Project 3

Due Monday April 3, 2017

Lucy's Tattoo Parlor

Customer Description:

Welcome to Lucy's Tattoo Parlor, home of responsible tattooing. Around here, we've noticed that we get lots of people walking in, late at night, to get tattoos. But we want to be sure they don't regret it, so we want a computer system that lets each customer sign up to get a tattoo....the next night. We need you to write us a program that will do that. Here's how it has to work:

When we run the program at first, we have to tell it how many tattoo artists are going to be on duty the next night, and how many customers per tattoo artist can be put into the waitlist. We'll give those things as command-line arguments. All tattoo artists have the same capacity so they will all have the same size waitlist.

Then, once things are set up, the program will tell us to enter the name of the customer to add, along with what that customer wants a tattoo of. It also needs to let us enter whether the customer wants a specific tattoo technician (technician 0 or technician 1 or technician 3 for example), or if the customer just wants us to add them to the shortest waitlist among all the technicians. After requesting the artist, the user must then enter how many minutes that tattoo is expected to take. If a tech's waitlist is full or if the total estimated minutes would add up to more than 8 hours with this customer added, then the customer cannot be added to it. If all waitlists are full, no more customers can be added. If a customer asks to be added to a specific technician and that technician's waitlist is full because of number of customers or minutes, then the customer will not be added and the system will print an error message.

At the end of the night, we must be able to enter a special customer name, "Print Waitlist" and that name will cause the program to print out the waitlists of all tattoo artists on the screen, and then the program will end.

Implementation Details

We already consulted with an expert, Layla, a software engineer from a big computer science firm, and she made a design for our program. But she couldn't keep working on it, so you have to take over! Here are the requirements:

The program must be run with specifying how many artists you have, and also the maximum number of customers that you will allow to be in line for each artist at a time. Those are command line arguments. Note that the artist number start at 0.

You must use a 2-D array with a row for every artist. Each row has all the customer names that could be in line for that artist.

The program lets the user enter customers and add them to the 2-d array either for a specific artist (they could request artist 1 or artist 4 for example) or add them to whatever artist has the shortest wait at the moment.

If an artist's line is filled, nobody else can be placed in line for that artist. If all artist's lines are filled, nobody else can be placed in line at all.

Any empty spots(spots with no customer) in the array must be null

Rubric:

Category	**Points**
Menu – all functional elements present (manual test cases)	20
Menu – clarity and visual appeal	10
Junit test cases (computeMinutesOfWork (10), addCustomer (20), TattooCustomer constructor (10), Tattoo getters (10)	50
Code style (formatting, variable names, etc.)	10
Javadoc	10
Total points	**100**

You must create 2 classes for this project. LucyTattooParlor (LucyTattooParlor.java) and TattooCustomer (TattooCustomer.java).

In the class that contains your main method, you **must** implement the following methods:

```
/**
* Computes how many minutes of work the specified tattoo artist has.
* @param The array of customers for one particular tattoo artist
*/
public static int computeMinutesOfWork(TattooCustomer [] a) {
        //TODO
        return 0;
}

/**
* Adds customer to the waitlist for a specific artist.
* If the artist is at capacity (in terms of number of customers or minutes)
* Then the customer is not added and the method returns false
* If the customer is successfully added the method returns true
* @param
*/
public static boolean addCustomer(TattooCustomer [][] a, TattooCustomer c, int artistNum) {
        //TODO
        return false;
}

/**
* Adds customer to the shortest waitlist in terms of minutes. If some artists have equal length waitlists
* then the customer is added to the lowest numbered artist. If no artist has space then the method does not
* add the customer, and returns false.
* TODO - finish this javadoc
```

```
	* @return true if the customer was added to the waitlist, false otherwise (if all artists were full)
	*/
	public static boolean addCustomer(TattooCustomer [][] a, TattooCustomer c) {
		//TODO
		return false;
	}
```

Do **NOT** modify the method signatures in any way (or you will fail the junit test). Please note that computeMinutesOfWork method takes a 1-D Array, this is on purpose.

In the class TattooCustomer, you must create a constructor and getters for all **private** variables. The constructor must take a string, a string, an int as its parameters (representing name, tattoo, minutes in that **exact** order). The getters should be called getName, getMinutes, getTattoo. Note that the tattoo and name must be strings, and the minutes must be an integer.