Objectives: Study of the behavior of an M/M/1 queue by discrete simulation. Programming in Java for the management of the event list. Comparison by simulation with the theoretical quantities studied in class for this type of queue.

Generalities

The Markov chain of an M/M/1 queue is a birth-and-death process. Arrivals are determined by a Poisson process (inter-arrival times following an exponential distribution with parameter λ), and the service times follow an exponential distribution with parameter μ.
Simulations will be carried out with $\lambda < \mu$, in order to obtain stability in steady state.

Durations following an exponential distribution

Recall: the exponential law with parameter λ has the probability density function

$$p(X = x) = \lambda e^{-\lambda x} \quad (x > 0)$$

An interesting property is that a random variable X following an exponential law with parameter λ has an expected value

$$E(X) = \frac{1}{\lambda}.$$

To generate durations following an exponential distribution with parameter λ, we use the cumulative distribution function $F(X) = 1 - e^{-\lambda X}$. We know that $F(X) \in]0, 1[$. Thus, we can draw a random variable $R = F(X)$ uniformly distributed between 0 and 1, and find the corresponding X.

We then have:

$$R = 1 - e^{-\lambda X} \Leftrightarrow 1 - R = e^{-\lambda X} \Leftrightarrow \ln(1 - R) = -\lambda X \Leftrightarrow X = -\frac{1}{\lambda} \ln(1 - R)$$

Therefore, inter-arrival or service durations following an exponential law with parameter λ can be expressed as:

$$t_{i+1} = t_i + \frac{-\log(1 - x)}{\lambda} \qquad (x : \text{uniform random variable in }]0, 1[)$$

Events and Event List (Scheduler)

In the case of an M/M/1 queue, there are two types of events: arrivals and departures. To initialize the event list, simply place one arrival event at $t = 0$.

Processing an arrival event for customer n triggers the creation of two new events:

- the future arrival of customer $n + 1$ (inter–arrival times follow an exponential law with parameter λ, see previous section). One of your simulation parameters will be the simulated horizon; therefore, do *not* create arrival events whose time would be later than the end of the simulation, otherwise the simulation would never terminate;
- the future departure of customer n (service time follows an exponential law with parameter μ, see previous section).

  To predict the departure time, there are two cases:

  - if the queue is empty: this is easy—draw a service time and add it to the current time (i.e., the arrival time of customer n);
  - if the queue is not empty: the departure time of customer n then depends on the departure time of customer $n - 1$. To simplify matters, since arrivals are handled in order, you may use a variable that stores the latest (most recent) computed departure time so far.

The simulation ends when the event list is empty.

Implementation

You will implement the simulation process in Java by creating the following classes:

- `MM1`: main class;
- `Evt`: event class;
- `Ech`: event-list (scheduler) class. You may use the Java API class `LinkedList` to store `Evt` objects in a linked list;
- `Stats`: class that collects and displays the simulation results;

- `Util`: class grouping all useful static methods (random draws, exponential law, ...).

The input parameters will be λ, μ, the simulated duration, and a switch to enable debugging mode (i.e., display every processed event).

Results

The objective of this simulation is to statistically recover certain theoretical values studied in class for the M/M/1 queue:

- Probability of service without waiting (empty queue upon an arrival):

$$1 - \frac{\lambda}{\mu}$$

- Utilization rate = probability that the server is busy:

$$\frac{\lambda}{\mu}$$

- Throughput:

$$\lambda$$

- Average number of customers in the system:

$$\frac{\rho}{1-\rho}$$

- Average time spent in the system:

$$\frac{1}{\mu(1-\rho)}$$

The simulation results are not exact: you can observe this by varying the simulation duration and measuring the difference from the theoretical results, as well as by calculating the variances or standard deviations of the results obtained from multiple simulations. Perform several runs and compute confidence intervals.

Example of Execution

```
> java MM1 5 6 1000 1
Date=0.0                Arrive  client #0
Date=0.03627054952825975 Depart client #0       arrive at t=0.0
Date=0.3129462895233852  Arrive  client #1
Date=0.3229906252413475  Depart client #1       arrive at t=0.3129462895233852
Date=0.424299442998096   Arrive  client #2
Date=0.575492786687346   Depart client #2       arrive at t=0.424299442998096
Date=0.8382389796147136  Arrive  client #3
Date=1.0334555371545838  Arrive  client #4
Date=1.0470496555022897  Depart client #3       arrive at t=0.8382389796147136
Date=1.220592510488126   Arrive  client #5
[...]
Date=999.7903751479797   Depart client #4992     arrive at t=998.0708779291505
Date=999.79041431310723  Depart client #4993     arrive at t=998.1454900318407
Date=999.8132794264721   Arrive  client #5002
Date=1000.0116446615608  Depart client #4994     arrive at t=998.1885168192186
Date=1000.1378729402567  Depart client #4995     arrive at t=998.3940000355528
Date=1000.3294044541082  Depart client #4996     arrive at t=998.5175124230218
Date=1000.476042788417   Depart client #4997     arrive at t=998.8305172502718
Date=1001.075832604617   Depart client #4998     arrive at t=999.3667698695706
Date=1001.4573867204617  Depart client #4999     arrive at t=999.4639073372005
Date=1001.484174790929   Depart client #5000     arrive at t=999.5846147300545
Date=1001.5500147799029  Depart client #5001     arrive at t=999.7242365292011
Date=1001.5950008671978  Depart client #5002     arrive at t=999.8132179246271
```

THEORETICAL RESULTS

```
lambda < mu : stable queue
rho (lambda/mu) = 0.8333333333333334

Expected number of clients (lambda × duration) = 5000.0
Probability of service without waiting (1 - rho) = 0.16666666666666663
Server busy probability (rho) = 0.8333333333333334
Throughput (lambda) = 5.0
Expected number of clients (rho / (1 - rho)) = 5.000000000000002
Average residence time (1 / (mu(1 - rho))) = 1.000000000000002
```

SIMULATION RESULTS

```
Total number of clients = 5003
Proportion of clients served without waiting = 0.17489506296222265
Proportion of clients that waited = 0.8251049370377773
Throughput = 5.03
Average number of clients in the system = 3.98552146993969
Average residence time = 0.7966263181970213
```

What to Submit?

You will submit your work on Moodle, in the form of an archive containing:

Java Source Code

All of your commented Java source files; it is not necessary to include the compiled `.class` files.

Report

A PDF report covering the following points:

- Comparison between the simulation results and the theoretical results studied in class. Provide annotated figures showing the impact of certain parameters on the obtained simulation results and their deviation from the theoretical values. Explain and justify the statistical calculations you performed on your simulation results.
- Performance of your simulator: Explain what allowed you to reduce the simulation time (reference execution time on Turing with $\lambda = 5$, $\mu = 6$, `duration=10,000,000`, `debug=0`: between **7 and 11 seconds**).