

## **D5: Using a Java Program to do Manual Simulation**

Melissa Von Wald, ISAT 341 Spring 2013

### ***Objective***

To create a Java program that simulates the system described in D5 Lab.  
To be able to read the Java code, write a method, and update the code.

### ***Background***

Java is an object oriented programming language. It was originally developed by James Gosling at Sun Microsystems (now Oracle) in 1995 and today is one of the most popular programming languages. There were five primary goals when creating Java which reflect why it is so popular today.

1. Simple, object-oriented, and familiar
2. Robust and secure
3. Architecture-neutral and portable
4. Execute with high performance
5. Interpreted, threaded, and dynamic

Now it is not necessary to understand all these qualities as a first time programmer but as you get more familiar with Java you can clearly see the benefit of these characteristics. Object-oriented means that the programs are structured by objects (instances of classes). Architecture-neutral means Java applications are compiled to bytecode files that can be run on any computer with Java Virtual Machine (JVM). This machine independence makes Java very portable. Interpreted refers to how the applications are compiled and run. Java is object oriented. All code is written inside a class. Everything is an object except for primitive data types (like integer, floating-point, Boolean, and character). An example of an intro Hello World program is below.

```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!"); // Display the string.  
    }  
}
```

**Class:** How the data is organized.

**Public:** A keyword that denotes that a method can be called from code inside other classes.

**Static:** Static methods are associated only within the class. Static objects are the same for every instance of a class. Main method is always static and any variables or methods it accesses must also be static.

**Void:** In this example the keyword means that the main method does not return a value.

**Main:** The method the Java launcher initially calls. Must have an array of String objects as the parameter ( String[] args ) and it is usually called args.

**System.out.println :** A function that takes a string as the parameter ("Hello World!") and outputs that string plus a new line character to the console.

The D5 simulation lab is also a major part of this assignment. The event chart is shown below. Make sure you understand what each of the simulation variables and summary statistics are and how to calculate them by hand. Computer programs can automate processes previously done by hand. For example, the computer program to simulate D5 takes less than .001 milliseconds where it would take a student at least 5 minutes to complete the simulation. However, the computer only does what it is told to do and it needs to be told exactly what to calculate and print out as results.

### ***Data Format***

For this exercise we will use the chart given in lab D5. The goal of this exercise is to perform a simple discrete-event simulation with a Java application. There is computer software that would schedule events, keep track of impending events, execute those events, update the simulation clock, and update the system statistics. We are going to create a simpler version of the software.

The characteristics of the system are the same as in D5 Lab. This system consists of only one process. Interarrival times and service times are stochastic. Arrivals and departures will be determined by the random number streams shown in the chart below. Only one entity can be in the machine at a time and entities in the queue enter the machine at exactly the same time entities in the machine are completed (and depart). Arrivals and departures are to be handled as separate events. If two events are scheduled to happen at the same time, always do the Departure first before the Arrival.



## ***Code***

The code is given in the Simulation.java file. Now we will step through how to use Eclipse to create and run a Java program.

Open up Eclipse by clicking Start and searching for eclipse. When the welcome screen appears click on the arrow labeled 'Go to workbench'.

To create a new Java project go to File -> New -> Java Project. Type in the Project name: D5\_Lab and click Finish. You should now see 'D5\_Lab' in the 'Package Explorer' on the left hand side of the screen.

To create a new Java class, right click on 'D5\_Lab' -> New -> Class. Type in the Name: Simulation. Note: It is very important that you name it exactly 'Simulation' and it is case sensitive. Click finish and now if you look in the 'Package Explorer' on the left hand side of the screen you will see 'Simulation.java' listed under '(default package)' under 'src' under 'D5\_Lab'.

Now open up the files provided with this lab Simulation.java. Copy and paste the code into the open files in Eclipse. Note: make sure when you paste the code you overwrite all the code in the file. There may be yellow underlines and yellow boxes- those symbolize warnings. There should not be any angry red underlines or red boxes- those symbolize errors.

Now you are ready to run the simulation. On the top menu bar navigate to 'Run' -> 'Run As' -> 'Java Application'. A shortcut is 'Run'->'Run' or Control-F11.

In the console box at the bottom of the screen you will see the output of the run. The application is formatted so the results appear as they do in the D5 Manual Simulation Chart.

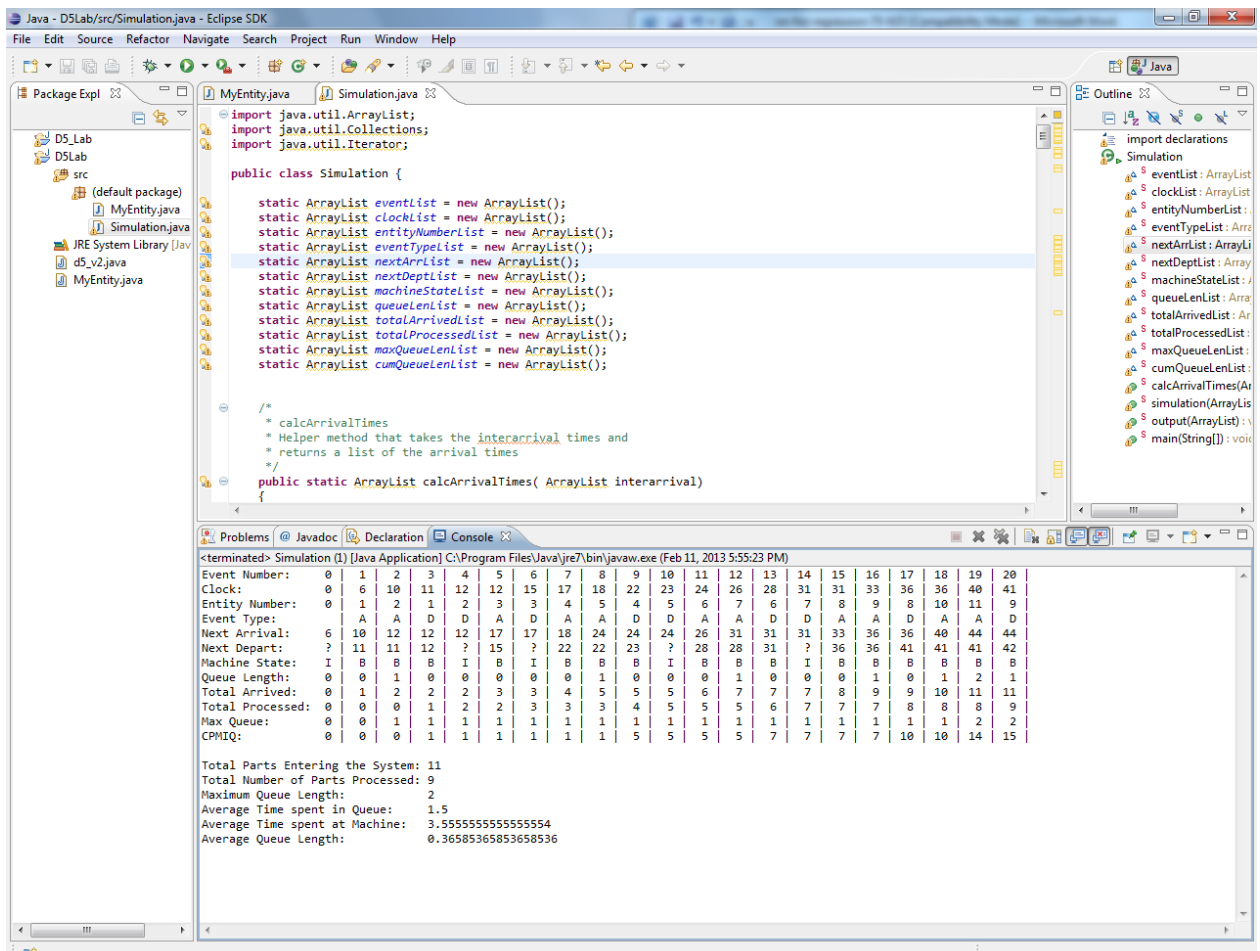
## Outline of the code:

The file contains two classes: Entity and Simulation. Entity is a simple class with attributes and no methods. Entity objects can be created by the line `Entity a = new Entity()` line #83 and the attributes are set and accessed by the dot operator, for example: `a.startTime` and `a.serviceTime`.

Look inside the class file Simulation. You will first see a list of static ArrayLists that will hold the data that is printed out at the end of the program. Then there are four methods/functions. The program begins in the main method line #342 with the header `'public static void main(String[] args)'`. Main method calls the function simulation.

## Results

Your results should be the same as the results from your D5 lab.



```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;

public class Simulation {

    static ArrayList eventList = new ArrayList();
    static ArrayList clockList = new ArrayList();
    static ArrayList entityNumberList = new ArrayList();
    static ArrayList eventTypeList = new ArrayList();
    static ArrayList nextArrList = new ArrayList();
    static ArrayList nextDeptList = new ArrayList();
    static ArrayList machineStateList = new ArrayList();
    static ArrayList queueLenList = new ArrayList();
    static ArrayList totalArrivedList = new ArrayList();
    static ArrayList totalProcessedList = new ArrayList();
    static ArrayList maxQueueLenList = new ArrayList();
    static ArrayList cumQueueLenList = new ArrayList();

    /*
     * calcArrivalTimes
     * Helper method that takes the interarrival times and
     * returns a list of the arrival times
     */
    public static ArrayList calcArrivalTimes (ArrayList interarrival)
    {
        ...
    }
}
```

terminated- Simulation [1] [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Feb 11, 2013 5:55:23 PM)

Event Number:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Clock:	0	6	10	11	12	12	15	17	18	22	23	24	26	28	31	31	33	36	36	40	41	
Entity Number:	0	1	2	1	2	3	3	4	5	4	5	6	7	6	7	8	9	8	10	11	9	
Event Type:		A	A	D	D	A	D	A	A	D	D	A	A	D	D	A	A	D	A	A	D	
Next Arrival:	6	10	12	12	17	17	18	24	24	24	26	31	31	31	33	36	36	40	41	41	44	
Next Depart:	?	11	11	12	?	15	?	22	22	23	?	28	28	31	?	36	36	41	41	41	42	
Machine State:	1	8	8	8	1	8	1	8	8	8	1	8	8	8	1	8	8	8	8	8	8	
Queue Length:	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	1	2	1	
Total Arrived:	0	1	2	2	2	3	3	4	5	5	5	6	7	7	7	8	9	10	11	11	11	
Total Processed:	0	0	0	1	2	2	3	3	3	4	5	5	6	6	7	7	7	8	8	8	9	
Max Queue:	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	
CPMIQ:	0	0	0	1	1	1	1	1	1	1	5	5	5	5	7	7	7	7	10	10	14	15

Total Parts Entering the System: 11  
Total Number of Parts Processed: 9  
Maximum Queue Length: 2  
Average Time spent in Queue: 1.5  
Average Time spent at Machine: 3.555555555555554  
Average Queue Length: 0.36585365853658536

## ***Conclusions***

1. How many attributes are in the Entity class?
2. How many functions are in the Simulation class?
3. What does `queue.get(0)` represent?
4. What does `maxQueueLenList.get(maxQueueLenList.size()-1)` represent?
5. How is the requirement “always do the departure before the arrival” expressed in the code?
6. Without changing the code, think about what line(s) of code would you change if you realized the first entity in our simulation arrives at  $t=1$  not  $t=6$ ?
7. Without changing the code, think about what line(s) of code would you change if you wanted the length of the simulation to be in terms of time instead of events?
8. What does this if statement mean in the context of our simulation?

```
if(nextArrival == -1 && nextDeparture == -1)  
    return;
```

If there are no more arrivals scheduled and there are no more items in the system to depart, then return null which ends the simulation function.

9. An important part of coding is writing readable code. Throughout the Java file you will see comments.

Commented lines begin with `// comment`

Or a commented block is surrounded by `/* comment */`

Comments help explain confusing parts of the code. Are there any places that you find would be helpful to have comments to explain the code? Another part of creating readable code is giving meaningful variable names. Are there any variables that have names that are confusing or don't aid in identifying what the variable holds?

10. Currently the simulation runs correctly and gives the output. A concept of object-oriented programming is modularity. Modularity is separating sections of code. For example, the code to format and print the information to the console is separated in a function called `output`. Currently there are 6 lines of code (lines #360-366) to assign the list of arrival times. There is a blank function `calcArrivalTimes`. Fill in the function so if those 6 lines were deleted and line #46 that is commented out is run, the program runs correctly with no errors.

### ***Other Resources:***

Official Site: <http://docs.oracle.com/javase/tutorial/>

Helpful tutorial: <http://computer.howstuffworks.com/program.htm>

Overview: [http://en.wikipedia.org/wiki/Java\\_\(programming\\_language\)](http://en.wikipedia.org/wiki/Java_(programming_language))