interval**);**

**if(**index **<** bars**){**

interDepartureTimeList**[**index**]++;**

**}**

//interDepartureTime.add(t - tempDepartureTime);

tempDepartureTime **=** t**;**

job **=** mainJobsQueue**.**poll**();**

job**.**setDepartureTime**(**t**);**

rTime **=** t **-** job**.**getArrivalTime**();**

sumResponseTime **+=** rTime**;**

sdResponseTime **+=** rTime **\*** rTime**;**

responseTime**.**add**(**rTime**);**

index **=** **(**int**)** **(**rTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

responseTimeList**[**index**]++;**

**}**

**}** **else** **{**

eventQueue**.**add**(new** Event**(**t**,** t**,** Event**.**MIDARRIVAL**,** 1**));**

**}**

**if** **(**customers **>** 0**)** **{**

serviceTime **=** serviceDistribution**.**generateRandomTimes**();**

sumServiceTime1 **+=** serviceTime**;**

sdServiceTime1 **+=** serviceTime **\*** serviceTime**;**

index **=** **(**int**)** **(**serviceTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

serviceTimeList1**[**index**]++;**

**}**

//busyTimeFirstServer.add(serviceTime);

eventQueue**.**add**(new** Event**(**t**,** t **+** serviceTime**,** Event**.**DEPARTURE**,** 1**));**

**}**

**break;**

**case** Event**.**MIDARRIVAL**:**

numOfMidArrivaledJobs**++;**

midCustomers**++;**

sumMidArrivalTime **+=** t **-** tempMidArraialTime**;**

sdMidArrivalTime **+=** **(**t **-** tempMidArraialTime**)** **\*** **(**t **-** tempMidArraialTime**);**

index **=** **(**int**)** **((**t **-** tempMidArraialTime**)** **/** interval**);**

**if** **(**index **<** bars**)** **{**

midArrivalTimeList**[**index**]++;**

**}**

//interMidArrivalTime.add(t - tempMidArraialTime);

tempMidArraialTime **=** t**;**

job **=** mainJobsQueue**.**poll**();**

job**.**setMidArrivalTime**(**t**);**

branchJobsQueue**.**add**(**job**);**

**if** **(**midCustomers **==** 1**)** **{**

midServiceTime **=** midServiceDistribution**.**generateRandomTimes**();**

sumServiceTime **+=** midServiceTime**;**

sdServiceTime **+=** midServiceTime **\*** midServiceTime**;**

index **=** **(**int**)** **(**midServiceTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

serviceTimeList**[**index**]++;**

**}**

//busyTimeSecondServer.add(midServiceTime);

eventQueue**.**add**(new** Event**(**t**,** t **+** midServiceTime**,** Event**.**MIDDEPARTURE**,** 2**));**

**}**

**break;**

**case** Event**.**MIDDEPARTURE**:**

midCustomers**--;**

numOfMidServicedJobs**++;**

sumInterMidDepartureTime **+=** t **-** tempMidDepartureTime**;**

sdInterMidDepartureTime **+=** **(**t **-** tempMidDepartureTime**)** **\*** **(**t **-** tempMidDepartureTime**);**

index **=** **(**int**)** **((**t **-** tempMidDepartureTime**)** **/** interval**);**

**if** **(**index **<** bars**)** **{**

interMidDepartureTimeList**[**index**]++;**

**}**

//interMidDepartureTime.add(t - tempMidDepartureTime);

tempMidDepartureTime **=** t**;**

eventQueue**.**add**(new** Event**(**t**,** t**,** Event**.**ARRIVAL**,** 2**));**

**if** **(**midCustomers **>** 0**)** **{**

midServiceTime **=** midServiceDistribution**.**generateRandomTimes**();**

sumServiceTime **+=** midServiceTime**;**

sdServiceTime **+=** midServiceTime **\*** midServiceTime**;**

index **=** **(**int**)** **(**midServiceTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

serviceTimeList**[**index**]++;**

**}**

//busyTimeSecondServer.add(midServiceTime);

eventQueue**.**add**(new** Event**(**t**,** t **+** midServiceTime**,** Event**.**MIDDEPARTURE**,** 2**));**

**}**

**break;**

**}**

**}**

**}**

public ArrayList**<**Double**>** getInterArrivalTime**()** **{**

**return** interArrivalTime**;**

**}**

public ArrayList**<**Double**>** getInterTotalArrivalTime**()** **{**

**return** interTotalArrivalTime**;**

**}**

public ArrayList**<**Double**>** getInterDepartureTime**()** **{**

**return** interDepartureTime**;**

**}**

public ArrayList**<**Double**>** getInterMidArrivalTime**()** **{**

**return** interMidArrivalTime**;**

**}**

public ArrayList**<**Double**>** getInterMidDepartureTime**()** **{**

**return** interMidDepartureTime**;**

**}**

public ArrayList**<**Double**>** getBusyTimeFirstServer**()** **{**

**return** busyTimeFirstServer**;**

**}**

public ArrayList**<**Double**>** getResponseTime**(){**

**return** responseTime**;**

**}**

public ArrayList**<**Double**>** getBusyTimeSecondServer**()** **{**

**return** busyTimeSecondServer**;**

**}**

public HashMap**<**Integer**,** Double**>** getQueueLength**()** **{**

**return** queueLength**;**

**}**

/\*\*

\* get sum of the list

\*

\* **@param** list

\* **@return**

\*/

public double getSum**(**ArrayList**<**Double**>** list**)** **{**

double sum **=** 0**;**

**for** **(**Double d **:** list**)** **{**

sum **+=** d**;**

**}**

**return** sum**;**

**}**

/\*\*

\* get mean value of the list

\*

\* **@param** list

\* **@return** double result

\*/

public double getMean**(**ArrayList**<**Double**>** list**)** **{**

**return** getSum**(**list**)** **/** **(**list**.**size**()** **\*** 1.0**);**

**}**

public double getAverageJobs**(**HashMap**<**Integer**,** Double**>** map**)** **{**

double total **=** 0**;**

double totalTime **=** 0**;**

**for** **(**Entry**<**Integer**,** Double**>** e **:** map**.**entrySet**())** **{**

total **+=** e**.**getKey**()** **\*** e**.**getValue**();**

totalTime **+=** e**.**getValue**();**

**}**

**return** total **/** totalTime**;**

**}**

public double getFirstUtilization**()** **{**

**return** sumServiceTime1 **/** t**;**

**}**

public double getSecondUtilization**()** **{**

**return** sumServiceTime **/** t**;**

**}**

public double getMaxInterArrivalTime**()** **{**

**return** maxInterArrivalTime**;**

**}**

public double getMaxTotalInterArrivalTime**()** **{**

**return** maxTotalInterArrivalTime**;**

**}**

public double getMaxInterDepartureTime**()** **{**

**return** maxInterDepartureTime**;**

**}**

public double getMaxInterMidArrivalTime**()** **{**

**return** maxInterMidArrivalTime**;**

**}**

public double getMaxInterMidDepartureTime**()** **{**

**return** maxInterMidDepartureTime**;**

**}**

public double getMaxServiceTime1**()** **{**

**return** maxServiceTime1**;**

**}**

public double getMaxServiceTime**()** **{**

**return** maxServiceTime**;**

**}**

public double getMaxResponseTime**()** **{**

**return** maxResponseTime**;**

**}**

public int**[]** getInterArrivalTimeList**()** **{**

**return** interArrivalTimeList**;**

**}**

public int**[]** getTotalInterArrivalTimeList**()** **{**

**return** totalInterArrivalTimeList**;**

**}**

public int**[]** getInterDepartureTimeList**()** **{**

**return** interDepartureTimeList**;**

**}**

public int**[]** getMidArrivalTimeList**()** **{**

**return** midArrivalTimeList**;**

**}**

public int**[]** getInterMidDepartureTimeList**()** **{**

**return** interMidDepartureTimeList**;**

**}**

public int**[]** getServiceTimeList1**()** **{**

**return** serviceTimeList1**;**

**}**

public int**[]** getServiceTimeList**()** **{**

**return** serviceTimeList**;**

**}**

public int**[]** getResponseTimeList**()** **{**

**return** responseTimeList**;**

**}**

public double getAvgInterArrivalTime**()** **{**

**return** sumInterArrivalTime **/** numOfArrivaledJobs**;**

**}**

public double getSDInterArrivalTime**()** **{**

**return** Math**.**sqrt**(**sdInterArrivalTime **/** numOfArrivaledJobs **-** Math**.**pow**(**getAvgInterArrivalTime**(),** 2**));**

**}**

public double getAvgTotalInterArrivalTime**()** **{**

**return** sumTotalInterArrivalTime **/** numOfTotalArrivaledJobs**;**

**}**

public double getSDTotalInterArrivalTime**()** **{**

**return** Math**.**sqrt**(**sdTotalInterArrivalTime **/** numOfTotalArrivaledJobs **-** Math**.**pow**(**getAvgTotalInterArrivalTime**(),** 2**));**

**}**

public double getAvgInterDepartureTime**()** **{**

**return** sumInterDepartureTime **/** numOfServicedJobs**;**

**}**

public double getSDInterDepartureTime**()** **{**

**return** Math**.**sqrt**(**sdInterDepartureTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgInterDepartureTime**(),** 2**));**

**}**

public double getAvgMidArrivalTime**()** **{**

**return** sumMidArrivalTime **/** numOfMidArrivaledJobs**;**

**}**

public double getSDMidArrivalTime**()** **{**

**return** Math**.**sqrt**(**sdMidArrivalTime **/** numOfMidArrivaledJobs **-** Math**.**pow**(**getAvgMidArrivalTime**(),** 2**));**

**}**

public double getAvgInterMidDepartureTime**()** **{**

**return** sumInterMidDepartureTime **/** numOfMidServicedJobs**;**

**}**

public double getSDInterMidDepartureTime**()** **{**

**return** Math**.**sqrt**(**sdInterMidDepartureTime **/** numOfMidServicedJobs **-** Math**.**pow**(**getAvgInterMidDepartureTime**(),** 2**));**

**}**

public double getAvgServiceTime1**()** **{**

**return** sumServiceTime1 **/** **(**numOfMidArrivaledJobs **+** numOfServicedJobs**);**

**}**

public double getSDServiceTime1**()** **{**

double result **=** sdServiceTime1 **/** **(**numOfMidArrivaledJobs **+** numOfServicedJobs**)** **-** Math**.**pow**(**getAvgServiceTime1**(),** 2**);**

**if(**result **<** 0**){**

**return** 0.0000**;**

**}**

**return** Math**.**sqrt**(**result**);**

**}**

public double getAvgServiceTime**()** **{**

**return** sumServiceTime **/** numOfMidServicedJobs**;**

**}**

public double getSDServiceTime**()** **{**

double result **=** sdServiceTime **/** numOfMidServicedJobs **-** Math**.**pow**(**getAvgServiceTime**(),** 2**);**

**if(**result **<** 0**){**

**return** 0.0000**;**

**}**

**return** Math**.**sqrt**(**result**);**

**}**

public double getAvgResponseTime**()** **{**

**return** sumResponseTime **/** numOfServicedJobs**;**

**}**

public double getSDResponseTime**()** **{**

**return** Math**.**sqrt**(**sdResponseTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgResponseTime**(),** 2**));**

**}**

public double getAvgQueueLength**()** **{**

**return** sumQueueLength **/** t**;**

**}**

public double getSDQueueLength**()** **{**

**return** Math**.**sqrt**(**sdQueueLength **/** t **-** Math**.**pow**(**getAvgQueueLength**(),** 2**));**

**}**

public int getTotalArrials**(){**

**return** numOfTotalArrivaledJobs**;**

**}**

public int getArrivals1**(){**

**return** numOfArrivaledJobs**;**

**}**

public int getArrivals2**(){**

**return** midCustomers**;**

**}**

public int getTotalDepartures**(){**

**return** numOfServicedJobs**;**

**}**

public int getServicedJobs1**(){**

**return** numOfJobs**;**

**}**

public int getServicedJobs2**(){**

**return** numOfMidServicedJobs**;**

**}**

**}**

/\*

\* MultipleServerSimulator.java

\*/

package gg1simulator**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**HashMap**;**

**import** java**.**util**.**LinkedList**;**

**import** java**.**util**.**PriorityQueue**;**

**import** java**.**util**.**Queue**;**

**import** java**.**util**.**Random**;**

public class MultipleServersSimulator **{**

protected int bars**;**

protected double interval**;**

protected Distribution arrivalDistribution**;**

protected Distribution serviceDistribution**;**

protected int numOfServers**;**

protected double t **=** 0**,** lastEventTime **=** 0**,** delta**;**

protected PriorityQueue**<**Event**>** eventQueue **=** **new** PriorityQueue**<>();**

protected int customers**;**

protected int midCustomers**;**

private int numOfArrivaledJobs **=** 0**,** numOfServicedJobs **=** 0**;**

private ArrayList**<**Double**>** interArrival **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** interDeparture **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** busyTime **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** responseTime **=** **new** ArrayList**<>();**

private HashMap**<**Integer**,** Double**>** queueMap **=** **new** HashMap**<>();**

private Queue**<**Jobs**>** jobsQueue **=** **new** LinkedList**<>();**

private boolean**[]** isTerminalBusy**,** isServerBusy**;**

private double maxInterArrivalTime **=** 0**,** maxServiceTime **=** 0**,** maxInterDepartureTime **=** 0**,**

maxResponseTime **=** 0**;**

private double sumInterArrivalTime**,** sumServiceTime**,** sumInterDepartureTime**,**

sumResponseTime**,** sumQueueLength**,** sumBusyTime**,** sumCycleTime**,** sumWaitTime**;**

private double sdInterArrivalTime**,** sdServiceTime**,** sdInterDepartureTime**,**

sdResponseTime**,** sdQueueLength**,** sdCycleTime**,** sdWaitTime**;**

private int**[]** interArrivalList**,** serviceList**,** interDepartureList**,** responseTimeList**,** cycleTimeList**;**

public MultipleServersSimulator**(**Distribution arrivalDistribution**,** Distribution serviceDistribution**,** int numOfServers**,** int bars**,** double interval**)** **{**

**this.**arrivalDistribution **=** arrivalDistribution**;**

**this.**serviceDistribution **=** serviceDistribution**;**

**this.**numOfServers **=** numOfServers**;**

**this.**bars **=** bars**;**

**this.**interval **=** interval**;**

initArray**();**

**}**

public int getNumOfServers**()** **{**

**return** numOfServers**;**

**}**

public final void initArray**()** **{**

interArrivalList **=** **new** int**[**bars**];**

serviceList **=** **new** int**[**bars**];**

interDepartureList **=** **new** int**[**bars**];**

responseTimeList **=** **new** int**[**bars**];**

cycleTimeList **=** **new** int**[**bars**];**

**for** **(**int i **=** 0**;** i **<** bars**;** i**++)** **{**

interArrivalList**[**i**]** **=** 0**;**

serviceList**[**i**]** **=** 0**;**

interDepartureList**[**i**]** **=** 0**;**

responseTimeList**[**i**]** **=** 0**;**

cycleTimeList**[**i**]** **=** 0**;**

**}**

isServerBusy **=** **new** boolean**[**numOfServers**];**

**for** **(**int i **=** 0**;** i **<** numOfServers**;** i**++)** **{**

isServerBusy**[**i**]** **=** **false;**

**}**

**}**

public int getAFreeServer**()** **{**

Random r **=** **new** Random**();**

int num**;**

**do** **{**

num **=** r**.**nextInt**(**numOfServers**);**

**}** **while** **(**isServerBusy**[**num**]);**

**return** num**;**

**}**

public int getLength**()** **{**

**return** customers**;**

**}**

public double getDelta**()** **{**

**return** delta**;**

**}**

public int getBars**()** **{**

**return** bars**;**

**}**

public void run**(**int servicedJobs**)** **{**

double arrivalTime**,** serviceTime**,** rTime**,** cTime**,** wTime**;**

double tempArrival **=** 0**,** tempDeparture **=** 0**;**

int index**;**

arrivalTime **=** arrivalDistribution**.**generateRandomTimes**();**

eventQueue**.**add**(new** Event**(**t**,** t **+** arrivalTime**,** Event**.**ARRIVAL**,** 1**));**

**while** **(**numOfServicedJobs **<** servicedJobs**)** **{**

Event e **=** eventQueue**.**poll**();**

t **=** e**.**endsTime**;** //current simulation time

delta **=** t **-** lastEventTime**;** //

sumQueueLength **+=** delta **\*** customers**;**

sdQueueLength **+=** delta **\*** customers **\*** customers**;**

lastEventTime **=** t**;**

Double length **=** queueMap**.**get**(**customers**);**

**if** **(**length **==** **null)** **{**

length **=** 0.0**;**

**}**

queueMap**.**put**(**customers**,** length **+** delta**);**

**switch** **(**e**.**code**)** **{**

**case** Event**.**ARRIVAL**:**

customers**++;**

numOfArrivaledJobs**++;**

Jobs jobA **=** **new** Jobs**(**e**.**id**,** e**.**startsTime**,** t**,** t**);**

jobsQueue**.**add**(**jobA**);**

arrivalTime **=** arrivalDistribution**.**generateRandomTimes**();**

eventQueue**.**add**(new** Event**(**t**,** t **+** arrivalTime**,** Event**.**ARRIVAL**,** 1**));**

sumInterArrivalTime **+=** t **-** tempArrival**;**

sdInterArrivalTime **+=** **(**t **-** tempArrival**)** **\*** **(**t **-** tempArrival**);**

index **=** **(**int**)** **((**t **-** tempArrival**)** **/** interval**);**

**if** **(**index **<** bars**)** **{**

interArrivalList**[**index**]++;**

**}**

tempArrival **=** t**;**

**if** **(**customers **<=** numOfServers**)** **{**

serviceTime **=** serviceDistribution**.**generateRandomTimes**();**

index **=** **(**int**)** **(**serviceTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

serviceList**[**index**]++;**

**}**

sumServiceTime **+=** serviceTime**;**

sdServiceTime **+=** serviceTime **\*** serviceTime**;**

int branch **=** getAFreeServer**();**

eventQueue**.**add**(new** Event**(**jobA**.**getJobId**(),** t**,** t **+** serviceTime**,** Event**.**DEPARTURE**,** branch **+** 1**));**

isServerBusy**[**branch**]** **=** **true;**

**}**

**break;**

**case** Event**.**DEPARTURE**:**

customers**--;**

numOfServicedJobs**++;**

isServerBusy**[**e**.**branch **-** 1**]** **=** **false;**

index **=** **(**int**)** **((**t **-** tempDeparture**)** **/** interval**);**

**if** **(**index **<** bars**)** **{**

interDepartureList**[**index**]++;**

**}**

sumInterDepartureTime **+=** t **-** tempDeparture**;**

sdInterDepartureTime **+=** **(**t **-** tempDeparture**)** **\*** **(**t **-** tempDeparture**);**

tempDeparture **=** t**;**

Jobs jobD **=** jobsQueue**.**poll**();**

jobD**.**setDepartureTime**(**t**);**

rTime **=** jobD**.**getDepartureTime**()** **-** jobD**.**getLastArrivalTime**();**

//responseTime.add(rTime);

sumResponseTime **+=** rTime**;**

sdResponseTime **+=** rTime **\*** rTime**;**

index **=** **(**int**)** **(**rTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

responseTimeList**[**index**]++;**

**}**

wTime **=** rTime **-** **(**e**.**getEndsTime**()** **-** e**.**getStartsTime**());**

sumWaitTime **+=** wTime**;**

sdWaitTime **+=** wTime **\*** wTime**;**

**if** **(**customers **>** numOfServers **-** 1**)** **{**

serviceTime **=** serviceDistribution**.**generateRandomTimes**();**

index **=** **(**int**)** **(**serviceTime **\*** bars **/** maxServiceTime**);**

**if** **(**index **<** bars**)** **{**

serviceList**[**index**]++;**

**}**

sumServiceTime **+=** serviceTime**;**

sdServiceTime **+=** serviceTime **\*** serviceTime**;**

int id **=** jobsQueue**.**peek**().**getJobId**();**

eventQueue**.**add**(new** Event**(**id**,** t**,** t **+** serviceTime**,** Event**.**DEPARTURE**,** e**.**branch**));**

**}**

**break;**

**}**

**}**

**}**

public double getMaxInterArrivalTime**()** **{**

**return** maxInterArrivalTime**;**

**}**

public double getMaxInterDepartureTime**()** **{**

**return** maxInterDepartureTime**;**

**}**

public double getMaxServiceTime**()** **{**

**return** maxServiceTime**;**

**}**

public double getMaxResponseTime**()** **{**

**return** maxResponseTime**;**

**}**

public int getArrivals**()** **{**

**return** numOfArrivaledJobs**;**

**}**

public int getDepartures**()** **{**

**return** numOfServicedJobs**;**

**}**

public int**[]** getInterArrivalList**()** **{**

**return** interArrivalList**;**

**}**

public int**[]** getServiceList**()** **{**

**return** serviceList**;**

**}**

public int**[]** getInterDepartureList**()** **{**

**return** interDepartureList**;**

**}**

public int**[]** getResponseTimeList**()** **{**

**return** responseTimeList**;**

**}**

public int**[]** getCycleTimeList**()** **{**

**return** cycleTimeList**;**

**}**

public ArrayList**<**Double**>** getInterArrival**()** **{**

**return** interArrival**;**

**}**

public ArrayList**<**Double**>** getInterDeparture**()** **{**

**return** interDeparture**;**

**}**

public ArrayList**<**Double**>** getBusyTime**()** **{**

**return** busyTime**;**

**}**

public ArrayList**<**Double**>** getResponseTime**()** **{**

**return** responseTime**;**

**}**

public HashMap**<**Integer**,** Double**>** getQueueMap**()** **{**

**return** queueMap**;**

**}**

public double getUtilization**()** **{**

**return** sumServiceTime **/** **(**numOfServers **\*** t**);**

**}**

public double getAvgInterArrivalTime**()** **{**

**return** sumInterArrivalTime **/** numOfArrivaledJobs**;**

**}**

public double getSDInterArrivalTime**()** **{**

**return** Math**.**sqrt**(**sdInterArrivalTime **/** numOfArrivaledJobs **-** Math**.**pow**(**getAvgInterArrivalTime**(),** 2**));**

**}**

public double getAvgCycleTime**()** **{**

**return** sumCycleTime **/** numOfServicedJobs**;**

**}**

public double getSDCycleTime**()** **{**

**return** Math**.**sqrt**(**sdCycleTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgCycleTime**(),** 2**));**

**}**

public double getAvgWaitTime**()** **{**

**return** sumWaitTime **/** numOfServicedJobs**;**

**}**

public double getSDWaitTime**()** **{**

**return** Math**.**sqrt**(**sdWaitTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgWaitTime**(),** 2**));**

**}**

public double getAvgResponseTime**()** **{**

**return** sumResponseTime **/** numOfServicedJobs**;**

**}**

public double getSDResponseTime**()** **{**

**return** Math**.**sqrt**(**sdResponseTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgResponseTime**(),** 2**));**

**}**

public double getAvgServiceTime**()** **{**

**return** sumServiceTime **/** numOfServicedJobs**;**

**}**

public double getSDServiceTime**()** **{**

**return** Math**.**sqrt**(**sdServiceTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgServiceTime**(),** 2**));**

**}**

public double getAvgInterDeparutreTime**()** **{**

**return** sumInterDepartureTime **/** numOfServicedJobs**;**

**}**

public double getSDInterDeparutreTime**()** **{**

**return** Math**.**sqrt**(**sdInterDepartureTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgInterDeparutreTime**(),** 2**));**

**}**

public double getAvgJobsIntheSystem**()** **{**

**return** sumQueueLength **/** t**;**

**}**

public double getSDJobsIntheSystem**()** **{**

**return** Math**.**sqrt**(**sdQueueLength **/** t **-** Math**.**pow**(**getAvgJobsIntheSystem**(),** 2**));**

**}**

**}**

/\*

\* InteractiveSystemSimuator.java

\*/

package gg1simulator**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**HashMap**;**

**import** java**.**util**.**LinkedList**;**

**import** java**.**util**.**PriorityQueue**;**

**import** java**.**util**.**Queue**;**

**import** java**.**util**.**Random**;**

public class InteractiveSystemSimuator **{**

protected int bars**;**

protected double interval**;**

protected Distribution thinkTimeDistribution**;**

protected Distribution serviceDistribution**;**

protected int numOfTerminals**,** numOfServers**;**

protected double t **=** 0**,** lastEventTime **=** 0**,** delta**;**

protected PriorityQueue**<**Event**>** eventQueue **=** **new** PriorityQueue**<>();**

protected int customers**;**

protected int midCustomers**;**

private int numOfArrivaledJobs **=** 0**,** numOfServicedJobs **=** 0**;**

private ArrayList**<**Double**>** interArrival **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** interDeparture **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** busyTime **=** **new** ArrayList**<>();**

private ArrayList**<**Double**>** responseTime **=** **new** ArrayList**<>();**

private HashMap**<**Integer**,** Double**>** queueMap **=** **new** HashMap**<>();**

private Queue**<**Jobs**>** jobsQueue **=** **new** LinkedList**<>();**

private boolean**[]** isTerminalBusy**,** isServerBusy**;**

private double maxInterArrivalTime **=** 0**,** maxServiceTime **=** 0**,** maxInterDepartureTime **=** 0**,**

maxResponseTime **=** 0**;**

private double sumInterArrivalTime**,** sumServiceTime**,** sumInterDepartureTime**,**

sumResponseTime**,** sumQueueLength**,** sumBusyTime**,** sumCycleTime**,** sumWaitTime**;**

private double sdInterArrivalTime**,** sdServiceTime**,** sdInterDepartureTime**,**

sdResponseTime**,** sdQueueLength**,** sdCycleTime**,** sdWaitTime**;**

private int**[]** interArrivalList**,** serviceList**,** interDepartureList**,** responseTimeList**,** cycleTimeList**;**

public InteractiveSystemSimuator**(**Distribution thinkTimeDistribution**,** Distribution serviceDistribution**,** int numOfTerminals**,** int numOfServers**,** int bars**,** double interval**)** **{**

**this.**thinkTimeDistribution **=** thinkTimeDistribution**;**

**this.**serviceDistribution **=** serviceDistribution**;**

**this.**numOfTerminals **=** numOfTerminals**;**

**this.**numOfServers **=** numOfServers**;**

**this.**bars **=** bars**;**

**this.**interval **=** interval**;**

initArray**();**

**}**

public int getNumOfServers**()** **{**

**return** numOfServers**;**

**}**

public final void initArray**()** **{**

interArrivalList **=** **new** int**[**bars**];**

serviceList **=** **new** int**[**bars**];**

interDepartureList **=** **new** int**[**bars**];**

responseTimeList **=** **new** int**[**bars**];**

cycleTimeList **=** **new** int**[**bars**];**

**for** **(**int i **=** 0**;** i **<** bars**;** i**++)** **{**

interArrivalList**[**i**]** **=** 0**;**

serviceList**[**i**]** **=** 0**;**

interDepartureList**[**i**]** **=** 0**;**

responseTimeList**[**i**]** **=** 0**;**

cycleTimeList**[**i**]** **=** 0**;**

**}**

isTerminalBusy **=** **new** boolean**[**numOfTerminals**];**

isServerBusy **=** **new** boolean**[**numOfServers**];**

**for** **(**int i **=** 0**;** i **<** numOfServers**;** i**++)** **{**

isServerBusy**[**i**]** **=** **false;**

**}**

**}**

public int getAFreeServer**()** **{**

Random r **=** **new** Random**();**

int num**;**

**do** **{**

num **=** r**.**nextInt**(**numOfServers**);**

**}** **while** **(**isServerBusy**[**num**]);**

**return** num**;**

**}**

public int getAFreeTerminal**()** **{**

Random r **=** **new** Random**();**

int num**;**

**do** **{**

num **=** r**.**nextInt**(**numOfTerminals**);**

**}** **while** **(**isTerminalBusy**[**num**]);**

**return** num**;**

**}**

public int getLength**()** **{**

**return** customers**;**

**}**

public double getDelta**()** **{**

**return** delta**;**

**}**

public int getBars**()** **{**

**return** bars**;**

**}**

public void run**(**int servicedJobs**)** **{**

double arrivalTime**,** serviceTime**,** rTime**,** cTime**,** wTime**;**

double tempArrival **=** 0**,** tempDeparture **=** 0**;**

int index**;**

**for** **(**int i **=** 0**;** i **<** numOfTerminals**;** i**++)** **{**

eventQueue**.**add**(new** Event**(**i**,** t**,** t **+** thinkTimeDistribution**.**generateRandomTimes**(),** Event**.**ARRIVAL**,** i**));**

isTerminalBusy**[**i**]** **=** **true;**

**}**

**while** **(**numOfServicedJobs **<** servicedJobs**)** **{**

Event e **=** eventQueue**.**poll**();**

t **=** e**.**endsTime**;** //current simulation time

delta **=** t **-** lastEventTime**;** //

sumQueueLength **+=** delta **\*** customers**;**

sdQueueLength **+=** delta **\*** customers **\*** customers**;**

lastEventTime **=** t**;**

Double length **=** queueMap**.**get**(**customers**);**

**if** **(**length **==** **null)** **{**

length **=** 0.0**;**

**}**

queueMap**.**put**(**customers**,** length **+** delta**);**

**switch** **(**e**.**code**)** **{**

**case** Event**.**ARRIVAL**:**

customers**++;**

numOfArrivaledJobs**++;**

isTerminalBusy**[**e**.**id**]** **=** **false;**

Jobs jobA **=** **new** Jobs**(**e**.**id**,** e**.**startsTime**,** t**,** t**);**

jobsQueue**.**add**(**jobA**);**

sumInterArrivalTime **+=** t **-** tempArrival**;**

sdInterArrivalTime **+=** **(**t **-** tempArrival**)** **\*** **(**t **-** tempArrival**);**

index **=** **(**int**)** **((**t **-** tempArrival**)** **/** interval**);**

**if** **(**index **<** bars**)** **{**

interArrivalList**[**index**]++;**

**}**

tempArrival **=** t**;**

**if** **(**customers **<=** numOfServers**)** **{**

serviceTime **=** serviceDistribution**.**generateRandomTimes**();**

index **=** **(**int**)** **(**serviceTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

serviceList**[**index**]++;**

**}**

sumServiceTime **+=** serviceTime**;**

sdServiceTime **+=** serviceTime **\*** serviceTime**;**

int branch **=** getAFreeServer**();**

eventQueue**.**add**(new** Event**(**jobA**.**getJobId**(),** t**,** t **+** serviceTime**,** Event**.**DEPARTURE**,** branch **+** 1**));**

isServerBusy**[**branch**]** **=** **true;**

**}**

**break;**

**case** Event**.**DEPARTURE**:**

customers**--;**

numOfServicedJobs**++;**

eventQueue**.**add**(new** Event**(**e**.**id**,** t**,** t **+** thinkTimeDistribution**.**generateRandomTimes**(),** Event**.**ARRIVAL**,** 1**));**

isServerBusy**[**e**.**branch **-** 1**]** **=** **false;**

tempDeparture **=** t**;**

Jobs jobD **=** jobsQueue**.**poll**();**

jobD**.**setDepartureTime**(**t**);**

rTime **=** jobD**.**getDepartureTime**()** **-** jobD**.**getLastArrivalTime**();**

cTime **=** jobD**.**getDepartureTime**()** **-** jobD**.**getStartTime**();**

wTime **=** rTime **-** **(**e**.**getEndsTime**()** **-** e**.**getStartsTime**());**

sumCycleTime **+=** cTime**;**

sdCycleTime **+=** cTime **\*** cTime**;**

index **=** **(**int**)** **(**cTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

cycleTimeList**[**index**]++;**

**}**

sumWaitTime **+=** wTime**;**

sdWaitTime **+=** wTime **\*** wTime**;**

//responseTime.add(rTime);

sumResponseTime **+=** rTime**;**

sdResponseTime **+=** rTime **\*** rTime**;**

index **=** **(**int**)** **(**rTime **/** interval**);**

**if** **(**index **<** bars**)** **{**

responseTimeList**[**index**]++;**

**}**

**if** **(**customers **>** numOfServers **-** 1**)** **{**

serviceTime **=** serviceDistribution**.**generateRandomTimes**();**

index **=** **(**int**)** **(**serviceTime **\*** bars **/** maxServiceTime**);**

**if** **(**index **<** bars**)** **{**

serviceList**[**index**]++;**

**}**

sumServiceTime **+=** serviceTime**;**

sdServiceTime **+=** serviceTime **\*** serviceTime**;**

int id **=** jobsQueue**.**peek**().**getJobId**();**

eventQueue**.**add**(new** Event**(**id**,** t**,** t **+** serviceTime**,** Event**.**DEPARTURE**,** e**.**branch**));**

**}**

**break;**

**}**

**}**

**}**

public double getMaxInterArrivalTime**()** **{**

**return** maxInterArrivalTime**;**

**}**

public double getMaxInterDepartureTime**()** **{**

**return** maxInterDepartureTime**;**

**}**

public double getMaxServiceTime**()** **{**

**return** maxServiceTime**;**

**}**

public double getMaxResponseTime**()** **{**

**return** maxResponseTime**;**

**}**

public int getArrivals**(){**

**return** numOfArrivaledJobs**;**

**}**

public int getDepartures**(){**

**return** numOfServicedJobs**;**

**}**

public int**[]** getInterArrivalList**()** **{**

**return** interArrivalList**;**

**}**

public int**[]** getServiceList**()** **{**

**return** serviceList**;**

**}**

public int**[]** getInterDepartureList**()** **{**

**return** interDepartureList**;**

**}**

public int**[]** getResponseTimeList**()** **{**

**return** responseTimeList**;**

**}**

public int**[]** getCycleTimeList**(){**

**return** cycleTimeList**;**

**}**

public ArrayList**<**Double**>** getInterArrival**()** **{**

**return** interArrival**;**

**}**

public ArrayList**<**Double**>** getInterDeparture**()** **{**

**return** interDeparture**;**

**}**

public ArrayList**<**Double**>** getBusyTime**()** **{**

**return** busyTime**;**

**}**

public ArrayList**<**Double**>** getResponseTime**()** **{**

**return** responseTime**;**

**}**

public HashMap**<**Integer**,** Double**>** getQueueMap**()** **{**

**return** queueMap**;**

**}**

public double getUtilization**()** **{**

**return** sumServiceTime **/** **(**numOfServers **\*** t**);**

**}**

public double getAvgInterArrivalTime**()** **{**

**return** sumInterArrivalTime **/** numOfArrivaledJobs**;**

**}**

public double getSDInterArrivalTime**()** **{**

**return** Math**.**sqrt**(**sdInterArrivalTime **/** numOfArrivaledJobs **-** Math**.**pow**(**getAvgInterArrivalTime**(),** 2**));**

**}**

public double getAvgCycleTime**()** **{**

**return** sumCycleTime **/** numOfServicedJobs**;**

**}**

public double getSDCycleTime**()** **{**

**return** Math**.**sqrt**(**sdCycleTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgCycleTime**(),** 2**));**

**}**

public double getAvgWaitTime**()** **{**

**return** sumWaitTime **/** numOfServicedJobs**;**

**}**

public double getSDWaitTime**()** **{**

**return** sumWaitTime **/** numOfServicedJobs**;**

**}**

public double getAvgResponseTime**()** **{**

**return** sumResponseTime **/** numOfServicedJobs**;**

**}**

public double getSDResponseTime**()** **{**

**return** Math**.**sqrt**(**sdResponseTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgResponseTime**(),** 2**));**

**}**

public double getAvgServiceTime**()** **{**

**return** sumServiceTime **/** numOfServicedJobs**;**

**}**

public double getSDServiceTime**()** **{**

**return** Math**.**sqrt**(**sdServiceTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgServiceTime**(),** 2**));**

**}**

public double getAvgInterDeparutreTime**()** **{**

**return** sumInterDepartureTime **/** numOfServicedJobs**;**

**}**

public double getSDInterDeparutreTime**()** **{**

**return** Math**.**sqrt**(**sdInterDepartureTime **/** numOfServicedJobs **-** Math**.**pow**(**getAvgInterDeparutreTime**(),** 2**));**

**}**

public double getAvgJobsIntheSystem**()** **{**

**return** sumQueueLength **/** t**;**

**}**

public double getSDJobsIntheSystem**()** **{**

**return** Math**.**sqrt**(**sdQueueLength **/** t **-** Math**.**pow**(**getAvgJobsIntheSystem**(),** 2**));**

**}**

**}**

/\*

\* Simulator.java

\*/

package gg1simulator**.**GUI**;**

**import** gg1simulator**.**DeterministicDistribution**;**

**import** gg1simulator**.**Distribution**;**

**import** gg1simulator**.**ExponentialDistribution**;**

**import** gg1simulator**.**PlotGenerator**;**

**import** gg1simulator**.**UniformDistribution**;**

**import** gg1simulator**.**Utility**;**

**import** java**.**awt**.**BorderLayout**;**

**import** java**.**awt**.**Dimension**;**

**import** java**.**awt**.**FlowLayout**;**

**import** java**.**awt**.**GridLayout**;**

**import** java**.**awt**.**Image**;**

**import** java**.**awt**.**Toolkit**;**

**import** java**.**awt**.**event**.**ItemEvent**;**

**import** java**.**awt**.**event**.**ItemListener**;**

**import** java**.**net**.**MalformedURLException**;**

**import** java**.**net**.**URL**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**HashMap**;**

**import** java**.**util**.**logging**.**Level**;**

**import** java**.**util**.**logging**.**Logger**;**

**import** javax**.**swing**.**BorderFactory**;**

**import** javax**.**swing**.**BoxLayout**;**

**import** javax**.**swing**.**ImageIcon**;**

**import** javax**.**swing**.**JApplet**;**

**import** javax**.**swing**.**JComboBox**;**

**import** javax**.**swing**.**JComponent**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JScrollPane**;**

**import** javax**.**swing**.**JTextField**;**

**import** org**.**jfree**.**chart**.**ChartPanel**;**

**import** org**.**jfree**.**chart**.**JFreeChart**;**

public abstract class Simulator **extends** JApplet **{**

private HashMap**<**String**,** JComponent**>** coltrolsMap **=** **new** HashMap**<>();**

private String M **=** "Exponential (M)"**;**

private String G **=** "Uniform (G)"**;**

private String D **=** "Deterministic (D)"**;**

private String**[]** distributions **=** **new** String**[]{**M**,** G**,** D**};**

private String**[]** numbers **=** **new** String**[]{**"10"**,** "15"**,** "20"**};**

private int m\_width **=** 1200**;**

private int m\_height **=** 700**;**

private static final String STEPS **=** "Steps"**;**

private static final String ARRIVAL **=** "Arrival"**;**

private static final String SERVICEONE **=** "ServiceOne"**;**

private static final String SERVICETWO **=** "ServiceTwo"**;**

private static final String SERVICETHREE **=** "ServiceThree"**;**

private static final String SERVICEFOUR **=** "ServiceFour"**;**

private static final String SERVICEFIVE **=** "ServiceFive"**;**

private static final String LAMBDA **=** "Lambda"**;**

private static final String CONSTANT **=** "Constant"**;**

private static final String UNIFORMLOWER **=** "Uniform lower"**;**

private static final String UNIFORMUPPER **=** "Uniform upper"**;**

private static final String PRECENTAGE **=** "Precentage"**;**

private static final String NUMBEROFBARS **=** "bars"**;**

private static final String INTERVAL **=** "Interval"**;**

private static final String P **=** "p"**;**

private static final String P1 **=** "p1"**;**

private static final String P2 **=** "p2"**;**

private static final String P3 **=** "p3"**;**

private static final String P4 **=** "p4"**;**

private static final String NUMOFTERMINALS **=** "numOfTerminals"**;**

private static final String NUMOFSERVERS **=** "numOfServers"**;**

private String imageName **=** **null;**

private Image bgImage**;**

protected JPanel mainResultPanel**;**

protected JScrollPane resultScrollPanel**;**

private int numOfServers**;**

public int modelType**;** // 0: non feedback model; 1: feedback model

public void setNumOfServers**(**int servers**)** **{**

numOfServers **=** servers**;**

**}**

public void setModelType**(**int type**)** **{**

modelType **=** type**;**

**}**

public void setImageName**(**String name**)** **{**

imageName **=** name**;**

**}**

public abstract JPanel createSimButtons**();**

public abstract JPanel createChartPanel**();**

public abstract JPanel createServiceDistributionPanel**();**

public abstract boolean checkErrors**();**

public static String getARRIVAL**()** **{**

**return** ARRIVAL**;**

**}**

public static String getSERVICEONE**()** **{**

**return** SERVICEONE**;**

**}**

public static String getSERVICETWO**()** **{**

**return** SERVICETWO**;**

**}**

public static String getSERVICETHREE**()** **{**

**return** SERVICETHREE**;**

**}**

public static String getSERVICEFOUR**()** **{**

**return** SERVICEFOUR**;**

**}**

public static String getSERVICEFIVE**()** **{**

**return** SERVICEFIVE**;**

**}**

public static String getCONSTANT**()** **{**

**return** CONSTANT**;**

**}**

public HashMap**<**String**,** JComponent**>** getColtrolsMap**()** **{**

**return** coltrolsMap**;**

**}**

public static String getSTEPS**()** **{**

**return** STEPS**;**

**}**

public static String getLAMBDA**()** **{**

**return** LAMBDA**;**

**}**

public static String getUNIFORMLOWER**()** **{**

**return** UNIFORMLOWER**;**

**}**

public static String getUNIFORMUPPER**()** **{**

**return** UNIFORMUPPER**;**

**}**

public static String getPRECENTAGE**()** **{**

**return** PRECENTAGE**;**

**}**

public static String getNUMBEROFBARS**()** **{**

**return** NUMBEROFBARS**;**

**}**

public static String getP**()** **{**

**return** P**;**

**}**

public static String getP1**()** **{**

**return** P1**;**

**}**

public static String getP2**()** **{**

**return** P2**;**

**}**

public static String getP3**()** **{**

**return** P3**;**

**}**

public static String getP4**()** **{**

**return** P4**;**

**}**

public static String getInterval**()** **{**

**return** INTERVAL**;**

**}**

public static String getNumOfTerminals**()** **{**

**return** NUMOFTERMINALS**;**

**}**

public static String getNumOfServers**()** **{**

**return** NUMOFSERVERS**;**

**}**

@Override

public void init**()** **{**

**this.**setSize**(**m\_width**,** m\_height**);**

Dimension screenSize **=** Toolkit**.**getDefaultToolkit**().**getScreenSize**();**

**this.**setLocation**((**screenSize**.**width **-** m\_width**)** **/** 2**,** **(**screenSize**.**height **-** m\_height**)** **/** 2**);**

**try** **{**

javax**.**swing**.**UIManager**.**setLookAndFeel**(**javax**.**swing**.**UIManager**.**getSystemLookAndFeelClassName**());**

**}** **catch** **(**ClassNotFoundException **|** InstantiationException **|** IllegalAccessException **|** javax**.**swing**.**UnsupportedLookAndFeelException ex**)** **{**

java**.**util**.**logging**.**Logger**.**getLogger**(**Simulator**.**class**.**getName**()).**log**(**java**.**util**.**logging**.**Level**.**SEVERE**,** **null,** ex**);**

**}**

JPanel mainPanel **=** **new** JPanel**(new** GridLayout**(**1**,** 2**,** 10**,** 10**));**

JPanel leftPanel **=** **new** JPanel**();**

leftPanel**.**setLayout**(new** BoxLayout**(**leftPanel**,** BoxLayout**.**Y\_AXIS**));**

JPanel leftUpPanel **=** **new** JPanel**(new** BorderLayout**());**

leftUpPanel**.**setBorder**(**BorderFactory**.**createTitledBorder**(**"Simulatoion controls"**));**

JPanel distributionPanel **=** **new** JPanel**();**

distributionPanel**.**setLayout**(new** BoxLayout**(**distributionPanel**,** BoxLayout**.**Y\_AXIS**));**

JPanel imagePanel **=** **new** JPanel**();**

imagePanel**.**setBorder**(**BorderFactory**.**createCompoundBorder**());**

**try** **{**

bgImage **=** getImage**(new** URL**(**getCodeBase**(),** imageName**));**

**}** **catch** **(**MalformedURLException ex**)** **{**

Logger**.**getLogger**(**Simulator**.**class**.**getName**()).**log**(**Level**.**SEVERE**,** **null,** ex**);**

**}**

JLabel imageLabel **=** **new** JLabel**(new** ImageIcon**(**bgImage**));**

imageLabel**.**repaint**();**

imagePanel**.**add**(**imageLabel**);**

distributionPanel**.**add**(**imagePanel**);**

JPanel placePanel **=** **new** JPanel**(new** GridLayout**(**0**,** 1**));**

placePanel**.**setBorder**(**BorderFactory**.**createTitledBorder**(**"Simulation Parameters"**));**

JPanel simParametersPanel**;**

simParametersPanel **=** **new** JPanel**(new** GridLayout**(**0**,** 3**));**

simParametersPanel**.**add**(**createSimulationParameters**());**

simParametersPanel**.**add**(**createBarsControlPanel**());**

simParametersPanel**.**add**(**createBarsInterval**());**

placePanel**.**add**(**simParametersPanel**);**

**if** **(**modelType **==** 1**)** **{**

JPanel simParameterPanel2 **=** **new** JPanel**(new** GridLayout**(**0**,** 1**));**

simParameterPanel2**.**add**(**createPrecentagePanel**(**getP**()));**

placePanel**.**add**(**simParameterPanel2**);**

**}**

**if** **(**modelType **==** 3**)** **{**

JPanel simParameterPanel2 **=** **new** JPanel**(new** GridLayout**(**0**,** 2**));**

simParameterPanel2**.**add**(**createTerminalsPanel**());**

simParameterPanel2**.**add**(**createServersPanel**());**

placePanel**.**add**(**simParameterPanel2**);**

**}**

**if** **(**modelType **==** 4**)** **{**

JPanel simParameterPanel2 **=** **new** JPanel**(new** GridLayout**(**0**,** 1**));**

simParameterPanel2**.**add**(**createServersPanel**());**

placePanel**.**add**(**simParameterPanel2**);**

**}**

distributionPanel**.**add**(**placePanel**);**

JPanel arrivalDistributionPanel **=** **new** JPanel**(new** GridLayout**(**0**,** 1**));**

String title **=** "Arrival Distribution (IAT = Interarrival Time)"**;**

**if** **(**modelType **==** 3**)** **{**

title **=** "Think Time Distribution (TT = Think Time)"**;**

**}**

arrivalDistributionPanel**.**setBorder**(**BorderFactory**.**createTitledBorder**(**title**));**

arrivalDistributionPanel**.**add**(**createDistributionControl**(**getARRIVAL**(),** ""**,** 0**));**

distributionPanel**.**add**(**arrivalDistributionPanel**);**

distributionPanel**.**add**(**createServiceDistributionPanel**());**

leftUpPanel**.**add**(**distributionPanel**,** BorderLayout**.**CENTER**);**

leftUpPanel**.**add**(**createSimButtons**(),** BorderLayout**.**SOUTH**);**

leftPanel**.**add**(**leftUpPanel**);**

leftPanel**.**add**(**createResultPanel**());**

mainPanel**.**add**(**leftPanel**);**

mainPanel**.**add**(**createChartPanel**());**

**this.**setContentPane**(**mainPanel**);**

**((**JTextField**)** coltrolsMap**.**get**(**STEPS**)).**setText**(**"100000"**);**

**((**JTextField**)** coltrolsMap**.**get**(**ARRIVAL **+** "-" **+** LAMBDA**)).**setText**(**"2"**);**

**((**JTextField**)** coltrolsMap**.**get**(**SERVICEONE **+** "-" **+** LAMBDA**)).**setText**(**"1"**);**

**((**JTextField**)** coltrolsMap**.**get**(**SERVICEONE **+** "-" **+** LAMBDA**)).**setText**(**"1"**);**

**((**JTextField**)** coltrolsMap**.**get**(**INTERVAL**)).**setText**(**"1"**);**

**this.**setVisible**(true);**

**}**

public JPanel createResultPanel**()** **{**

JPanel resultPanel **=** **new** JPanel**();**

resultPanel**.**setLayout**(new** BoxLayout**(**resultPanel**,** BoxLayout**.**Y\_AXIS**));**

resultPanel**.**setPreferredSize**(new** Dimension**(**550**,** 400**));**

String title**;**

**if(**modelType **!=** 2**){**

title **=** "Simulation Results (Values in brackets are analytic results for exponential model)"**;**

**}else{**

title **=** "Simulation Results (Values in brackets are analytic results)"**;**

**}**

resultPanel**.**setBorder**(**BorderFactory**.**createTitledBorder**(**title**));**

mainResultPanel **=** **new** JPanel**(new** GridLayout**(**0**,** 1**));**

resultScrollPanel **=** **new** JScrollPane**(**mainResultPanel**,** JScrollPane**.**VERTICAL\_SCROLLBAR\_ALWAYS**,**

JScrollPane**.**HORIZONTAL\_SCROLLBAR\_NEVER**);**

resultScrollPanel**.**getVerticalScrollBar**().**setUnitIncrement**(**16**);**

resultPanel**.**add**(**resultScrollPanel**);**

**return** resultPanel**;**

**}**

public JPanel createSimulationParameters**()** **{**

JPanel simParameters **=** **new** JPanel**(new** FlowLayout**(**FlowLayout**.**LEFT**));**

simParameters**.**add**(new** JLabel**(**"Served jobs: "**));**

JTextField txtStep **=** **new** JTextField**();**

txtStep**.**setPreferredSize**(new** Dimension**(**50**,** 24**));**

simParameters**.**add**(**txtStep**);**

coltrolsMap**.**put**(**STEPS**,** txtStep**);**

**return** simParameters**;**

**}**

public JPanel createPrecentagePanel**(**String name**)** **{**

JPanel precentPanel **=** **new** JPanel**(new** FlowLayout**(**FlowLayout**.**LEFT**));**

precentPanel**.**add**(new** JLabel**(**name **+** ": "**));**

JTextField txtPrecent **=** **new** JTextField**();**

txtPrecent**.**setPreferredSize**(new** Dimension**(**50**,** 24**));**

precentPanel**.**add**(**txtPrecent**);**

coltrolsMap**.**put**(**name**,** txtPrecent**);**

**return** precentPanel**;**

**}**

public JPanel createDistributionControl**(**final String name**,** final String number**,** final int type**)** **{**

final JPanel disPanel **=** **new** JPanel**(new** FlowLayout**(**FlowLayout**.**LEFT**));**

String title**;**

**if(**name**.**contains**(**"Arrival"**)){**

title **=** "Type:"**;**

**}else{**

title **=** "Server " **+** number **+** "Type:"**;**

**}**

disPanel**.**add**(new** JLabel**(**title**));**

JComboBox distribution **=** **new** JComboBox**(**distributions**);**

distribution**.**setSelectedIndex**(**0**);**

distribution**.**addItemListener**(new** ItemListener**()** **{**

@Override

public void itemStateChanged**(**ItemEvent e**)** **{**

**if** **(**e**.**getStateChange**()** **==** ItemEvent**.**SELECTED**)** **{**

**if** **(**e**.**getItem**().**equals**(**M**))** **{**

disPanel**.**add**(**createExponentialFields**(**name**,** number**,** type**));**

**}** **else** **if** **(**e**.**getItem**().**equals**(**G**))** **{**

disPanel**.**add**(**createUniformFields**(**name**,** number**,** type**));**

**}** **else** **if** **(**e**.**getItem**().**equals**(**D**))** **{**

disPanel**.**add**(**createDeterministicFields**(**name**,** number**,** type**));**

**}**

**}** **else** **if** **(**e**.**getStateChange**()** **==** ItemEvent**.**DESELECTED**)** **{**

disPanel**.**remove**(**2**);**

**}**

disPanel**.**validate**();**

disPanel**.**repaint**();**

**}**

**});**

coltrolsMap**.**put**(**name**,** distribution**);**

disPanel**.**add**(**distribution**);**

disPanel**.**add**(**createExponentialFields**(**name**,** number**,** type**));**

**return** disPanel**;**

**}**

public JPanel createExponentialFields**(**String name**,** String number**,** int type**)** **{**

JPanel eFieldPanel **=** **new** JPanel**(new** FlowLayout**(**FlowLayout**.**LEFT**));**

// if (type == 0 && !number.equals("")) {

// int counter = Integer.parseInt(number.substring(1, 2));

// if (counter != numOfServers) {

// if (number.equals("#1 ")) {

// eFieldPanel.add(createPrecentagePanel(getP1()));

// }

// if (number.equals("#2 ")) {

// eFieldPanel.add(createPrecentagePanel(getP2()));

// }

// if (number.equals("#3 ")) {

// eFieldPanel.add(createPrecentagePanel(getP3()));

// }

// }

// }

String labelname**;**

**if** **(**name**.**contains**(**"Arrival"**))** **{**

**if** **(**modelType **==** 3**)** **{**

labelname **=** "Mean TT:"**;**

**}** **else** **{**

labelname **=** "Mean IAT:"**;**

**}**

**}** **else** **{**

labelname **=** "Mean ST:"**;**

**}**

JLabel eFieldLabel **=** **new** JLabel**(**labelname**);**

eFieldPanel**.**add**(**eFieldLabel**);**

JTextField eField **=** **new** JTextField**();**

eField**.**setPreferredSize**(new** Dimension**(**50**,** 24**));**

eFieldPanel**.**add**(**eField**);**

coltrolsMap**.**put**(**name **+** "-" **+** LAMBDA**,** eField**);**

**return** eFieldPanel**;**

**}**

public JPanel createUniformFields**(**String name**,** String number**,** int type**)** **{**

JPanel uFieldPanel **=** **new** JPanel**(new** FlowLayout**(**FlowLayout**.**LEFT**));**

// if (type == 0 && !number.equals("")) {

// int counter = Integer.parseInt(number.substring(1, 2));

// if (counter != numOfServers) {

// if (number.equals("#1 ")) {

// uFieldPanel.add(createPrecentagePanel(getP1()));

// }

// if (number.equals("#2 ")) {

// uFieldPanel.add(createPrecentagePanel(getP2()));

// }

// if (number.equals("#3 ")) {

// uFieldPanel.add(createPrecentagePanel(getP3()));

// }

// }

// }

String labelname**;**

**if** **(**name**.**contains**(**"Arrival"**))** **{**

**if** **(**modelType **==** 3**)** **{**

labelname **=** "TT:"**;**

**}** **else** **{**

labelname **=** "IAT:"**;**

**}**

**}** **else** **{**

labelname **=** "ST:"**;**

**}**

JLabel uFieldLowerLabel **=** **new** JLabel**(**"Min " **+** labelname**);**

uFieldPanel**.**add**(**uFieldLowerLabel**);**

JTextField uLowerBoundField **=** **new** JTextField**();**

uLowerBoundField**.**setPreferredSize**(new** Dimension**(**50**,** 24**));**

uFieldPanel**.**add**(**uLowerBoundField**);**

coltrolsMap**.**put**(**name **+** "-" **+** UNIFORMLOWER**,** uLowerBoundField**);**

JLabel uFieldUpperLabekl **=** **new** JLabel**(**"Max " **+** labelname**);**

uFieldPanel**.**add**(**uFieldUpperLabekl**);**

JTextField uUpperBoundField **=** **new** JTextField**();**

uUpperBoundField**.**setPreferredSize**(new** Dimension**(**50**,** 24**));**

uFieldPanel**.**add**(**uUpperBoundField**);**

coltrolsMap**.**put**(**name **+** "-" **+** UNIFORMUPPER**,** uUpperBoundField**);**

**return** uFieldPanel**;**

**}**

public JPanel createDeterministicFields**(**String name**,** String number**,** int type**)** **{**

JPanel dFieldPanel **=** **new** JPanel**(new** FlowLayout**(**FlowLayout**.**LEFT**));**

String labelname**;**

**if** **(**name**.**contains**(**"Arrival"**))** **{**

**if** **(**modelType **==** 3**)** **{**

labelname **=** "TT:"**;**

**}** **else** **{**

labelname **=** "IAT:"**;**

**}**

**}** **else** **{**

labelname **=** "ST:"**;**

**}**

JLabel dFieldLabel **=** **new** JLabel**(**"Constant " **+** labelname**);**

dFieldPanel**.**add**(**dFieldLabel**);**

JTextField dField **=** **new** JTextField**();**

dField**.**setPreferredSize**(new** Dimension**(**50**,** 24**));**

dFieldPanel**.**add**(**dField**);**

coltrolsMap**.**put**(**name **+** "-" **+** CONSTANT**,** dField**);**

**return** dFieldPanel**;**

**}**

public JPanel createBarsControlPanel**()** **{**

JPanel barsPanel **=** **new** JPanel**(new** FlowLayout**(**FlowLayout**.**LEFT**));**

JLabel barsLabel **=** **new** JLabel**(**"Number of bars: "**);**

barsPanel**.**add**(**barsLabel**);**

JComboBox numberOfBars **=** **new** JComboBox**(**numbers**);**

numberOfBars**.**setPreferredSize**(new** Dimension**(**50**,** 24**));**

numberOfBars**.**setSelectedIndex**(**1**);**

barsPanel**.**add**(**numberOfBars**);**

coltrolsMap**.**put**(**NUMBEROFBARS**,** numberOfBars**);**

**return** barsPanel**;**

**}**

public JPanel createBarsInterval**()** **{**

JPanel intervalPanel **=** **new** JPanel**(new** FlowLayout**(**FlowLayout**.**LEFT**));**

JLabel barsInterval **=** **new** JLabel**(**"Interval between bars: "**);**

intervalPanel**.**add**(**barsInterval**);**

JTextField dField **=** **new** JTextField**();**

dField**.**setPreferredSize**(new** Dimension**(**50**,** 24**));**

intervalPanel**.**add**(**dField**);**

coltrolsMap**.**put**(**INTERVAL**,** dField**);**

**return** intervalPanel**;**

**}**

public JPanel createTerminalsPanel**()** **{**

JPanel intervalPanel **=** **new** JPanel**(new** FlowLayout**(**FlowLayout**.**LEFT**));**

JLabel barsInterval **=** **new** JLabel**(**"Number of terminals: "**);**

intervalPanel**.**add**(**barsInterval**);**

JTextField dField **=** **new** JTextField**();**

dField**.**setPreferredSize**(new** Dimension**(**50**,** 24**));**

intervalPanel**.**add**(**dField**);**

coltrolsMap**.**put**(**NUMOFTERMINALS**,** dField**);**

**return** intervalPanel**;**

**}**

public JPanel createServersPanel**()** **{**

JPanel intervalPanel **=** **new** JPanel**(new** FlowLayout**(**FlowLayout**.**LEFT**));**

JLabel barsInterval **=** **new** JLabel**(**"Number of servers: "**);**

intervalPanel**.**add**(**barsInterval**);**

JTextField dField **=** **new** JTextField**();**

dField**.**setPreferredSize**(new** Dimension**(**50**,** 24**));**

intervalPanel**.**add**(**dField**);**

**if** **(**modelType **==** 4**)** **{**

JLabel note **=** **new** JLabel**(**"(Must be less than 16)"**);**

intervalPanel**.**add**(**note**);**

**}**

coltrolsMap**.**put**(**NUMOFSERVERS**,** dField**);**

**return** intervalPanel**;**

**}**

public JPanel createTitle**()** **{**

JPanel titlePanel **=** **new** JPanel**(new** GridLayout**(**1**,** 5**));**

titlePanel**.**add**(new** JLabel**(**" Indictor"**));**

titlePanel**.**add**(new** JLabel**(**"Mean value (̩"**));**

titlePanel**.**add**(new** JLabel**(**"Error [%]"**));**

titlePanel**.**add**(new** JLabel**(**"SD (ҩ"**));**

titlePanel**.**add**(new** JLabel**(**"CV (v=100\*̯ҩ"**));**

**return** titlePanel**;**

**}**

public JPanel createRow**(**String name**,** ArrayList**<**Double**>** values**)** **{**

JPanel row **=** **new** JPanel**(new** GridLayout**(**1**,** 4**));**

row**.**add**(new** JLabel**(**" " **+** name**));**

double average **=** Utility**.**getMean**(**values**);**

row**.**add**(new** JLabel**(**Utility**.**formatOutput**(**average**)));**

double stddev **=** Utility**.**getSD**(**values**);**

row**.**add**(new** JLabel**(**Utility**.**formatOutput**(**stddev**)));**

row**.**add**(new** JLabel**(**Utility**.**formatOutput**(**100 **\*** stddev **/** average**)));**

**return** row**;**

**}**

public JPanel createRow**(**String name**,** double avg**,** double sdv**,** String real**,** String error**)** **{**

double res**;**

JPanel row **=** **new** JPanel**(new** GridLayout**(**1**,** 4**));**

row**.**add**(new** JLabel**(**" " **+** name**));**

**if** **(!**real**.**equals**(**""**))** **{**

row**.**add**(new** JLabel**(**Utility**.**formatOutput**(**avg**)** **+** " [" **+** real **+** "]"**));**

row**.**add**(new** JLabel**(**error**));**

**}** **else** **{**

row**.**add**(new** JLabel**(**Utility**.**formatOutput**(**avg**)));**

row**.**add**(new** JLabel**(**""**));**

**}**

row**.**add**(new** JLabel**(**Utility**.**formatOutput**(**sdv**)));**

**if** **(**avg **==** 0**)** **{**

res **=** 0.000**;**

**}** **else** **{**

res **=** 100 **\*** sdv **/** avg**;**

**}**

row**.**add**(new** JLabel**(**Utility**.**formatOutput**(**res**)));**

**return** row**;**

**}**

public JPanel createRow**(**String name**,** double u**,** String real**,** String error**)** **{**

JPanel row **=** **new** JPanel**(new** GridLayout**(**1**,** 4**));**

row**.**add**(new** JLabel**(**" " **+** name**));**

**if** **(!**real**.**equals**(**""**))** **{**

row**.**add**(new** JLabel**(**Utility**.**formatOutput**(**u**)** **+** " [" **+** real **+** "]"**));**

row**.**add**(new** JLabel**(**error**));**

**}** **else** **{**

row**.**add**(new** JLabel**(**Utility**.**formatOutput**(**u**)));**

row**.**add**(new** JLabel**(**""**));**

**}**

row**.**add**(new** JLabel**(**""**));**

row**.**add**(new** JLabel**(**""**));**

**return** row**;**

**}**

public ChartPanel createChartPanel**(**int**[]** chartArray**,** double max**,** String name**,** int type**)** **{**

JFreeChart chart **=** PlotGenerator**.**generateHistogramChart**(**chartArray**,** max**,** name**,** type**);**

ChartPanel panel **=** **new** ChartPanel**(**chart**,** **false);**

panel**.**setPreferredSize**(new** Dimension**(**560**,** 350**));**

**return** panel**;**

**}**

public ChartPanel createChartPanel**(**int**[]** chartArray**,** int size**,** double interval**,** String name**,** int type**)** **{**

JFreeChart chart **=** PlotGenerator**.**generateHistogramChart2**(**chartArray**,** size**,** interval**,** name**,** type**);**

ChartPanel panel **=** **new** ChartPanel**(**chart**,** **false);**

panel**.**setPreferredSize**(new** Dimension**(**560**,** 350**));**

**return** panel**;**

**}**

public ChartPanel createChartPanel**(**HashMap**<**Integer**,** Double**>** map**,** String name**,** int bars**,** int type**)** **{**

JFreeChart chart **=** PlotGenerator**.**generateHistogram**(**map**,** name**,** bars**,** type**);**

ChartPanel panel **=** **new** ChartPanel**(**chart**,** **false);**

panel**.**setPreferredSize**(new** Dimension**(**560**,** 350**));**

**return** panel**;**

**}**

public ChartPanel createChartPanel**(**ArrayList list**,** int bars**,** String name**)** **{**

JFreeChart chart **=** PlotGenerator**.**generateHistogramChart1**(**list**,** bars**,** name**);**

ChartPanel panel **=** **new** ChartPanel**(**chart**,** **false);**

panel**.**setPreferredSize**(new** Dimension**(**560**,** 350**));**

**return** panel**;**

**}**

public double convertStringToDouble**(**String number**)** **{**

**try** **{**

**return** Double**.**parseDouble**(**number**);**

**}** **catch** **(**Exception ex**)** **{**

JOptionPane**.**showMessageDialog**(this,** number **+** " is not a valid number"**,** "Number Error"**,** JOptionPane**.**ERROR\_MESSAGE**);**

**throw** ex**;**

**}**

**}**

public Distribution getDistribution**(**String source**,** String type**)** **{**

Distribution distribution **=** **null;**

**if** **(**source**.**equals**(**M**))** **{**

double lamdba **=** convertStringToDouble**(((**JTextField**)** coltrolsMap**.**get**(**type **+** "-" **+** LAMBDA**)).**getText**());**

distribution **=** **new** ExponentialDistribution**(**lamdba**);**

**}** **else** **if** **(**source**.**equals**(**G**))** **{**

double lowerBound **=** convertStringToDouble**(((**JTextField**)** coltrolsMap**.**get**(**type **+** "-" **+** UNIFORMLOWER**)).**getText**());**

double upperBound **=** convertStringToDouble**(((**JTextField**)** coltrolsMap**.**get**(**type **+** "-" **+** UNIFORMUPPER**)).**getText**());**

distribution **=** **new** UniformDistribution**(**lowerBound**,** upperBound**);**

**}** **else** **if** **(**source**.**equals**(**D**))** **{**

double constant **=** convertStringToDouble**(((**JTextField**)** coltrolsMap**.**get**(**type **+** "-" **+** CONSTANT**)).**getText**());**

distribution **=** **new** DeterministicDistribution**(**constant**);**

**}**

**return** distribution**;**

**}**

**}**

/\*

\* SingleServerModel.java

\*/

package gg1simulator**.**GUI**;**

**import** gg1simulator**.**Calculator**;**

**import** gg1simulator**.**Distribution**;**

**import** gg1simulator**.**GG1Calculator**;**

**import** gg1simulator**.**GG1Simulator**;**

**import** gg1simulator**.**MD1Calculator**;**

**import** gg1simulator**.**MG1Calculator**;**

**import** gg1simulator**.**MM1Calculator**;**

**import** gg1simulator**.**UniformDistribution**;**

**import** gg1simulator**.**Utility**;**

**import** java**.**awt**.**BorderLayout**;**

**import** java**.**awt**.**Dimension**;**

**import** java**.**awt**.**GridLayout**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** java**.**util**.**HashMap**;**

**import** javax**.**swing**.**BorderFactory**;**

**import** javax**.**swing**.**BoxLayout**;**

**import** javax**.**swing**.**JButton**;**

**import** javax**.**swing**.**JComboBox**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JScrollPane**;**

**import** javax**.**swing**.**JTextField**;**

public class SingleServerModel **extends** Simulator **{**

private JPanel myChartPanel**;**

private JScrollPane mainChartPanel**;**

private boolean flag **=** **false;**

private int preArrivals**,** preDepartures**;**

private double preInterval**;**

private int**[]** preInterArrivalTime**,** preServiceTime**,** preInterDepartureTime**,** preResponseTime**;**

private HashMap**<**Integer**,** Double**>** preQueueMap**;**

@Override

public void init**()** **{**

**super.**setModelType**(**2**);**

**super.**setImageName**(**"SingleServer.jpg"**);**

**super.**setNumOfServers**(**1**);**

**super.**init**();**

**}**

@Override

public JPanel createSimButtons**()** **{**

JPanel btnPanel **=** **new** JPanel**(new** BorderLayout**());**

JLabel msgLabel **=** **new** JLabel**();**

msgLabel**.**setPreferredSize**(new** Dimension**(**580**,** 18**));**

JPanel buttonPanel **=** **new** JPanel**();**

JButton btnSim **=** **new** JButton**(**"Run Simulator"**);**

btnSim**.**setPreferredSize**(new** Dimension**(**100**,** 24**));**

btnSim**.**addActionListener**(new** ActionListener**()** **{**

@Override

public void actionPerformed**(**ActionEvent e**)** **{**

System**.**out**.**println**(**flag**);**

**if** **(**checkErrors**())** **{**

**return;**

**}**

int steps**,** bars**;**

double interval**;**

String arrivalStr**,** serviceOneStr**;**

Distribution arrivalDistribution**,** serviceOneDistribution**;**

steps **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getSTEPS**())).**getText**());**

bars **=** Integer**.**parseInt**(((**JComboBox**)** getColtrolsMap**().**get**(**getNUMBEROFBARS**())).**getSelectedItem**().**toString**());**

arrivalStr **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getARRIVAL**())).**getSelectedItem**().**toString**();**

serviceOneStr **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getSERVICEONE**())).**getSelectedItem**().**toString**();**

interval **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getInterval**())).**getText**());**

arrivalDistribution **=** getDistribution**(**arrivalStr**,** getARRIVAL**());**

serviceOneDistribution **=** getDistribution**(**serviceOneStr**,** getSERVICEONE**());**

GG1Simulator simulator **=** **new** GG1Simulator**(**arrivalDistribution**,** serviceOneDistribution**,** bars**,** interval**);**

Calculator c **=** **new** MM1Calculator**(**arrivalDistribution**.**getMean**(),** serviceOneDistribution**.**getMean**());;**

int temp **=** 1**;**

**if** **(**arrivalStr**.**equals**(**"Exponential (M)"**))** **{**

temp **=** 0**;**

**switch** **(**serviceOneStr**)** **{**

**case** "Exponential (M)"**:**

c **=** **new** MM1Calculator**(**arrivalDistribution**.**getMean**(),** serviceOneDistribution**.**getMean**());**

**break;**

**case** "Uniform (G)"**:**

c **=** **new** MG1Calculator**(**arrivalDistribution**.**getMean**(),** **((**UniformDistribution**)** serviceOneDistribution**).**getMin**(),** **((**UniformDistribution**)** serviceOneDistribution**).**getMax**());**

**break;**

**case** "Deterministic (D)"**:**

c **=** **new** MD1Calculator**(**arrivalDistribution**.**getMean**(),** serviceOneDistribution**.**getMean**());**

**break;**

**}**

**}**

**if** **(**arrivalStr**.**equals**(**"Uniform (G)"**)** **&&** serviceOneStr**.**equals**(**"Uniform (G)"**))** **{**

temp **=** 0**;**

c **=** **new** GG1Calculator**(((**UniformDistribution**)** arrivalDistribution**).**getMin**(),** **((**UniformDistribution**)** arrivalDistribution**).**getMax**(),**

**((**UniformDistribution**)** serviceOneDistribution**).**getMin**(),** **((**UniformDistribution**)** serviceOneDistribution**).**getMax**());**

**}**

simulator**.**run**(**steps**);**

mainResultPanel**.**removeAll**();**

mainResultPanel**.**add**(**createTitle**());**

double avg**,** sdv**,** realAvg**,** error**;**

String realValue**,** errorStr**;**

avg **=** simulator**.**getUtilization**();**

realAvg **=** c**.**getUtilization**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if(**temp **==** 1**){**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Server Utilization"**,** avg**,** realValue**,** errorStr**));**

avg **=** simulator**.**getAvgInterArrivalTime**();**

sdv **=** simulator**.**getSDInterArrivalTime**();**

realAvg **=** c**.**getArrivalMean**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if(**temp **==** 1**){**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Interarrival time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** simulator**.**getAvgServiceTime**();**

sdv **=** simulator**.**getSDServiceTime**();**

realAvg **=** c**.**getServiceMean**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if(**temp **==** 1**){**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Service time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** simulator**.**getAvgInterDeparutreTime**();**

sdv **=** simulator**.**getSDInterDeparutreTime**();**

realValue **=** ""**;**

mainResultPanel**.**add**(**createRow**(**"Interdeparture time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** 1 **/** avg**;**

realValue **=** ""**;**

mainResultPanel**.**add**(**createRow**(**"Throughput"**,** avg**,** realValue**,** errorStr**));**

avg **=** simulator**.**getAvgResponseTime**();**

sdv **=** simulator**.**getSDResponseTime**();**

realAvg **=** c**.**getResponseTime**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if(**temp **==** 1**){**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Response time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** Utility**.**getAverage**(**simulator**.**getQueueMap**());**

sdv **=** Utility**.**getStandardDeviation**(**simulator**.**getQueueMap**(),** avg**);**

realAvg **=** c**.**getQueueLength**();**

**if** **(**realAvg **==** 0**)** **{**

realValue **=** ""**;**

**}** **else** **{**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

**}**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if(**temp **==** 1**){**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Number of jobs"**,** avg**,** sdv**,** realValue**,** errorStr**));**

mainResultPanel**.**revalidate**();**

mainResultPanel**.**repaint**();**

myChartPanel**.**removeAll**();**

**if** **(**flag **==** **true)** **{**

myChartPanel**.**add**(**createChartPanel**(**preInterArrivalTime**,** preArrivals**,** preInterval**,** "Interarrival Time (Previous Exp)"**,** 0**));**

**}**

myChartPanel**.**add**(**createChartPanel**(**simulator**.**getInterArrivalList**(),** simulator**.**getArrivals**(),** interval**,** "Interarrival Time"**,** 1**));**

**if** **(**flag **==** **true)** **{**

myChartPanel**.**add**(**createChartPanel**(**preServiceTime**,** preDepartures**,** preInterval**,** "Service Time (Previous Exp)"**,** 0**));**

**}**

myChartPanel**.**add**(**createChartPanel**(**simulator**.**getServiceList**(),** simulator**.**getDepartures**(),** interval**,** "Service Time"**,** 1**));**

**if** **(**flag **==** **true)** **{**

myChartPanel**.**add**(**createChartPanel**(**preInterDepartureTime**,** preDepartures**,** preInterval**,** "Interdeparture Time (Previous Exp)"**,** 0**));**

**}**

myChartPanel**.**add**(**createChartPanel**(**simulator**.**getInterDepartureList**(),** simulator**.**getDepartures**(),** interval**,** "Interdeparture Time"**,** 1**));**

**if** **(**flag **==** **true)** **{**

myChartPanel**.**add**(**createChartPanel**(**preResponseTime**,** preDepartures**,** preInterval**,** "Response Time (Previous Exp)"**,** 0**));**

**}**

myChartPanel**.**add**(**createChartPanel**(**simulator**.**getResponseTimeList**(),** simulator**.**getDepartures**(),** interval**,** "Response Time"**,** 1**));**

**if** **(**flag **==** **true)** **{**

myChartPanel**.**add**(**createChartPanel**(**preQueueMap**,** "Number of jobs (Previous Exp)"**,** bars**,** 0**));**

**}**

preQueueMap **=** **(**HashMap**)** simulator**.**getQueueMap**().**clone**();**

myChartPanel**.**add**(**createChartPanel**(**simulator**.**getQueueMap**(),** "Number of jobs"**,** bars**,** 1**));**

myChartPanel**.**revalidate**();**

myChartPanel**.**repaint**();**

flag **=** **true;**

preInterArrivalTime **=** **new** int**[**bars**];**

System**.**arraycopy**(**simulator**.**getInterArrivalList**(),** 0**,** preInterArrivalTime**,** 0**,** bars**);**

preServiceTime **=** **new** int**[**bars**];**

System**.**arraycopy**(**simulator**.**getServiceList**(),** 0**,** preServiceTime**,** 0**,** bars**);**

preInterDepartureTime **=** **new** int**[**bars**];**

System**.**arraycopy**(**simulator**.**getInterDepartureList**(),** 0**,** preInterDepartureTime**,** 0**,** bars**);**

preResponseTime **=** **new** int**[**bars**];**

System**.**arraycopy**(**simulator**.**getResponseTimeList**(),** 0**,** preResponseTime**,** 0**,** bars**);**

preArrivals **=** simulator**.**getArrivals**();**

preDepartures **=** simulator**.**getDepartures**();**

preInterval **=** interval**;**

**}**

**});**

buttonPanel**.**add**(**btnSim**);**

btnPanel**.**add**(**msgLabel**,** BorderLayout**.**NORTH**);**

btnPanel**.**add**(**buttonPanel**,** BorderLayout**.**CENTER**);**

**return** btnPanel**;**

**}**

@Override

public JPanel createChartPanel**()** **{**

JPanel rightPanel **=** **new** JPanel**();**

rightPanel**.**setLayout**(new** BoxLayout**(**rightPanel**,** BoxLayout**.**Y\_AXIS**));**

rightPanel**.**setBorder**(**BorderFactory**.**createTitledBorder**(**"Simulatoion Charts (The results of previous experiment are shown in blue)"**));**

myChartPanel **=** **new** JPanel**(new** GridLayout**(**0**,** 1**,** 0**,** 5**));**

mainChartPanel **=** **new** JScrollPane**(**myChartPanel**,** JScrollPane**.**VERTICAL\_SCROLLBAR\_ALWAYS**,**

JScrollPane**.**HORIZONTAL\_SCROLLBAR\_NEVER**);**

mainChartPanel**.**getVerticalScrollBar**().**setUnitIncrement**(**16**);**

rightPanel**.**add**(**mainChartPanel**);**

**return** rightPanel**;**

**}**

@Override

public JPanel createServiceDistributionPanel**()** **{**

JPanel serviceDistributionPanel **=** **new** JPanel**(new** GridLayout**(**0**,** 1**));**

serviceDistributionPanel**.**setBorder**(**BorderFactory**.**createTitledBorder**(**"Service Distribution (ST = Service Time)"**));**

serviceDistributionPanel**.**add**(**createDistributionControl**(**getSERVICEONE**(),** ""**,** 0**));**

**return** serviceDistributionPanel**;**

**}**

@Override

public boolean checkErrors**()** **{**

int arrival**,** service**;**

double arrivalMean **=** 0**,** serviceMean **=** 0**;**

double tmp1**,** tmp2**;**

arrival **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getARRIVAL**())).**getSelectedIndex**();**

service **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getSERVICEONE**())).**getSelectedIndex**();**

String errorName **=** ""**;**

**try** **{**

errorName **=** "the number of jobs "**;**

int jobs **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getSTEPS**())).**getText**());**

**if** **(**jobs **<** 0**)** **{**

errorName **+=** "must be a postive integer"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "the interval of histogram "**;**

double inerval **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getInterval**())).**getText**());**

**if** **(**inerval **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

**if** **(**arrival **==** 0**)** **{**

errorName **=** "Arrival Rate "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getLAMBDA**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

arrivalMean **=** tmp1**;**

**}**

**if** **(**arrival **==** 1**)** **{**

errorName **=** "Lower bound for arrival distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getUNIFORMLOWER**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "Upper bound for arrival distribution "**;**

tmp2 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getUNIFORMUPPER**())).**getText**());**

**if** **(**tmp2 **<** tmp1**)** **{**

errorName **+=** "must greater than lower bound"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

arrivalMean **=** **(**tmp1 **+** tmp2**)** **/** 2**;**

**}**

**if** **(**arrival **==** 2**)** **{**

errorName **=** "Constant of arrival distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getCONSTANT**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

arrivalMean **=** tmp1**;**

**}**

**if** **(**service **==** 0**)** **{**

errorName **=** "Service Rate "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getLAMBDA**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean **=** tmp1**;**

**}**

**if** **(**service **==** 1**)** **{**

errorName **=** "Lower bound for service distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getUNIFORMLOWER**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "Upper bound for service distribution "**;**

tmp2 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getUNIFORMUPPER**())).**getText**());**

**if** **(**tmp2 **<** tmp1**)** **{**

errorName **+=** "must greater than lower bound"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean **=** **(**tmp1 **+** tmp2**)** **/** 2**;**

**}**

**if** **(**service **==** 2**)** **{**

errorName **=** "Constant of service distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getCONSTANT**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean **=** tmp1**;**

**}**

**if** **(**arrivalMean **-** serviceMean **<** 0**)** **{**

JOptionPane**.**showMessageDialog**(null,** "Mean service time must less than mean arrival time"**);**

**return** **true;**

**}**

**}** **catch** **(**Exception ex**)** **{**

JOptionPane**.**showMessageDialog**(null,** errorName **+** "is not a invalid number"**);**

**return** **true;**

**}**

**return** **false;**

**}**

**}**

/\*

\* FeedbackModel.java

\*/

package gg1simulator**.**GUI**;**

**import** gg1simulator**.**Distribution**;**

**import** gg1simulator**.**FeedbackQueueCalculator**;**

**import** gg1simulator**.**FeedbackQueueSimulator**;**

**import** gg1simulator**.**Utility**;**

**import** java**.**awt**.**Dimension**;**

**import** java**.**awt**.**GridLayout**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** java**.**util**.**HashMap**;**

**import** javax**.**swing**.**BorderFactory**;**

**import** javax**.**swing**.**BoxLayout**;**

**import** javax**.**swing**.**JButton**;**

**import** javax**.**swing**.**JComboBox**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JScrollPane**;**

**import** javax**.**swing**.**JTextField**;**

public class FeedbackModel **extends** Simulator **{**

private JPanel myChartPanel**;**

private JScrollPane mainChartPanel**;**

private boolean flag **=** **false;**

private double preMaxInterArrivalTime**,** preMaxTotalInterArrivalTime**,** preMaxInterDepartureTime**,** preMaxInterMidArrivalTime**,**

preMaxInterMidDepartureTime**,** preMaxServiceTime1**,** preMaxServiceTime**,** preMaxResponseTime**;**

private int**[]** preInterArrivalTimeList**,** preTotalInterArrivalTimeList**,** preInterDepartureTimeList**,** preMidArrivalTimeList**,**

preInterMidDepartureTimeList**,** preServiceTimeList1**,** preServiceTimeList**,** preResponseTimeList**;**

private int preTotalArrivals**,** preArrivals1**,** preMidArrivals**,** preDepartures**;**

private double preInterval**;**

private HashMap**<**Integer**,** Double**>** preQueueMap **=** **new** HashMap**<>();**

@Override

public void init**()** **{**

**super.**setImageName**(**"FeedbackSingleServer.jpg"**);**

**super.**setModelType**(**1**);**

**super.**setNumOfServers**(**2**);**

**super.**init**();**

**((**JTextField**)** getColtrolsMap**().**get**(**getP**())).**setText**(**"50"**);**

**((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getLAMBDA**())).**setText**(**"4"**);**

**((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getLAMBDA**())).**setText**(**"1"**);**

**((**JTextField**)** getColtrolsMap**().**get**(**getSERVICETWO**()** **+** "-" **+** getLAMBDA**())).**setText**(**"1"**);**

**this.**setVisible**(true);**

**}**

@Override

public JPanel createSimButtons**()** **{**

JPanel btnPanel **=** **new** JPanel**();**

JButton btnSim **=** **new** JButton**(**"Run Simulator"**);**

btnSim**.**setPreferredSize**(new** Dimension**(**100**,** 24**));**

btnSim**.**addActionListener**(new** ActionListener**()** **{**

@Override

public void actionPerformed**(**ActionEvent e**)** **{**

**if** **(**checkErrors**())** **{**

**return;**

**}**

int steps**,** bars**;**

double p**,** interval**;**

String arrivalStr**,** serviceOneStr**,** serviceTwoStr**;**

Distribution arrivalDistribution**,** serviceOneDistribution**,** serviceTwoDistribution**;**

steps **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getSTEPS**())).**getText**());**

bars **=** Integer**.**parseInt**(((**JComboBox**)** getColtrolsMap**().**get**(**getNUMBEROFBARS**())).**getSelectedItem**().**toString**());**

p **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getP**())).**getText**())** **/** 100**;**

arrivalStr **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getARRIVAL**())).**getSelectedItem**().**toString**();**

serviceOneStr **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getSERVICEONE**())).**getSelectedItem**().**toString**();**

serviceTwoStr **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getSERVICETWO**())).**getSelectedItem**().**toString**();**

interval **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getInterval**())).**getText**());**

arrivalDistribution **=** getDistribution**(**arrivalStr**,** getARRIVAL**());**

serviceOneDistribution **=** getDistribution**(**serviceOneStr**,** getSERVICEONE**());**

serviceTwoDistribution **=** getDistribution**(**serviceTwoStr**,** getSERVICETWO**());**

FeedbackQueueSimulator simulator **=** **new** FeedbackQueueSimulator**(**arrivalDistribution**,** serviceOneDistribution**,** serviceTwoDistribution**,** bars**,** interval**,** p**);**

//simulator.getMaxValues();

simulator**.**run**(**steps**);**

simulator**.**outputValues**();**

FeedbackQueueCalculator c **=** **new** FeedbackQueueCalculator**(**arrivalDistribution**.**getMean**(),** serviceOneDistribution**.**getMean**(),** serviceTwoDistribution**.**getMean**(),** p**);**

mainResultPanel**.**removeAll**();**

mainResultPanel**.**add**(**createTitle**());**

int temp **=** 1**;**

**if** **(**arrivalDistribution**.**equals**(**"Exponential (M)"**)** **&&** serviceOneDistribution**.**equals**(**"Exponential (M)"**)** **&&** serviceTwoDistribution**.**equals**(**"Exponential (M)"**))** **{**

temp **=** 0**;**

**}**

double avg**,** sdv**,** realAvg**,** error**;**

String realValue**,** errorStr**;**

avg **=** simulator**.**getFirstUtilization**();**

realAvg **=** c**.**getFirstUtilization**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"#1 Utilization"**,** avg**,** realValue**,** errorStr**));**

avg **=** simulator**.**getSecondUtilization**();**

realAvg **=** c**.**getSecondUtilization**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"#2 Utilization"**,** avg**,** realValue**,** errorStr**));**

avg **=** simulator**.**getAvgInterArrivalTime**();**

sdv **=** simulator**.**getSDInterArrivalTime**();**

realAvg **=** c**.**getMeanArrivalTime**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Interarrival time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** simulator**.**getAvgTotalInterArrivalTime**();**

sdv **=** simulator**.**getSDTotalInterArrivalTime**();**

errorStr **=** ""**;**

mainResultPanel**.**add**(**createRow**(**"#1 Interarrival time"**,** avg**,** sdv**,** "" **,** errorStr**));**

avg **=** simulator**.**getAvgMidArrivalTime**();**

sdv **=** simulator**.**getSDMidArrivalTime**();**

errorStr **=** ""**;**

mainResultPanel**.**add**(**createRow**(**"#2 Interarrival time"**,** avg**,** sdv**,** ""**,** errorStr**));**

avg **=** simulator**.**getAvgServiceTime1**();**

sdv **=** simulator**.**getSDServiceTime1**();**

realAvg **=** c**.**getMeanFirstServiceTime**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"#1 Service time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** simulator**.**getAvgServiceTime**();**

sdv **=** simulator**.**getSDServiceTime**();**

realAvg **=** c**.**getMeanSecondServiceTime**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"#2 Service time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** simulator**.**getAvgInterDepartureTime**();**

sdv **=** simulator**.**getSDInterDepartureTime**();**

errorStr **=** ""**;**

mainResultPanel**.**add**(**createRow**(**"Interdeparture time"**,** avg**,** sdv**,** ""**,** errorStr**));**

avg **=** 1 **/**avg**;**

mainResultPanel**.**add**(**createRow**(**"Throughput"**,** avg**,** ""**,** errorStr**));**

avg **=** simulator**.**getAvgResponseTime**();**

sdv **=** simulator**.**getSDResponseTime**();**

realAvg **=** c**.**getResponseTime**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Response time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** simulator**.**getAvgQueueLength**();**

sdv **=** simulator**.**getSDQueueLength**();**

realAvg **=** c**.**getJobsInTheSystem**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Number of jobs"**,** avg**,** sdv**,** realValue**,** errorStr**));**

mainResultPanel**.**revalidate**();**

mainResultPanel**.**repaint**();**

myChartPanel**.**removeAll**();**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preInterArrivalTimeList**,** preArrivals1**,** preInterval**,** "System Interarrival Time (Previous Exp)"**,** 0**));**

**}**

myChartPanel**.**add**(**createChartPanel**(**simulator**.**getInterArrivalTimeList**(),** simulator**.**getArrivals1**(),** interval**,** "System Interarrival Time"**,** 1**));**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preTotalInterArrivalTimeList**,** preTotalArrivals**,** preInterval**,** "Interarrival Time-Queue #1 (Previous Exp)"**,** 0**));**

**}**

//myChartPanel.add(createChartPanel(simulator.getTotalInterArrivalTimeList(), simulator.getMaxTotalInterArrivalTime()/10, "Interarrival Time-Server #1", 1));

myChartPanel**.**add**(**createChartPanel**(**simulator**.**getTotalInterArrivalTimeList**(),** simulator**.**getTotalArrials**(),** interval**,** "Interarrival Time-Queue #1"**,** 1**));**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preMidArrivalTimeList**,** preMidArrivals**,** preInterval**,** "Interarrival Time-Queue #2 (Previous Exp)"**,** 0**));**

**}**

//myChartPanel.add(createChartPanel(simulator.getTotalInterArrivalTimeList(), simulator.getMaxTotalInterArrivalTime(), "Interarrival Time-Server #2", 1));

myChartPanel**.**add**(**createChartPanel**(**simulator**.**getMidArrivalTimeList**(),** simulator**.**getNumOfMidArrivalJobs**(),** interval**,** "Interarrival Time-Queue #2"**,** 1**));**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preInterDepartureTimeList**,** preDepartures**,** preInterval**,** "Interdeparture time (Previous Exp)"**,** 0**));**

**}**

//myChartPanel.add(createChartPanel(simulator.getInterDepartureTimeList(), simulator.getMaxInterDepartureTime(), "Interdeparture time", 1));

myChartPanel**.**add**(**createChartPanel**(**simulator**.**getInterDepartureTimeList**(),** simulator**.**getTotalDepartures**(),** interval**,** "Interdeparture time"**,** 1**));**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preResponseTimeList**,** preDepartures**,** preInterval**,** "Response time (Previous Exp)"**,** 0**));**

**}**

//myChartPanel.add(createChartPanel(simulator.getResponseTimeList(), simulator.getMaxResponseTime(), "Response time", 1));

myChartPanel**.**add**(**createChartPanel**(**simulator**.**getResponseTimeList**(),** simulator**.**getTotalDepartures**(),** interval**,** "Response time"**,** 1**));**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preQueueMap**,** "Number of Jobs (Previous Exp)"**,** bars**,** 0**));**

**}**

preQueueMap **=** **(**HashMap**)** simulator**.**getQueueLength**().**clone**();**

myChartPanel**.**add**(**createChartPanel**(**simulator**.**getQueueLength**(),** "Number of Jobs"**,** bars**,** 1**));**

myChartPanel**.**revalidate**();**

myChartPanel**.**repaint**();**

flag **=** **true;**

preInterArrivalTimeList **=** **new** int**[**bars**];**

preTotalInterArrivalTimeList **=** **new** int**[**bars**];**

preInterDepartureTimeList **=** **new** int**[**bars**];**

preMidArrivalTimeList **=** **new** int**[**bars**];**

preInterMidDepartureTimeList **=** **new** int**[**bars**];**

preServiceTimeList1 **=** **new** int**[**bars**];**

preServiceTimeList **=** **new** int**[**bars**];**

preResponseTimeList **=** **new** int**[**bars**];**

System**.**arraycopy**(**simulator**.**getInterArrivalTimeList**(),** 0**,** preInterArrivalTimeList**,** 0**,** bars**);**

System**.**arraycopy**(**simulator**.**getTotalInterArrivalTimeList**(),** 0**,** preTotalInterArrivalTimeList**,** 0**,** bars**);**

System**.**arraycopy**(**simulator**.**getInterDepartureTimeList**(),** 0**,** preInterDepartureTimeList**,** 0**,** bars**);**

System**.**arraycopy**(**simulator**.**getMidArrivalTimeList**(),** 0**,** preMidArrivalTimeList**,** 0**,** bars**);**

System**.**arraycopy**(**simulator**.**getInterMidDepartureTimeList**(),** 0**,** preInterMidDepartureTimeList**,** 0**,** bars**);**

System**.**arraycopy**(**simulator**.**getServiceTimeList1**(),** 0**,** preServiceTimeList1**,** 0**,** bars**);**

System**.**arraycopy**(**simulator**.**getServiceTimeList**(),** 0**,** preServiceTimeList**,** 0**,** bars**);**

System**.**arraycopy**(**simulator**.**getResponseTimeList**(),** 0**,** preResponseTimeList**,** 0**,** bars**);**

preArrivals1 **=** simulator**.**getArrivals1**();**

preTotalArrivals **=** simulator**.**getTotalArrials**();**

preMidArrivals **=** simulator**.**getNumOfMidArrivalJobs**();**

preDepartures **=** simulator**.**getTotalDepartures**();**

preInterval **=** interval**;**

**}**

**});**

btnPanel**.**add**(**btnSim**);**

**return** btnPanel**;**

**}**

@Override

public JPanel createChartPanel**()** **{**

JPanel rightPanel **=** **new** JPanel**();**

rightPanel**.**setLayout**(new** BoxLayout**(**rightPanel**,** BoxLayout**.**Y\_AXIS**));**

rightPanel**.**setBorder**(**BorderFactory**.**createTitledBorder**(**"Simulatoion Charts (The results of previous experiment are shown in blue)"**));**

myChartPanel **=** **new** JPanel**(new** GridLayout**(**0**,** 1**,** 0**,** 5**));**

mainChartPanel **=** **new** JScrollPane**(**myChartPanel**,** JScrollPane**.**VERTICAL\_SCROLLBAR\_ALWAYS**,**

JScrollPane**.**HORIZONTAL\_SCROLLBAR\_NEVER**);**

rightPanel**.**add**(**mainChartPanel**);**

**return** rightPanel**;**

**}**

@Override

public JPanel createServiceDistributionPanel**()** **{**

JPanel serviceDistributionPanel **=** **new** JPanel**(new** GridLayout**(**0**,** 1**));**

serviceDistributionPanel**.**setBorder**(**BorderFactory**.**createTitledBorder**(**"Service Distribution (ST = Service Time)"**));**

serviceDistributionPanel**.**add**(**createDistributionControl**(**getSERVICEONE**(),** "#1 "**,** 1**));**

serviceDistributionPanel**.**add**(**createDistributionControl**(**getSERVICETWO**(),** "#2 "**,** 0**));**

**return** serviceDistributionPanel**;**

**}**

@Override

public boolean checkErrors**()** **{**

int arrival**,** service1**,** service2**;**

double arrivalMean **=** 0**,** serviceMean1 **=** 0**,** serviceMean2 **=** 0**;**

double tmp1**,** tmp2**;**

arrival **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getARRIVAL**())).**getSelectedIndex**();**

service1 **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getSERVICEONE**())).**getSelectedIndex**();**

service2 **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getSERVICETWO**())).**getSelectedIndex**();**

String errorName **=** ""**;**

**try** **{**

errorName **=** "the number of jobs "**;**

int jobs **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getSTEPS**())).**getText**());**

**if** **(**jobs **<** 0**)** **{**

errorName **+=** "must be a postive integer"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "the interval of histogram "**;**

double inerval **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getInterval**())).**getText**());**

**if** **(**inerval **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "p "**;**

double p **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getP**())).**getText**());**

**if** **(**p **<** 0 **||** p **>=** 100**)** **{**

errorName **+=** "must between 0 and 100"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

p **=** p **/** 100**;**

**if** **(**arrival **==** 0**)** **{**

errorName **=** "Mean arrival time "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getLAMBDA**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

arrivalMean **=** tmp1**;**

**}**

**if** **(**arrival **==** 1**)** **{**

errorName **=** "Lower bound for arrival distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getUNIFORMLOWER**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "Upper bound for arrival distribution "**;**

tmp2 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getUNIFORMUPPER**())).**getText**());**

**if** **(**tmp2 **<** tmp1**)** **{**

errorName **+=** "must greater than lower bound"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

arrivalMean **=** **(**tmp1 **+** tmp2**)** **/** 2**;**

**}**

**if** **(**arrival **==** 2**)** **{**

errorName **=** "Constant of arrival distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getCONSTANT**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

arrivalMean **=** tmp1**;**

**}**

**if** **(**service1 **==** 0**)** **{**

errorName **=** "Mean service time "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getLAMBDA**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean1 **=** tmp1**;**

**}**

**if** **(**service1 **==** 1**)** **{**

errorName **=** "Lower bound for service distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getUNIFORMLOWER**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "Upper bound for service distribution "**;**

tmp2 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getUNIFORMUPPER**())).**getText**());**

**if** **(**tmp2 **<** tmp1**)** **{**

errorName **+=** "must greater than lower bound"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean1 **=** **(**tmp1 **+** tmp2**)** **/** 2**;**

**}**

**if** **(**service1 **==** 2**)** **{**

errorName **=** "Constant of service distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getCONSTANT**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean1 **=** tmp1**;**

**}**

**if** **(**service2 **==** 0**)** **{**

errorName **=** "Service Rate "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICETWO**()** **+** "-" **+** getLAMBDA**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean2 **=** tmp1**;**

**}**

**if** **(**service2 **==** 1**)** **{**

errorName **=** "Lower bound for service distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICETWO**()** **+** "-" **+** getUNIFORMLOWER**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "Upper bound for service distribution "**;**

tmp2 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICETWO**()** **+** "-" **+** getUNIFORMUPPER**())).**getText**());**

**if** **(**tmp2 **<** tmp1**)** **{**

errorName **+=** "must greater than lower bound"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean2 **=** **(**tmp1 **+** tmp2**)** **/** 2**;**

**}**

**if** **(**service2 **==** 2**)** **{**

errorName **=** "Constant of service distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICETWO**()** **+** "-" **+** getCONSTANT**())).**getText**());**

**if** **(**tmp1 **<** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean2 **=** tmp1**;**

**}**

**if** **((**serviceMean1 **>** **(**1 **-** p**)** **\*** arrivalMean**)** **||** **(**p **\*** serviceMean2 **>** **(**1 **-** p**)** **\*** arrivalMean**))** **{**

JOptionPane**.**showMessageDialog**(null,** "Please increase the mean interarrival time or decrease the mean service time"**);**

**return** **true;**

**}**

**}** **catch** **(**Exception ex**)** **{**

JOptionPane**.**showMessageDialog**(null,** errorName **+** "is not a invalid number"**);**

**return** **true;**

**}**

**return** **false;**

**}**

**}**

/\*

\* MultipleServersModel.java

\*/

package gg1simulator**.**GUI**;**

**import** gg1simulator**.**Distribution**;**

**import** gg1simulator**.**InteractiveSystemCalculator**;**

**import** gg1simulator**.**MultipleServersCalculator**;**

**import** gg1simulator**.**MultipleServersSimulator**;**

**import** gg1simulator**.**Utility**;**

**import** java**.**awt**.**BorderLayout**;**

**import** java**.**awt**.**Dimension**;**

**import** java**.**awt**.**GridLayout**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** java**.**util**.**HashMap**;**

**import** javax**.**swing**.**BorderFactory**;**

**import** javax**.**swing**.**BoxLayout**;**

**import** javax**.**swing**.**JButton**;**

**import** javax**.**swing**.**JComboBox**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JScrollPane**;**

**import** javax**.**swing**.**JTextField**;**

public class MultipleServersModel **extends** Simulator **{**

private JPanel myChartPanel**;**

private JScrollPane mainChartPanel**;**

private boolean flag **=** **false;**

private int arrivals**,** departures**;**

private int**[]** preInterArrivalTime**,** preCycleTime**,** preInterDepartureTime**,** preResponseTime**;**

private double preInterval**;**

private HashMap**<**Integer**,** Double**>** preQueueMap **=** **new** HashMap**<>();**

@Override

public void init**()** **{**

**super.**setImageName**(**"MultipleServersModel.jpg"**);**

**super.**setNumOfServers**(**1**);**

**super.**setModelType**(**4**);**

**super.**init**();**

**((**JTextField**)** getColtrolsMap**().**get**(**getNumOfServers**())).**setText**(**"2"**);**

**}**

@Override

public JPanel createSimButtons**()** **{**

JPanel btnPanel **=** **new** JPanel**(new** BorderLayout**());**

JLabel msgLabel **=** **new** JLabel**();**

msgLabel**.**setPreferredSize**(new** Dimension**(**580**,** 18**));**

JPanel buttonPanel **=** **new** JPanel**();**

JButton btnSim **=** **new** JButton**(**"Run Simulator"**);**

btnSim**.**setPreferredSize**(new** Dimension**(**100**,** 24**));**

btnSim**.**addActionListener**(new** ActionListener**()** **{**

@Override

public void actionPerformed**(**ActionEvent e**)** **{**

**if** **(**checkErrors**())** **{**

**return;**

**}**

int steps**,** bars**,** m**;**

double interval**;**

String thinkTimeStr**,** serviceStr**;**

Distribution thinkTimeDistribution**,** serviceDistribution**;**

steps **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getSTEPS**())).**getText**());**

bars **=** Integer**.**parseInt**(((**JComboBox**)** getColtrolsMap**().**get**(**getNUMBEROFBARS**())).**getSelectedItem**().**toString**());**

m **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getNumOfServers**())).**getText**());**

thinkTimeStr **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getARRIVAL**())).**getSelectedItem**().**toString**();**

serviceStr **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getSERVICEONE**())).**getSelectedItem**().**toString**();**

interval **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getInterval**())).**getText**());**

thinkTimeDistribution **=** getDistribution**(**thinkTimeStr**,** getARRIVAL**());**

serviceDistribution **=** getDistribution**(**serviceStr**,** getSERVICEONE**());**

MultipleServersSimulator sim **=** **new** MultipleServersSimulator**(**thinkTimeDistribution**,** serviceDistribution**,** m**,** bars**,** interval**);**

sim**.**run**(**steps**);**

mainResultPanel**.**removeAll**();**

mainResultPanel**.**add**(**createTitle**());**

MultipleServersCalculator c **=** **new** MultipleServersCalculator**(**m**,** thinkTimeDistribution**.**getMean**(),** serviceDistribution**.**getMean**());**

c**.**Calculate**();**

int temp **=** 1**;**

**if** **(**thinkTimeStr**.**equals**(**"Exponential (M)"**)** **&&** serviceStr**.**equals**(**"Exponential (M)"**))** **{**

temp **=** 0**;**

**}**

double avg**,** sdv**,** realAvg**,** error**;**

String realValue**,** errorStr**;**

avg **=** sim**.**getUtilization**();**

realAvg **=** c**.**getUtilization**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Server Utilization"**,** avg**,** realValue**,** errorStr**));**

avg **=** sim**.**getAvgInterArrivalTime**();**

sdv **=** sim**.**getSDInterArrivalTime**();**

realAvg **=** c**.**getArrivalMean**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Interarrival time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** sim**.**getAvgResponseTime**();**

sdv **=** sim**.**getSDResponseTime**();**

realAvg **=** c**.**getResponseTime**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Response time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** sim**.**getAvgInterDeparutreTime**();**

sdv **=** sim**.**getSDInterDeparutreTime**();**

realValue **=** ""**;**

mainResultPanel**.**add**(**createRow**(**"Interdeparture time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** 1 **/** avg**;**

realValue **=** ""**;**

mainResultPanel**.**add**(**createRow**(**"Throughput"**,** avg**,** realValue**,** errorStr**));**

avg **=** sim**.**getAvgWaitTime**();**

sdv **=** sim**.**getSDWaitTime**();**

realAvg **=** c**.**getWaitTime**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Queue wait time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** sim**.**getAvgJobsIntheSystem**();**

sdv **=** sim**.**getSDJobsIntheSystem**();**

realAvg **=** c**.**getQueueLength**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Number of jobs:"**,** avg**,** sdv**,** realValue**,** errorStr**));**

mainResultPanel**.**revalidate**();**

mainResultPanel**.**repaint**();**

myChartPanel**.**removeAll**();**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preInterArrivalTime**,** arrivals**,** preInterval**,** "Interarrival time (Previous Exp)"**,** 0**));**

**}**

myChartPanel**.**add**(**createChartPanel**(**sim**.**getInterArrivalList**(),** sim**.**getArrivals**(),** interval**,** "Interarrival time"**,** 1**));**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preResponseTime**,** departures**,** preInterval**,** "Response time (Previous Exp)"**,** 0**));**

**}**

myChartPanel**.**add**(**createChartPanel**(**sim**.**getResponseTimeList**(),** sim**.**getDepartures**(),** interval**,** "Response time"**,** 1**));**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preCycleTime**,** departures**,** preInterval**,** "Interdeparture time (Previous Exp)"**,** 0**));**

**}**

myChartPanel**.**add**(**createChartPanel**(**sim**.**getInterDepartureList**(),** sim**.**getDepartures**(),** interval**,** "Interdeparture time"**,** 1**));**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preQueueMap**,** "Number of jobs (Previous Exp)"**,** bars**,** 0**));**

**}**

preQueueMap **=** **(**HashMap**)** sim**.**getQueueMap**().**clone**();**

myChartPanel**.**add**(**createChartPanel**(**sim**.**getQueueMap**(),** "Number of jobs"**,** bars**,** 1**));**

myChartPanel**.**revalidate**();**

myChartPanel**.**repaint**();**

flag **=** **true;**

preInterArrivalTime **=** **new** int**[**bars**];**

preResponseTime **=** **new** int**[**bars**];**

preCycleTime **=** **new** int**[**bars**];**

System**.**arraycopy**(**sim**.**getInterArrivalList**(),** 0**,** preInterArrivalTime**,** 0**,** bars**);**

System**.**arraycopy**(**sim**.**getResponseTimeList**(),** 0**,** preResponseTime**,** 0**,** bars**);**

System**.**arraycopy**(**sim**.**getInterDepartureList**(),** 0**,** preCycleTime**,** 0**,** bars**);**

arrivals **=** sim**.**getArrivals**();**

preInterval **=** interval**;**

departures **=** sim**.**getDepartures**();**

**}**

**});**

buttonPanel**.**add**(**btnSim**);**

btnPanel**.**add**(**msgLabel**,** BorderLayout**.**NORTH**);**

btnPanel**.**add**(**buttonPanel**,** BorderLayout**.**CENTER**);**

**return** btnPanel**;**

**}**

@Override

public JPanel createChartPanel**()** **{**

JPanel rightPanel **=** **new** JPanel**();**

rightPanel**.**setLayout**(new** BoxLayout**(**rightPanel**,** BoxLayout**.**Y\_AXIS**));**

rightPanel**.**setBorder**(**BorderFactory**.**createTitledBorder**(**"Simulatoion Charts (The results of previous experiment are shown in blue)"**));**

myChartPanel **=** **new** JPanel**(new** GridLayout**(**0**,** 1**,** 0**,** 5**));**

mainChartPanel **=** **new** JScrollPane**(**myChartPanel**,** JScrollPane**.**VERTICAL\_SCROLLBAR\_ALWAYS**,**

JScrollPane**.**HORIZONTAL\_SCROLLBAR\_NEVER**);**

mainChartPanel**.**getVerticalScrollBar**().**setUnitIncrement**(**16**);**

rightPanel**.**add**(**mainChartPanel**);**

**return** rightPanel**;**

**}**

@Override

public JPanel createServiceDistributionPanel**()** **{**

JPanel serviceDistributionPanel **=** **new** JPanel**(new** GridLayout**(**0**,** 1**));**

serviceDistributionPanel**.**setBorder**(**BorderFactory**.**createTitledBorder**(**"Service Distribution (ST = Service Time)"**));**

serviceDistributionPanel**.**add**(**createDistributionControl**(**getSERVICEONE**(),** ""**,** 0**));**

**return** serviceDistributionPanel**;**

**}**

@Override

public boolean checkErrors**()** **{**

int arrival**,** service**,** n **=** 1**;**

double arrivalMean **=** 0**,** serviceMean **=** 0**;**

double tmp1**,** tmp2**;**

arrival **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getARRIVAL**())).**getSelectedIndex**();**

service **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getSERVICEONE**())).**getSelectedIndex**();**

String errorName **=** ""**;**

**try** **{**

errorName **=** "the number of jobs "**;**

int jobs **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getSTEPS**())).**getText**());**

**if** **(**jobs **<** 0**)** **{**

errorName **+=** "must be a postive integer"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "the number of servers "**;**

n **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getNumOfServers**())).**getText**());**

**if** **(**n **<=** 0**)** **{**

errorName **+=** "must be a postive integer"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

**if** **(**n **>** 16**)** **{**

errorName **+=** "must be lest than 16"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "the interval of histogram "**;**

double inerval **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getInterval**())).**getText**());**

**if** **(**inerval **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

**if** **(**arrival **==** 0**)** **{**

errorName **=** "Mean think time "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getLAMBDA**())).**getText**());**

**if** **(**tmp1 **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

arrivalMean **=** tmp1**;**

**}**

**if** **(**arrival **==** 1**)** **{**

errorName **=** "Lower bound for think time distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getUNIFORMLOWER**())).**getText**());**

**if** **(**tmp1 **<=** 0**)** **{**

errorName **+=** "must be a postive number "**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "Upper bound for think time distribution "**;**

tmp2 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getUNIFORMUPPER**())).**getText**());**

**if** **(**tmp2 **<=** tmp1**)** **{**

errorName **+=** "must greater than lower bound"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

arrivalMean **=** **(**tmp1 **+** tmp2**)** **/** 2**;**

**}**

**if** **(**arrival **==** 2**)** **{**

errorName **=** "Constant of think time distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getCONSTANT**())).**getText**());**

**if** **(**tmp1 **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

arrivalMean **=** tmp1**;**

**}**

**if** **(**service **==** 0**)** **{**

errorName **=** "Mean service time"**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getLAMBDA**())).**getText**());**

**if** **(**tmp1 **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean **=** tmp1**;**

**}**

**if** **(**service **==** 1**)** **{**

errorName **=** "Lower bound for service distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getUNIFORMLOWER**())).**getText**());**

**if** **(**tmp1 **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "Upper bound for service distribution "**;**

tmp2 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getUNIFORMUPPER**())).**getText**());**

**if** **(**tmp2 **<=** tmp1**)** **{**

errorName **+=** "must greater than lower bound"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean **=** **(**tmp1 **+** tmp2**)** **/** 2**;**

**}**

**if** **(**service **==** 2**)** **{**

errorName **=** "Constant of service distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getCONSTANT**())).**getText**());**

**if** **(**tmp1 **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean **=** tmp1**;**

**}**

**if(**arrivalMean **<** serviceMean **/** n**){**

JOptionPane**.**showMessageDialog**(null,** "Mean service time must less than mean arrival time"**);**

**return** **true;**

**}**

**}** **catch** **(**Exception ex**)** **{**

JOptionPane**.**showMessageDialog**(null,** errorName **+** "is not a invalid number"**);**

**return** **true;**

**}**

**return** **false;**

**}**

**}**

/\*

\* InteractiveSystem.java

\*/

package gg1simulator**.**GUI**;**

**import** gg1simulator**.**Distribution**;**

**import** gg1simulator**.**InteractiveSystemCalculator**;**

**import** gg1simulator**.**InteractiveSystemSimuator**;**

**import** gg1simulator**.**Utility**;**

**import** java**.**awt**.**BorderLayout**;**

**import** java**.**awt**.**Dimension**;**

**import** java**.**awt**.**GridLayout**;**

**import** java**.**awt**.**event**.**ActionEvent**;**

**import** java**.**awt**.**event**.**ActionListener**;**

**import** java**.**util**.**HashMap**;**

**import** javax**.**swing**.**BorderFactory**;**

**import** javax**.**swing**.**BoxLayout**;**

**import** javax**.**swing**.**JButton**;**

**import** javax**.**swing**.**JComboBox**;**

**import** javax**.**swing**.**JLabel**;**

**import** javax**.**swing**.**JOptionPane**;**

**import** javax**.**swing**.**JPanel**;**

**import** javax**.**swing**.**JScrollPane**;**

**import** javax**.**swing**.**JTextField**;**

public class InteractiveSystem **extends** Simulator **{**

private JPanel myChartPanel**;**

private JScrollPane mainChartPanel**;**

private boolean flag **=** **false;**

private int arrivals**,** departures**;**

private int**[]** preInterArrivalTime**,** preCycleTime**,** preInterDepartureTime**,** preResponseTime**;**

private double preInterval**;**

private HashMap**<**Integer**,** Double**>** preQueueMap **=** **new** HashMap**<>();**

@Override

public void init**()** **{**

**super.**setImageName**(**"InteractiveSystem.jpg"**);**

**super.**setNumOfServers**(**1**);**

**super.**setModelType**(**3**);**

**super.**init**();**

**((**JTextField**)** getColtrolsMap**().**get**(**getNumOfTerminals**())).**setText**(**"3"**);**

**((**JTextField**)** getColtrolsMap**().**get**(**getNumOfServers**())).**setText**(**"1"**);**

**}**

@Override

public JPanel createSimButtons**()** **{**

JPanel btnPanel **=** **new** JPanel**(new** BorderLayout**());**

JLabel msgLabel **=** **new** JLabel**();**

msgLabel**.**setPreferredSize**(new** Dimension**(**580**,** 18**));**

JPanel buttonPanel **=** **new** JPanel**();**

JButton btnSim **=** **new** JButton**(**"Run Simulator"**);**

btnSim**.**setPreferredSize**(new** Dimension**(**100**,** 24**));**

btnSim**.**addActionListener**(new** ActionListener**()** **{**

@Override

public void actionPerformed**(**ActionEvent e**)** **{**

**if** **(**checkErrors**())** **{**

**return;**

**}**

int steps**,** bars**,** n**,** m**;**

double interval**;**

String thinkTimeStr**,** serviceStr**;**

Distribution thinkTimeDistribution**,** serviceDistribution**;**

steps **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getSTEPS**())).**getText**());**

bars **=** Integer**.**parseInt**(((**JComboBox**)** getColtrolsMap**().**get**(**getNUMBEROFBARS**())).**getSelectedItem**().**toString**());**

n **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getNumOfTerminals**())).**getText**());**

m **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getNumOfServers**())).**getText**());**

thinkTimeStr **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getARRIVAL**())).**getSelectedItem**().**toString**();**

serviceStr **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getSERVICEONE**())).**getSelectedItem**().**toString**();**

interval **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getInterval**())).**getText**());**

thinkTimeDistribution **=** getDistribution**(**thinkTimeStr**,** getARRIVAL**());**

serviceDistribution **=** getDistribution**(**serviceStr**,** getSERVICEONE**());**

InteractiveSystemSimuator sim **=** **new** InteractiveSystemSimuator**(**thinkTimeDistribution**,** serviceDistribution**,** n**,** m**,** bars**,** interval**);**

sim**.**run**(**steps**);**

mainResultPanel**.**removeAll**();**

mainResultPanel**.**add**(**createTitle**());**

InteractiveSystemCalculator c **=** **new** InteractiveSystemCalculator**(**thinkTimeDistribution**.**getMean**(),** serviceDistribution**.**getMean**(),** n**,** m**);**

c**.**Calculate**();**

int temp **=** 1**;**

**if** **(**thinkTimeStr**.**equals**(**"Exponential (M)"**)** **&&** serviceStr**.**equals**(**"Exponential (M)"**))** **{**

temp **=** 0**;**

**}**

double avg**,** sdv**,** realAvg**,** error**;**

String realValue**,** errorStr**;**

avg **=** sim**.**getUtilization**();**

realAvg **=** c**.**getUtilization**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Server Utilization"**,** avg**,** realValue**,** errorStr**));**

avg **=** sim**.**getAvgInterArrivalTime**();**

sdv **=** sim**.**getSDInterArrivalTime**();**

errorStr **=** ""**;**

mainResultPanel**.**add**(**createRow**(**"Interarrival time"**,** avg**,** sdv**,** ""**,** errorStr**));**

avg **=** sim**.**getAvgResponseTime**();**

sdv **=** sim**.**getSDResponseTime**();**

realAvg **=** c**.**getResponseTime**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Response time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** sim**.**getAvgCycleTime**();**

sdv **=** sim**.**getSDCycleTime**();**

realAvg **=** c**.**getCycleTime**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Cycle time"**,** avg**,** sdv**,** realValue**,** errorStr**));**

avg **=** n **/** avg**;**

mainResultPanel**.**add**(**createRow**(**"Thoughput"**,** avg**,** ""**,** ""**));**

avg **=** sim**.**getAvgJobsIntheSystem**();**

sdv **=** sim**.**getSDJobsIntheSystem**();**

realAvg **=** c**.**getQueueLength**();**

realValue **=** Utility**.**formatOutput**(**realAvg**);**

error **=** 100 **\*** **(**avg **-** realAvg**)** **/** realAvg**;**

**if** **(**temp **==** 1**)** **{**

error **\*=** **-**1**;**

**}**

errorStr **=** Utility**.**formatOutput**(**error**);**

mainResultPanel**.**add**(**createRow**(**"Number of jobs:"**,** avg**,** sdv**,** realValue**,** errorStr**));**

mainResultPanel**.**revalidate**();**

mainResultPanel**.**repaint**();**

myChartPanel**.**removeAll**();**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preInterArrivalTime**,** arrivals**,** preInterval**,** "Interarrival time (Previous Exp)"**,** 0**));**

**}**

myChartPanel**.**add**(**createChartPanel**(**sim**.**getInterArrivalList**(),** sim**.**getArrivals**(),** interval**,** "Interarrival time"**,** 1**));**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preResponseTime**,** departures**,** preInterval**,** "Response time (Previous Exp)"**,** 0**));**

**}**

myChartPanel**.**add**(**createChartPanel**(**sim**.**getResponseTimeList**(),** sim**.**getDepartures**(),** interval**,** "Response time"**,** 1**));**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preCycleTime**,** departures**,** preInterval**,** "Cycle time (Previous Exp)"**,** 0**));**

**}**

myChartPanel**.**add**(**createChartPanel**(**sim**.**getCycleTimeList**(),** sim**.**getDepartures**(),** interval**,** "Cycle time"**,** 1**));**

**if** **(**flag**)** **{**

myChartPanel**.**add**(**createChartPanel**(**preQueueMap**,** "Number of jobs (Previous Exp)"**,** bars**,** 0**));**

**}**

preQueueMap **=** **(**HashMap**)**sim**.**getQueueMap**().**clone**();**

myChartPanel**.**add**(**createChartPanel**(**sim**.**getQueueMap**(),** "Number of jobs"**,** bars**,** 1**));**

myChartPanel**.**revalidate**();**

myChartPanel**.**repaint**();**

flag **=** **true;**

preInterArrivalTime **=** **new** int**[**bars**];**

preResponseTime **=** **new** int**[**bars**];**

preCycleTime **=** **new** int**[**bars**];**

System**.**arraycopy**(**sim**.**getInterArrivalList**(),** 0**,** preInterArrivalTime**,** 0**,** bars**);**

System**.**arraycopy**(**sim**.**getResponseTimeList**(),** 0**,** preResponseTime**,** 0**,** bars**);**

System**.**arraycopy**(**sim**.**getCycleTimeList**(),** 0**,** preCycleTime**,** 0**,** bars**);**

arrivals **=** sim**.**getArrivals**();**

preInterval **=** interval**;**

departures **=** sim**.**getDepartures**();**

**}**

**});**

buttonPanel**.**add**(**btnSim**);**

btnPanel**.**add**(**msgLabel**,** BorderLayout**.**NORTH**);**

btnPanel**.**add**(**buttonPanel**,** BorderLayout**.**CENTER**);**

**return** btnPanel**;**

**}**

@Override

public JPanel createChartPanel**()** **{**

JPanel rightPanel **=** **new** JPanel**();**

rightPanel**.**setLayout**(new** BoxLayout**(**rightPanel**,** BoxLayout**.**Y\_AXIS**));**

rightPanel**.**setBorder**(**BorderFactory**.**createTitledBorder**(**"Simulatoion Charts (The results of previous experiment are shown in blue)"**));**

myChartPanel **=** **new** JPanel**(new** GridLayout**(**0**,** 1**,** 0**,** 5**));**

mainChartPanel **=** **new** JScrollPane**(**myChartPanel**,** JScrollPane**.**VERTICAL\_SCROLLBAR\_ALWAYS**,**

JScrollPane**.**HORIZONTAL\_SCROLLBAR\_NEVER**);**

mainChartPanel**.**getVerticalScrollBar**().**setUnitIncrement**(**16**);**

rightPanel**.**add**(**mainChartPanel**);**

**return** rightPanel**;**

**}**

@Override

public JPanel createServiceDistributionPanel**()** **{**

JPanel serviceDistributionPanel **=** **new** JPanel**(new** GridLayout**(**0**,** 1**));**

serviceDistributionPanel**.**setBorder**(**BorderFactory**.**createTitledBorder**(**"Service Distribution (ST = Service Time)"**));**

serviceDistributionPanel**.**add**(**createDistributionControl**(**getSERVICEONE**(),** ""**,** 0**));**

**return** serviceDistributionPanel**;**

**}**

@Override

public boolean checkErrors**()** **{**

int arrival**,** service**;**

double arrivalMean **=** 0**,** serviceMean **=** 0**;**

double tmp1**,** tmp2**;**

arrival **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getARRIVAL**())).**getSelectedIndex**();**

service **=** **((**JComboBox**)** getColtrolsMap**().**get**(**getSERVICEONE**())).**getSelectedIndex**();**

String errorName **=** ""**;**

**try** **{**

errorName **=** "the number of jobs "**;**

int jobs **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getSTEPS**())).**getText**());**

**if** **(**jobs **<** 0**)** **{**

errorName **+=** "must be a postive integer"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "the number of terminals "**;**

int m **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getNumOfTerminals**())).**getText**());**

**if** **(**m **<=** 0**)** **{**

errorName **+=** "must be a postive integer"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "the number of servers "**;**

int n **=** Integer**.**parseInt**(((**JTextField**)** getColtrolsMap**().**get**(**getNumOfServers**())).**getText**());**

**if** **(**n **<=** 0**)** **{**

errorName **+=** "must be a postive integer"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "the interval of histogram "**;**

double inerval **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getInterval**())).**getText**());**

**if** **(**inerval **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

**if** **(**arrival **==** 0**)** **{**

errorName **=** "Mean think time "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getLAMBDA**())).**getText**());**

**if** **(**tmp1 **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

arrivalMean **=** tmp1**;**

**}**

**if** **(**arrival **==** 1**)** **{**

errorName **=** "Lower bound for think time distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getUNIFORMLOWER**())).**getText**());**

**if** **(**tmp1 **<=** 0**)** **{**

errorName **+=** "must be a postive number "**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "Upper bound for think time distribution "**;**

tmp2 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getUNIFORMUPPER**())).**getText**());**

**if** **(**tmp2 **<=** tmp1**)** **{**

errorName **+=** "must greater than lower bound"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

arrivalMean **=** **(**tmp1 **+** tmp2**)** **/** 2**;**

**}**

**if** **(**arrival **==** 2**)** **{**

errorName **=** "Constant of think time distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getARRIVAL**()** **+** "-" **+** getCONSTANT**())).**getText**());**

**if** **(**tmp1 **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

arrivalMean **=** tmp1**;**

**}**

**if** **(**service **==** 0**)** **{**

errorName **=** "Mean service time"**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getLAMBDA**())).**getText**());**

**if** **(**tmp1 **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean **=** tmp1**;**

**}**

**if** **(**service **==** 1**)** **{**

errorName **=** "Lower bound for service distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getUNIFORMLOWER**())).**getText**());**

**if** **(**tmp1 **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

errorName **=** "Upper bound for service distribution "**;**

tmp2 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getUNIFORMUPPER**())).**getText**());**

**if** **(**tmp2 **<=** tmp1**)** **{**

errorName **+=** "must greater than lower bound"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean **=** **(**tmp1 **+** tmp2**)** **/** 2**;**

**}**

**if** **(**service **==** 2**)** **{**

errorName **=** "Constant of service distribution "**;**

tmp1 **=** Double**.**parseDouble**(((**JTextField**)** getColtrolsMap**().**get**(**getSERVICEONE**()** **+** "-" **+** getCONSTANT**())).**getText**());**

**if** **(**tmp1 **<=** 0**)** **{**

errorName **+=** "must be a postive number"**;**

JOptionPane**.**showMessageDialog**(null,** errorName**);**

**return** **true;**

**}**

serviceMean **=** tmp1**;**

**}**

**if** **(**m **-** n **<** 0**)** **{**

JOptionPane**.**showMessageDialog**(null,** "the number of terminals must greater than the number of servers"**);**

**return** **true;**

**}**

**}** **catch** **(**Exception ex**)** **{**

JOptionPane**.**showMessageDialog**(null,** errorName **+** "is not a invalid number"**);**

**return** **true;**

**}**

**return** **false;**

**}**

**}**