HW 03: Player Model

Release date: Friday, September 9th, 2016
Due date: Friday, September 16th, 2016 – 11:59 pm

Goals

- ★ To get familiar with using Java's standard library APIs
- ★ To understand the structure of Java classes
- ★ To learn techniques for comparing the equality of floating point values

Prerequisites

1. Knowledge of the Java standard libraries classes like Math and String classes
2. Knowledge of the basic syntactic structure of classes, the use of multiple constructors, and class methods calling

Description

In this homework assignment, you will write a class called "Player". Each Object of this class represents a Player entity. A Player has a name and a position in the playground. The playground is a cartesian plane; hence the position of the player consists of the x and y coordinates. The name and the position of the Player will be stored in the data members of the Object. **Figure 1** shows the location representation of the Player.

Methods in the Player class allow accessing and changing the data members of a Player Object. Changing the position of the Player can be done in two different ways. The first is to provide the difference between the old position and the new position of the Player in each of the X and Y directions. The second way is to provide the direction in which the Player moves; this is represented as a vector that starts at the old location of the Player and ends at his/her new position. Methods in the Player class will also allow measuring the distance between two players and checking whether both Players are at the same position.

You will also implement a main method to create Objects of the Player class and manipulate their location information.

Details on the Player class and the main method are given in the **Requirements Section.**

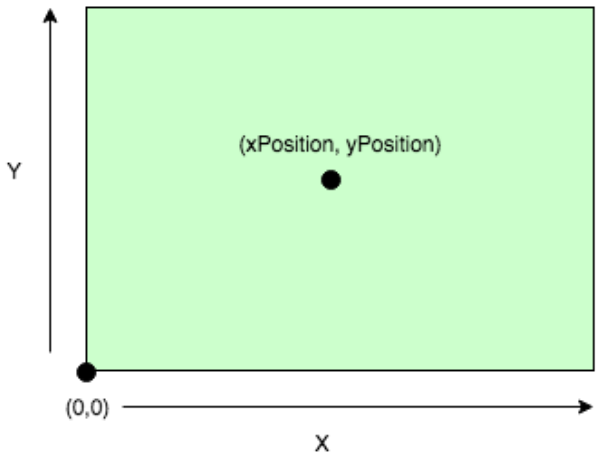

Figure 1: Location representation of a player

Requirements
You are required to write a program that matches the description given above. Exact details about the required methods are discussed next. You **must follow** the given naming for (1) files, (2) classes, (3) data members and (4) methods.

1) Files to develop
You are required to create a single file named "Player.java" that contains the "Player" class. The main method creates Objects of the Player class and calls methods on these objects to change the Players' position information as will be described later.

2) The Player class
The Player class is a public class. An Object of this class represents one Player entity. **Table 1** shows the data members of the Player class. **Table 2** shows the constructors of the Player class. **Tables 3 and 4** have the prototypes and descriptions of the accessors and mutators of the Player class respectively. **Table 5** contains prototypes and descriptions of other Player class methods. You are required to implement all the methods in **Tables 3, 4 and 5.**

3) The main method
You will need to include a main method to execute the game. The main method starts by getting the information of the players one at a time *(i.e., it starts by asking the user to input the name, X position and Y position of the first Player)*. After getting the information of the two Players, the main method creates two Objects of type Player.

Then, the main simulates three moves of the players. The first move is done by each players in the X-direction. The second move is done by each Player in the Y-direction. Each Player then moves in a specific direction provided by the user. The main method then finds the distance between the two Players and checks if the positions of the two Players are the same. A demo of a correctly-functioning program is shown in **Figure 4.**

Field Name	Type	Access modifier	Description
`name`	String	private	Contains the name of the player
`positionX`	double	private	Contains the horizontal distance from the origin to the position of the player (See Figure 1)
`positionY`	double	private	Contains the vertical distance from the origin to the position of the player (See Figure 1)

Table 1: Player class data members

Constructor prototype	Description
`public Player(String name)`	Sets the *name* of the player to the given name and sets the position of the player to the default position, which is (0, 0). This constructor calls the next constructor with the proper set of parameters
`public Player(String name, double positionX, double positionY)`	Sets the name of the player to the given name and sets the position of the player to the position (*positionX*, *positionY*)

Table 2: Player class constructors

Method prototype	Descrition
`public String getName()`	Accessor for the field *name*
`public double getPositionX()`	Accessor for the field *positionX*
`public double getPositionY()`	Accessor for the field *positionY*

Table 3: Player class accessors

Method prototype	Descrition
`public void setName(String name)`	Modifies the *name* of the player to the given name

Table 4: Player class mutators

Method prototype	Description

`public void moveX (double offsetX)`	Changes the X-position of the player by adding the offset value (*offsetX*) to the data member of the player *positionX* (See **Figure 2**)
`public void moveY (double offsetY)`	Changes the Y-position of the player by adding the offset value (*offsetY*) to the data member of the player *positionY* (See **Figure 2**)
`public void moveInDirection (double theta, double distance)`	Simulates the movement of the player from his current position to a new position in a specific direction. The angle of the direction of the move is given in the parameter theta. **Theta is expressed in degrees** and is measured in a counterclockwise manner from the X-axis as shown in **Figure 3**. The distance moved in the direction given is passed in the parameter *distance*. Remember that the distance is a non-negative value.
`public boolean hasSamePositionAs (Player player)`	Checks if the position of this player, i.e., the player on which this method is called, is the same as the position of the player given as an argument to the method. Two players are said to be in the same position if the distance between their positions is less than 0.001.
`public double distanceFrom (Player player)`	Computes the Euclidean distance between the position of *this* player, i.e., the player on which this method is called, to the position of the player given as an argument to the method.

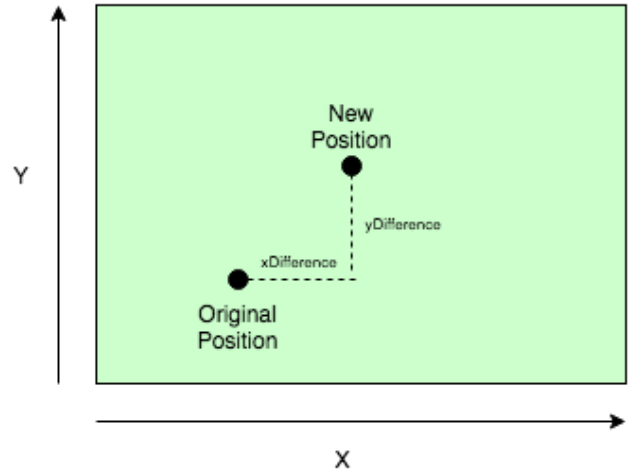

Figure 2: Changing the position of the player in the X and Y directions

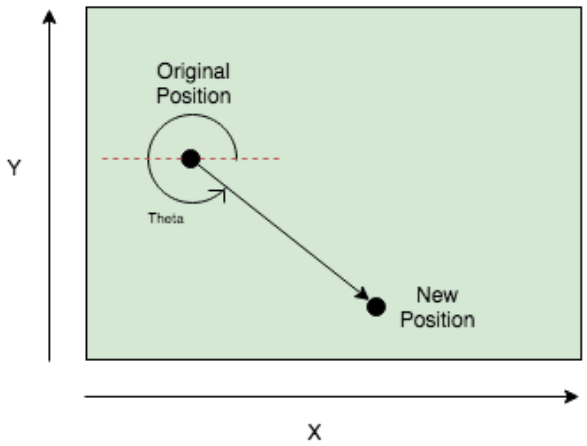

Figure 3: Changing the position of the player using an angle and offset

Demo

```
What is the name of player 1:
Alice
Enter the starting xPosition of Alice
-3
Enter the starting yPosition of Alice
4
What is the name of player 2:
Bob
Enter the starting xPosition of Bob
4
Enter the starting yPosition of Bob
-3
Enter Alice's horizontal move offset
4
Enter Alice's vertical move offset
-2
Enter Alice's diagonal move angle degrees
270
Enter Alice's diagonal move distance
1
Enter Bob's horizontal move offset
-3
Enter Bob's vertical move offset
-1
Enter Bob's diagonal move angle degrees
90
Enter Bob's diagonal move distance
5
Alice's position: (1.00000, 1.00000)
Bob's position: (1.00000, 1.00000)
Distance between players: 0.00000
Same position: true
```

Figure 4: Demo of a correctly-functioning program

Submission Instructions

Submit your files to Vocareum by following these instructions. Keep in mind that only your last submission will be considered.

Rubric

- Constructors: 10%
- Accessors/Mutators: 10%
- moveX/moveY: 5%
- moveInDirection: 15%
- hasSamePositionAs: 15%
- distanceFrom: 15%
- main: 30%

Good Luck