**Institute of Technology Tralee**

**Computing Department**

**Object Oriented Programming 1**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Session 4 – Review of some of last years CAs on methods**

A Java program must be written which will play 4 games of “Who is closer?” The main code for the game will take place within a user-defined method called **whoIsCloser**(). The main() here will just contain a loop that executes 4 times in total, and each time it executes it will make a call to the whoIsCloser() method.

The whoIsCloser() method **takes no arguments** and will first of all generate a random number between 1 and 100, using the **random**() method of the Math class (which is static). Then it will ask player 1 to guess what the number generated was. Next, it will ask player 2 to guess what the number generated was – of course, here you will be playing the part of both players when the game is being played. Once both players have entered their guesses, the method will determine which of the players was actually closer to the number that was generated. Once this has been determined, the method should return one of the following strings to main()

* “the generated number was <generated number> and the winner was player 1”
* “the generated number was <generated number> and the winner was player 2”
* “the generated number was <generated number> and neither player wins” - this last message gets returned if the difference between the generated number and the guess was the same for both players

Note that you can use a number of techniques to determine which of the guesses is closer to the randomly generated number. However, you are welcome to use the **abs**() method from the Math class to do the job (this is also a static method) – this method returns the **absolute value** of a number, so if the generated number was 45, and player 1 guessed the value 65, then the absolute value of the difference between the generated number and the guess would be **Math.abs(45-65)** = **Math.abs(-20)** **= 20**. This is very useful for comparing the guesses of the two players quickly to see which was closer to the generated value.

Once the whoIsCloser() method has returned its result, the main() then simply displays a message dialog to show the result of the game.

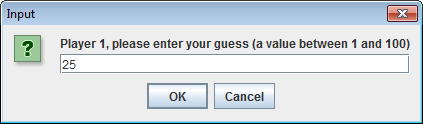
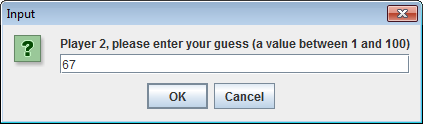
For full marks here your program should, along with a logically correct solution for the problem above, include the usual **single-line** and **multi-line comment** at the top of the program. The multi-line comment should briefly explain the purpose of the program.

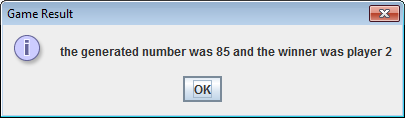
Your program should run as indicated in the following sample screenshots. The results from your program will not be exactly the same though, as random number generation is involved.

**Sample Screenshots**

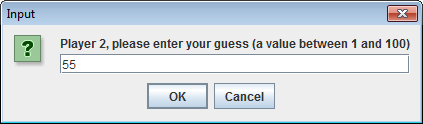
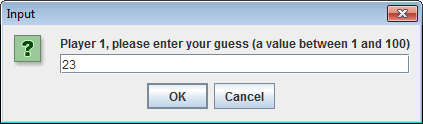
**Run 1**

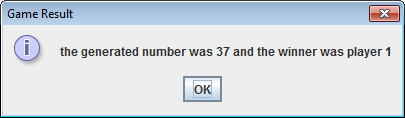
**1st Game:**

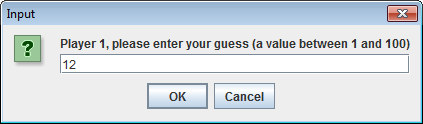
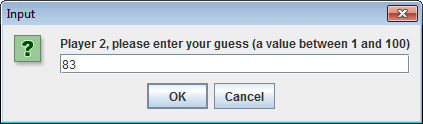


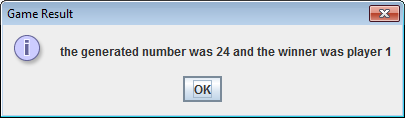
**2nd Game:**



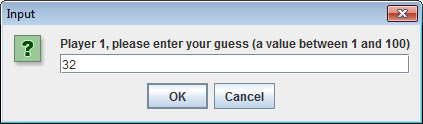
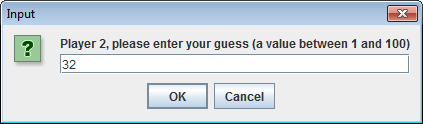


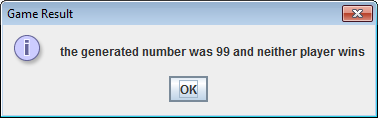
**3rd Game:**



**4th Game:**



A program is required that will first of all read in the x and y co-ordinates of the centre of a circle, along with its radius. It will then ask the user to enter the x and y co-ordinates of a second point. All the details to be entered will be read in within the main() method here and note that all these values can be **fractional**.

Next a user-defined method called **pointLocator**() will be called by main() to determine whether the second point lies inside, outside or on the circle. This method will take **5 numeric arguments** representing the x and y co-ordinates of the 2 points as well as the radius value. The method will use the following table to determine the location of the second point with respect to the circle:

|  |  |
| --- | --- |
| **Point Location** | **Distance between second point and circle centre** |
| Inside Circle | Less than radius |
| On Circle | Equal to radius |
| Outside Circle | Greater than radius |

The distance between the second point and the circle’s centre can be determined using the following formula:

**distance =**

where

is the x co-ordinate of the second point

is the y co-ordinate of the second point

is the x co-ordinate of the circle’s centre

is the y co-ordinate of the circle’s centre

You can use the **Math** class’ **sqrt**() method in carrying out this calculation. Note that this method had the following header in the Java API:

**public static double sqrt(double x)**

Once the pointLocator() method has determined the location of the second point relative to the circle, it will return the outcome from the method as a string. The three possible strings that can be returned from the method are “inside”, “on” and “outside”.

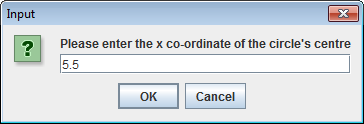
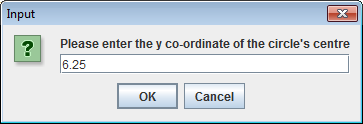
Once the outcome has been returned to main(), then the main() itself should display a message dialog to indicate the result.

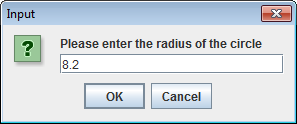
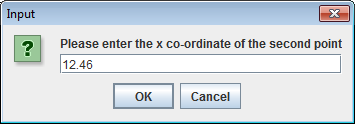
For full marks here your program should, along with a logically correct solution for the problem above, include the usual **single-line** and **multi-line comment** at the top of the program. The multi-line comment should briefly explain the purpose of the program.

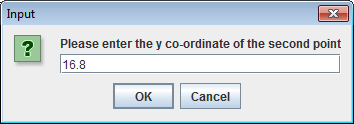
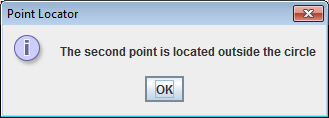
Your program should run exactly as indicated in the following sample screenshots. Also, you should use the test values indicated when testing your own program.

**Sample Screenshots**

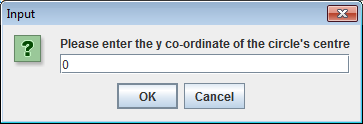
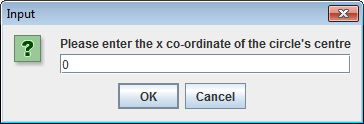
**Run 1 – The second point lies outside the circle**

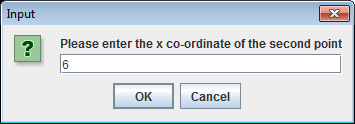
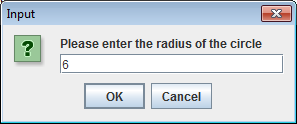
 

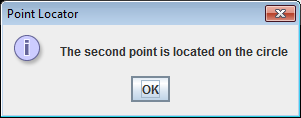
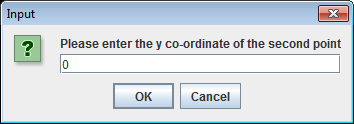
 

**Run 2 – The second point lies on the circle**







**Run 3 – The second point lies inside the circle**

Supply the following values here (input screenshots omitted for brevity)

0 for the x co-ordinate of the circle’s centre

0 for the y co-ordinate of the circle’s centre

6 for the radius of the circle

5.9 for the x co-ordinate of the second point

0 for the y co-ordinate of the second point

These inputs should produce the following output:

