SOS Application

Submission Date: 7th of April 11:59 PM

Contents

1 Introduction

Submit a **zipped folder** that is **only** containing your Java source files (*.java) in course's Black Board.

Please use the following naming convention for the submitted folders:

YourPSLetter_CourseCode_Surname_Name_HWNumber_Semester
Example folder names:

- **PSA_COMP130_Surname_Name_HW3_S19**
- **PSB_COMP130_Surname_Name_HW3_S19**

Additional notes:

- Using the naming convention properly is important, **failing** to do so will be **penalized**.
- **Do not** use Turkish characters when naming files or folders.
- Submissions with unidentifiable names will be **disregarded** completely. (ex. "homework1", "project" etc.)
- Please **write your name** into the Java source file where it is asked for. **Failing** to do so will be **penalized**.
- If you are resubmitting to update your solution, simply append **v#** where # denotes the resubmission version. (i.e. **v2**)

1.1 Academic Honesty

Koç University's *Statement on Academic Honesty* holds for all the homework given in this course. Failing to comply with the statement will be penalized accordingly. If you are unsure whether your action violates the code of conduct, please consult with your instructor.

1.2 Aim of The Homework

The aim of this project is to allow you to practice expressions, control statements, random numbers and graphical objects in Java.

1.3 Given Code

You are provided a code which has the necessary setup. It contains helper methods; variables and constants for you to start with.

1.3.1 Given Methods

There are methods provided for each part. It's up to you whether or not to use these methods. You may implement methods by yourself in the Helper Methods section. If you decide not to use some methods provided, please **make sure that every part is handled with at least one method**.

1.3.2 Given Variables and Constants

There are some variables and constants provided to you. Feel free to use the variables declared in the given code, meaning you can either use them or create new ones if you need to. HOWEVER, you need to **use the constant variables** provided to you. You can create additional constants if you need.

1.4 Further Questions

For further questions **about the project** you may send an email to **course SLs** at (comp198-spring-19-sl-group@ku.edu.tr] and **Ayca Tuzmen** at [atuzmen@ku.edu.tr]. Note that it may take up to 24 hours before you receive a response so please ask your questions **before** it is too late. No questions will be answered when there is **less than two days** left for the submission.

2 Task

You are asked to write a Graphics Program which simulates a SOS application. The program shall simulate people on a map, and capture SOS help requests and allow people who would like to respond and help out SOS seekers.

The program will capture SOS request of a person who needs help. The SOS signal will be sent out to the nearest 2 people. The program shall ask these people if they can help or not. According to the answer they give, they will either not help out and stay where they are, or they will be directed to the help seeker.

2.1 Part 1 - Creating The Map

Before we can have people use this program, we need the map of where they live. Luckily, they live in a very simple and tidy city, which is not that hard to map.

Our city is well-planned, its map should consist of several vertical and horizontal lines to indicate streets, creating a grid shape. The distance between two vertical lines is the same as the distance between two horizontal lines, this distance is given to you as a constant value.

TASK: You should write a helper method that creates the lines of the map using the given constant.

2.2 Part 2 - Adding People

After we have the map, we need people to start using our program. First, we will launch a closed alpha version and distribute our app to a specific amount of people, for testing purposes. There will be only 5 very special lucky people to have early-access to this program, and **you** need to add those!

You need to create users using circle (GOval object) to represent them. You need to create 5 circles and locate them on the map. The conditions for those circles are as in the following:

- Each circle should be sized as defined by the constant in the given code.

- Each circle should be stationed at a random intersection of a vertical and a horizontal line, i.e, each user sit on a random crossroad.
- Each circle should be colored **BLUE** to begin with and labelled with a number.
- The center of the circles should be considered the exact place of the user. Meaning, circles should be centered at the intersection of lines.

TASK: Write a helper method which creates circles with given conditions. **HINT:** You may use the instance variables declared in the given code for storing the circles assigned for each users.

2.3 Part 3 - Help

Now that we have our users set, we need to have one of them to ask for help and a few of them responding.

2.3.1 Step 1 - Who is in Need of Help?

One of the users will call for help, and our program should choose this user at randomly. Let us call this user in need of help *redUser*.

TASK: Write a helper method that picks a user at random and alerts us by painting the circle of chosen user to **RED**.

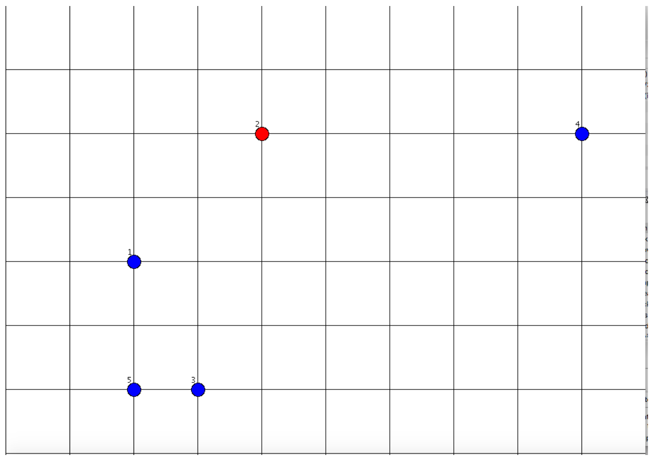


Figure 1: Map

2.3.2 Step 2 - Find The Helpers

Now that everybody knows *redUser* needs help, we should see who can help them out. Your program should;

- search for 2 users closest to the *redUser*.
- make this search in a square centered at the *redUser*, expanding the size of the squares each time in an animation.

- should continue searching until there are at least 2 users in the last SQUARE and should stop there. If it happens that there are more than 2 users found, the program should pick first 2 users found.
- should paint YELLOW the 2 users found.
- program should create squares increasing in side one by one and should display them in an animation.

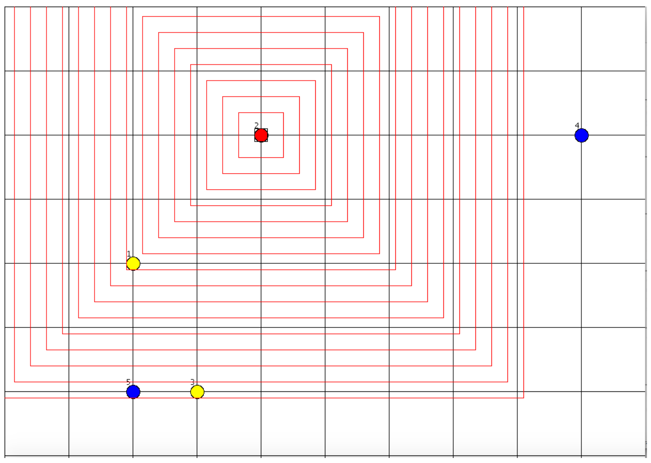
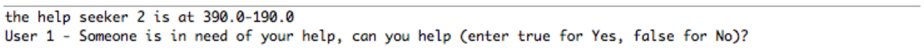


Figure 2: User in need of help is in RED and possible candidates who can help are in YELLOW

2.3.3 Step 3 - Will You Help?

Now that we have found 2 users around who are close by, we need to ask them to come for help. They have the right to choose whether or not to help; hence, we need to get the information, in this case, from the client (who runs your code) for testing purposes. Let us name them *helper1* and *helper2*.

TASK: Write a helper method which, asks the client for an input. A message is provided to the client and the input from the client is read from the console window.



```
the help seeker 2 is at 390.0-190.0
User 1 - Someone is in need of your help, can you help (enter true for Yes, false for No)?
```

Figure 3: Request for help

The client can respond the help request by entering **true** or **false**. Your program should keep on asking until the client provides true or false. Then, if the client agreed for *helper1* to help, *helper1* should change it's color to GREEN and move forward to the animation part. After its animation is done you should paint the *helper1* **RED** and repeat this procedure for *helper2*. If the user disagrees to help, the *helper* should be painted **BLUE** again and you should move forward with the other *helper*, without animating the *helper* that refused to help.

2.4 Part 4 - Animation

Assuming we have a user who agreed to help, they should move to the *redUser*.

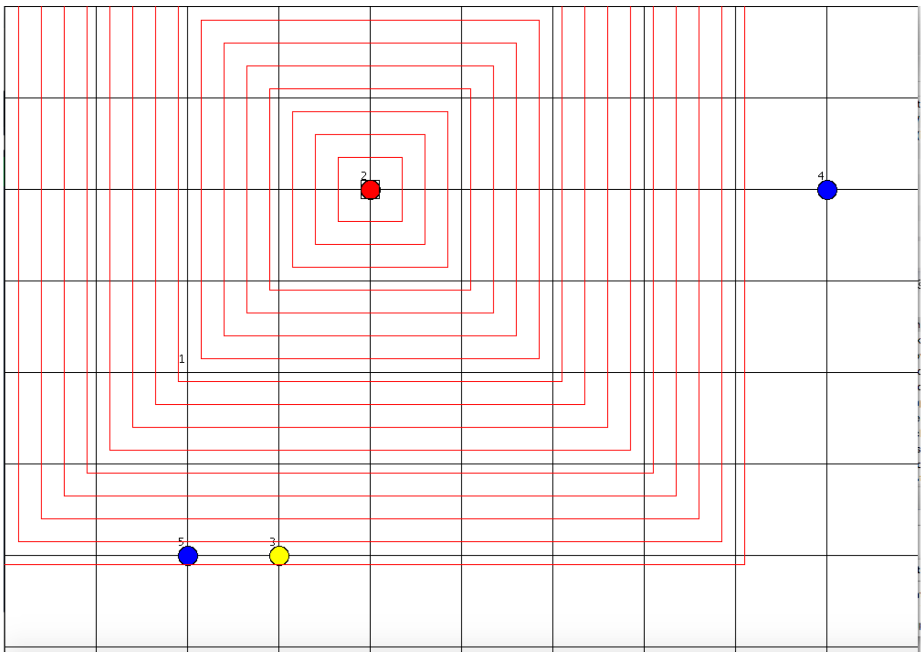


Figure 4: Request for help

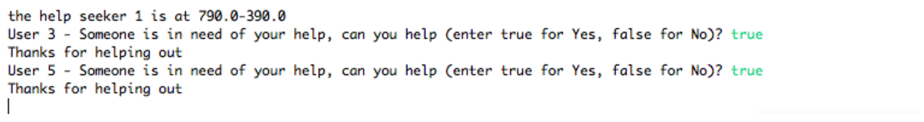
TASK: Write a helper method which moves the *helper* to the *redUser*. This moving process should satisfy the following conditions:

- The users can move only on the streets, so the circles' centers should always contact with a line.
- The *helper* should follow the closest path to the *redUser*, for example, if the *redUser* is to the northwest of the helper, the *helper* should never move towards South or East.
- The *helper* should follow a street until a crossing and then decide whether or not to turn at random, it should be possible for the helper to follow any of the closest paths to the *redUser*.

HINT: Write a helper method that moves the helper first in x direction and next in y direction or vice versa.

2.5 Part 5 - Did The Person Get Any Help?

If the *redUser* got help from at least one of the helpers, the program should thank the helper who has, of course, agreed to help.



```
the help seeker 1 is at 790.0-390.0
User 3 - Someone is in need of your help, can you help (enter true for Yes, false for No)? true
Thanks for helping out
User 5 - Someone is in need of your help, can you help (enter true for Yes, false for No)? true
Thanks for helping out
|
```

Figure 5: Thank you!

The program should print out "The person has been helped!", if there was at least one person between the two nearest who helped out. If no one decided to help, the program should print out "Nobody was available to help." and should paint the *redUser* to **BLACK**.

3 Bonus

This part is extra, you are not obligated to implement this; however, you can choose to do so for extra points. This is part where your imagination can go wild. Feel free to change the representation of the users, helpers, or help seeker in any way you like. You can apply an image, use a different shape for simulating them. However, you should allows use the same **COLORING STANDARD** so that we can easily grade your homework.