CS102 – Data Structures

**Final Project: Theme Park Waiting Line**

**Name:** Namrata Jagannath

**ID#:** 1016420

**Instructor:** Subrina Thompson

**Date:** Dec 15, 2016

**Table of Content:**

Abstract....................................................................................................................3

Introduction..............................................................................................................4

Screenshot................................................................................................................5

Source Code..............................................................................................................6

Conclusion..............................................................................................................13

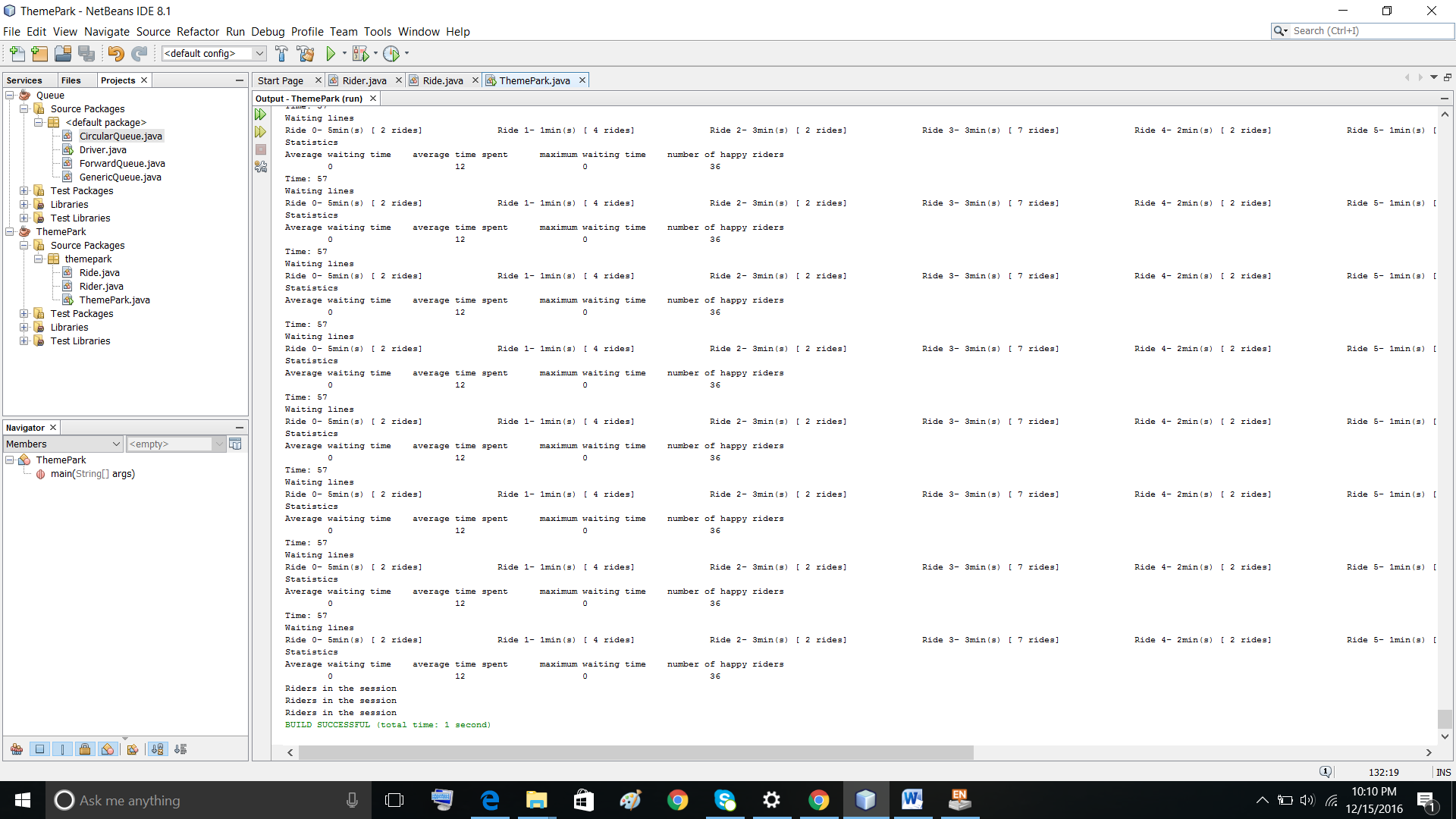
**Abstract**

To write a program that will use queues to simulate a Theme Park Waiting Line for n number of rides. Guests will arrive at the Theme Park at an average rate (probability). Each ride will take a certain amount of time, according to a random distribution. As each guest arrives, they will decide which ride to go on (including fast pass access). A newly arriving guess will randomly pick a ride to go on. The object then is to find a most popular ride before the Theme Park closes. Since probability is a factor, we will assume that “one hour” of operation to analyze our objective.

**Introduction**

This assignment introduces a program that will use queues to simulate a Theme Park Waiting Line for n number of rides. Guests will arrive at the Theme Park at an average rate (probability).

**Screenshot :**

****

**Source Code**

**(Ride class)**

package themepark**;**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\* @param <T>

\* @author Namrata Jagannath

\*/

public final class Ride **<**T**>{**

private T**[]** Regular**;**

private T**[]** FastTrack**;**

private int rearFast**;**

private int rearRegular**;**

private int frontFast**;**

private int frontRegular**;**

private int size\_t**;**

public int rideTime**;**

public int totalbusytime**;**

public float averageTimeSpent**;**

public int maxWaitingTime**;**

public int averagewaitingTime**;**

public int ridercount**=**0**;**

public Ride**(){**

rearRegular**=**0**;**

frontRegular **=** 0**;**

rearFast**=**0**;**

frontFast **=** 0**;**

size\_t **=** 120**;**

rideTime**=**3**;**

totalbusytime**=**0**;**

averageTimeSpent**=**0**;**

maxWaitingTime**=**0**;**

averagewaitingTime**=**0**;**

InitializeQ**();**

**}**

public Ride**(**int size\_t**,**int t**){**

**this.**size\_t**=**size\_t**;**

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

InitializeQ**();**

**}**

public void InitializeQ**()**

**{**

FastTrack **=** **(**T**[])** **new** Object**[**size\_t**];**

Regular **=(**T**[])** **new** Object**[**size\_t**];**

**}**

public boolean enQueueRegular**(**T x**)**

**{**

**if(**rearRegular**==**size\_t**-**1**)**

**{**

System**.**out**.**println**(**"Queue is full"**);**

**}**

**else**

**{**

ridercount**++;**

Regular**[**rearRegular**++]=**x**;**

**return** **true;**

**}**

**return** **false;**

**}**

public T deQueueRegular**()**

**{**

**if(**frontRegular**<**rearRegular**)**

**{**

frontRegular**++;**

**return** Regular**[**frontRegular**-**1**];**

**}**

**else**

System**.**out**.**println**(**"Queue is empty"**);**

**return** **null;**

**}**

public T lookRegular**()**

**{**

**if(**frontRegular**<**rearRegular**)**

**{**

**return** Regular**[**frontRegular**];**

**}**

**return** **null;**

**}**

public T lookFast**()**

**{**

**if(**frontRegular**<**rearRegular**)**

**{**

**return** FastTrack**[**frontRegular**];**

**}**

**return** **null;**

**}**

public boolean enQueueFast**(**T x**)**

**{**

**if(**rearFast**==**size\_t**-**1**)**

**{**

System**.**out**.**println**(**"Queue is full"**);**

**}**

**else**

**{**

ridercount**++;**

Regular**[**rearFast**++]=**x**;**

**return** **true;**

**}**

**return** **false;**

**}**

public T deQueueFast**()**

**{**

**if(**frontFast**<**rearFast**)**

**{**

frontFast**++;**

**return** Regular**[**frontFast**-**1**];**

**}**

**else**

System**.**out**.**println**(**"Queue is empty"**);**

**return** **null;**

**}**

public void display**()**

**{**

System**.**out**.**println**(**"Regular Queue contents"**);**

**for** **(**int i **=** frontRegular**;** i **!=** rearRegular**;** i**=(++**i**)%**size\_t**)** **{**

System**.**out**.**println**(**"["**+**i**+**"] ="**+** Regular**[**i**]);**

**}**

System**.**out**.**println**(**"Fast Queue contents"**);**

**for** **(**int i **=** frontFast**;** i **!=** rearFast**;** i**=(++**i**)%**size\_t**)** **{**

System**.**out**.**println**(**"["**+**i**+**"] ="**+** FastTrack**[**i**]);**

**}**

**}**

**}**

**(Rider class)**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package themepark**;**

/\*\*

\*

\* @author Namrata Jagannath

\*/

public class Rider **{**

private String rider**;**

private int Ride**;**

private int ArrivalTime**;**

public void setrider**(**String s**)**

**{**

rider**=**s**;**

**}**

public String getrider**()**

**{**

**return** rider**;**

**}**

public void setRide**(**int r**)**

**{**

Ride**=**r**;**

**}**

public int getRide**()**

**{**

**return** Ride**;**

**}**

public void setArrivalTime**(**int at**)**

**{**

ArrivalTime**=**at**;**

**}**

public int getArrivalTime**()**

**{**

**return** ArrivalTime**;**

**}**

public Rider**(**String s**,** int ride**,** int arrival**)**

**{**

setrider**(**s**);**

setArrivalTime**(**arrival**);**

setRide**(**ride**);**

**}**

**}**

**(Theme Park Main class)**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package themepark**;**

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

/\*\*

\*

\* @author Namrata Jagannath

\*/

public class ThemePark **{**

/\*\*

\* @param args the command line arguments

\*/

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

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

int n **=** rand**.**nextInt**(**10**)** **+** 1**;**

Ride**<**Rider**>[]** Rides **=** **new** Ride**[**n**];**

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

int t **=** rand**.**nextInt**(**5**)** **+** 1**;**

Rides**[**i**]=new** Ride**(**5**,**t**);**

**}**

System**.**out**.**println**(**"Number of rides: "**+**n**);**

**for** **(**int time **=** 1**;** time **<=** 60**;** time**++)** **{**

n **=** rand**.**nextInt**(**100**)** **+** 1**;**

System**.**out**.**println**(**"Riders in the session"**);**

**if(**n**<**50**)**

**continue;**

**else** **if(**n**<**90**)**

**{**

String s **=** "Rider"**+**time**;**

n **=** rand**.**nextInt**(**Rides**.**length**);**

Rider r **=** **new** Rider**(**s**,** n**,** time**);**

int prob **=** rand**.**nextInt**(**100**)+**1**;**

**if(**prob**<**50**)**

Rides**[**n**].**enQueueFast**(**r**);**

**else**

Rides**[**n**].**enQueueRegular**(**r**);**

System**.**out**.**println**(**s**+**"( "**+**time**+**"\_"**+**prob**+**") => Ride "**+**n **);**

**}**

**else**

**{**

**for** **(**int j **=** 0**;** j **<**2**;** j**++)** **{**

String s**=**"Rider"**+**time**\***j**;**

n **=** rand**.**nextInt**(**Rides**.**length**);**

Rider r **=** **new** Rider**(**s**,** n**,** time**);**

int prob **=** rand**.**nextInt**(**100**)+**1**;**

**if(**prob**<**50**)**

Rides**[**n**].**enQueueFast**(**r**);**

**else**

Rides**[**n**].**enQueueRegular**(**r**);**

System**.**out**.**println**(**s**+**"( "**+**time**+**"\_"**+**prob**+**") => Ride "**+**n **);**

**}**

**}**

Rider t**;**

**for** **(**int j **=** 0**;** j **<** Rides**.**length**;** j**++)** **{**

t**=** Rides**[**j**].**lookFast**();**

**if(**t**!=null)**

**if((**t**.**getArrivalTime**()+**Rides**[**j**].**rideTime**)>=**time**)**

**{**

Rides**[**j**].**averageTimeSpent**=**Rides**[**j**].**averageTimeSpent**+((**time**-**t**.**getArrivalTime**())+**Rides**[**j**].**rideTime**)/**Rides**[**j**].**ridercount**;**

Rides**[**j**].**averagewaitingTime **=** Rides**[**j**].**averagewaitingTime**+(**time**-**t**.**getArrivalTime**())/**Rides**[**j**].**ridercount**;**

Rides**[**j**].**totalbusytime**+=**Rides**[**j**].**rideTime**;**

**if(**Rides**[**j**].**maxWaitingTime**<(**time**-**t**.**getArrivalTime**()))**

Rides**[**j**].**maxWaitingTime**=(**time**-**t**.**getArrivalTime**());**

Rides**[**j**].**deQueueFast**();**

**}**

t**=** Rides**[**j**].**lookRegular**();**

**if(**t**!=null)**

**if((**t**.**getArrivalTime**()+**Rides**[**j**].**rideTime**)>=**time**)**

**{**

Rides**[**j**].**totalbusytime**+=**Rides**[**j**].**rideTime**;**

Rides**[**j**].**averageTimeSpent**=(**Rides**[**j**].**averageTimeSpent**+(**time**-**t**.**getArrivalTime**())+**Rides**[**j**].**rideTime**)/**Rides**[**j**].**ridercount**;**

Rides**[**j**].**averagewaitingTime **=** **(**Rides**[**j**].**averagewaitingTime**+**time**-**t**.**getArrivalTime**())/**Rides**[**j**].**ridercount**;**

**if(**Rides**[**j**].**maxWaitingTime**<(**time**-**t**.**getArrivalTime**()))**

Rides**[**j**].**maxWaitingTime**=(**time**-**t**.**getArrivalTime**());**

Rides**[**j**].**deQueueRegular**();**

**}**

print**(**time**,**Rides**);**

**}**

**}**

**}**

public static void print**(**int time**,**Ride**<**Rider**>[]** Rides**)**

**{**

System**.**out**.**println**(**"Time: "**+**time**);**

int taveragewaitingTime**=**0**;**

int taverageTimeSpent**=**0**;**

int tmaxWaitingTime**=**0**;**

int thr**=**0**;**

System**.**out**.**println**(**"Waiting lines"**);**

**for** **(**int k **=** 0**;** k **<** Rides**.**length**;** k**++)** **{**

taveragewaitingTime**+=**Rides**[**k**].**averagewaitingTime**;**

taverageTimeSpent**+=**Rides**[**k**].**averageTimeSpent**;**

tmaxWaitingTime**+=**Rides**[**k**].**maxWaitingTime**;**

thr**+=**Rides**[**k**].**ridercount**;**

System**.**out**.**print**(**"Ride "**+**k**+**"- "**+**Rides**[**k**].**rideTime**+**"min(s) [ "**+**Rides**[**k**].**ridercount**+**" rides] \t\t"**);**

**}**

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

System**.**out**.**println**(**"Statistics"**);**

System**.**out**.**println**(**"Average waiting time\taverage time spent\tmaximum waiting time\tnumber of happy riders"**);**

System**.**out**.**println**(**"\t"**+**taveragewaitingTime**+**"\t\t\t"**+**taverageTimeSpent**+**"\t\t\t"**+**tmaxWaitingTime**+**"\t\t\t"**+**thr**);**

**}**

**}**

**Conclusion**

I was able to demonstrate the program and determine the path successfully.