CSCI 2110
Data Structures and Algorithms
ASSIGNMENT NO. 1
Date Given: Monday, September 18, 2023
Date Due: Monday, October 2, 2023, 11.59 PM
Submission on Brightspace

Welcome to your first assignment. The objective of this assignment is to bring you back up to speed on the essentials of object-oriented programming with Java. You are to implement the program using an IDE such as Eclipse, NetBeans or IntelliJ.

Ball, Field and Player

In this exercise you will be writing a Ball class, a Field class and a Player class, and testing methods that emulate the kicking of a ball and bouncing it off the boundaries of the field. It is not a full simulation of a game, rather a simplified version to test your knowledge on the essential concepts in object-oriented programming. Develop the program class by class and test each class with a simple main method before you proceed further. You will be finally writing a demo program with a main method to make the classes work together.

First design a class called Ball with the following specifications. For simplicity, you are assuming that the Ball object is just a point in a two-dimensional space.

- bx, by (doubles) which represent the ball's x and y coordinates in a two-dimensional space.
- Getters and setters.
- toString method to display the ball's coordinates.

Next design a class called Field with the following specifications. The Field is just a Rectangle in a two-dimensional space. (xpos, ypos) represents the coordinates of the top left corner of the field, and length and width represent its dimensions.

(xpos, ypos)

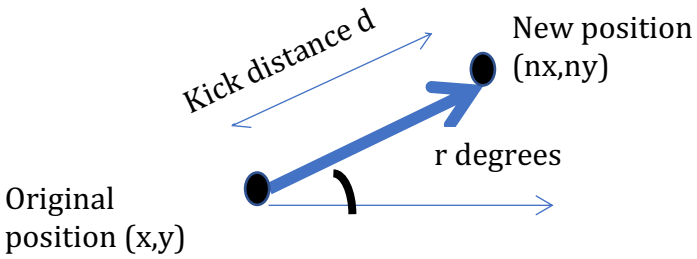
width

length

- Instance variables xpos, ypos, length and width that represents the coordinates of the top left corner of the field, and its length and width (all doubles).
- Constructor that accepts xpos, ypos, length and width as arguments and sets the instance variables.
- toString() method to display the values of the instance variables.

Next design a class called Player, which instantiates an entity for kicking the ball. For simplicity, we assume that the Player object is always near the Ball object. The Player will have a method called kick that specifies the **kick distance (double)** and **kick direction (which is an angle from 0 to 360 degrees – again a double variable).** The Ball object should move to the new position after the kick. If it hits the boundaries of the field, the ball should be reflected. Again, for simplicity, we will assume that the ball comes to a stop after it moves to its new position.

The new coordinates after the kick can be determined by simple trigonometry formulas and is explained in the example below.



New x coordinate nx is given by **x + d * cos(r_radians)**
New y coordinate ny is given by **y + d* sin(r_radians)**
Note: the angle must be in radians, so we should convert degrees r to radians before using the above formula.

Note that the y-axis in the Java coordinate system is inverted. But you can still use the above formulas by assuming that the kick angle is also inverted.

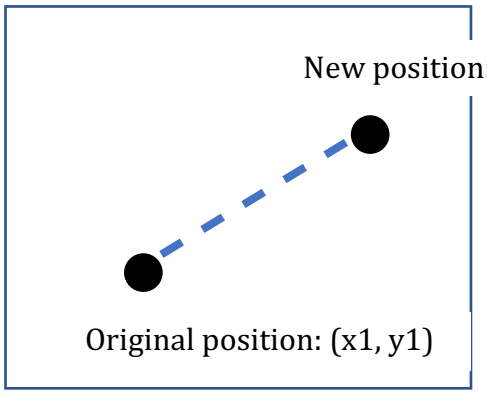
You can use the Java Math class methods to convert from degrees to radians and then use the cos and sin methods from the Math class, as in the example below:

```
double r = 45.0;
double r_radians = Math.toRadians(r);
double sinValue = Math.sin(r_radians);
double cosValue = Math.cos(r_radians);
```

You can use the above sinValue and cosValue to find the new x and y coordinates.
With the above background, design the Player class with the following specifications:

- Name: a String variable
- Constructor that creates a Player object with the given name.
- A toString method that displays the player's name.
- Method kick(Field f, Ball b, double d, double r) that kicks the instance of a given ball to a distance of d pixels and direction r degrees. If the ball hits the boundary of the field, it should reflect; otherwise, it should just move to a new position.
  (Again, remember that you are assuming that the player is always near the ball).

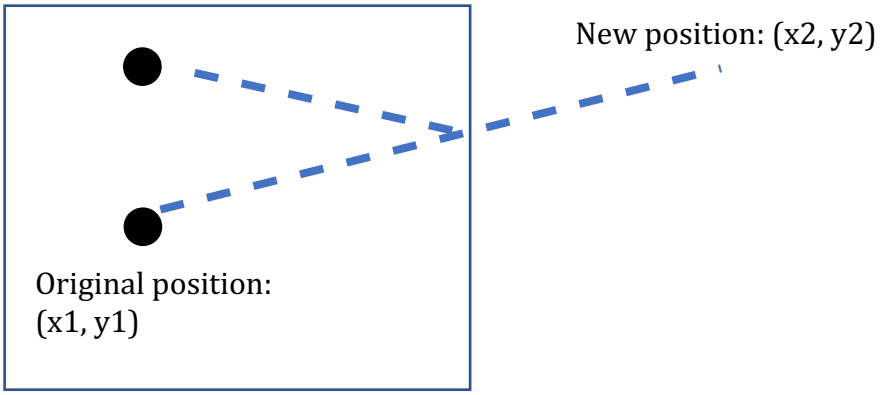
While the kick method looks intimidating, it is actually quite simple. Essentially, the kick method is a bunch of if statements. The kick method will determine the new position of the ball using the formulas given above. If the new position is within the field, then the ball just moves to the new position. If the new position is outside the field, then the ball gets reflected. We make simple assumptions about reflection, as illustrated in the examples (cases) below.



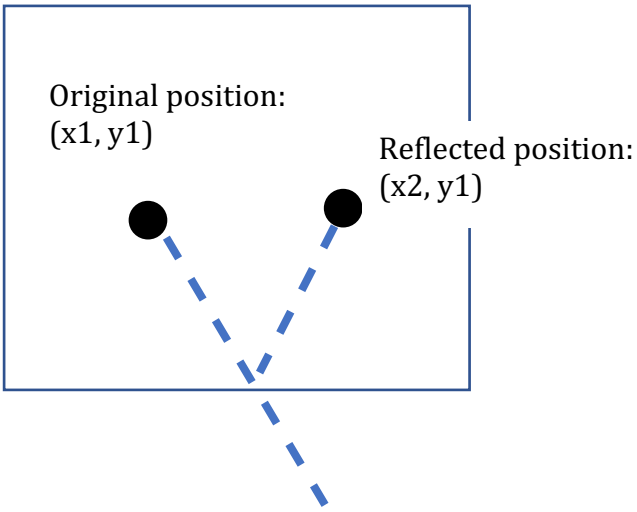
New position is within the field boundary, that is both new x and new y are within the field bounds.
In this case, change the x and y coordinates of the ball to the new values.

Even if the new position touches the boundary of the field, follow the above procedure (that is, change the x and y coordinates to the new values).

Reflected position:
(x1, y2)

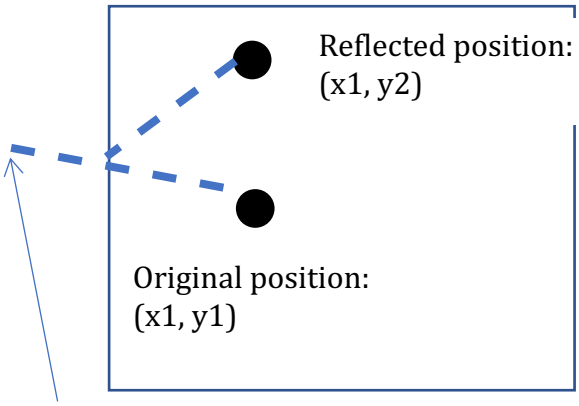


New x value takes it outside the field but the new y value is still within bounds. So the ball gets reflected.
The reflected position will have the original x value and the new y value.
Change the y coordinate of the ball to the new y value but do not change the x coordinate.



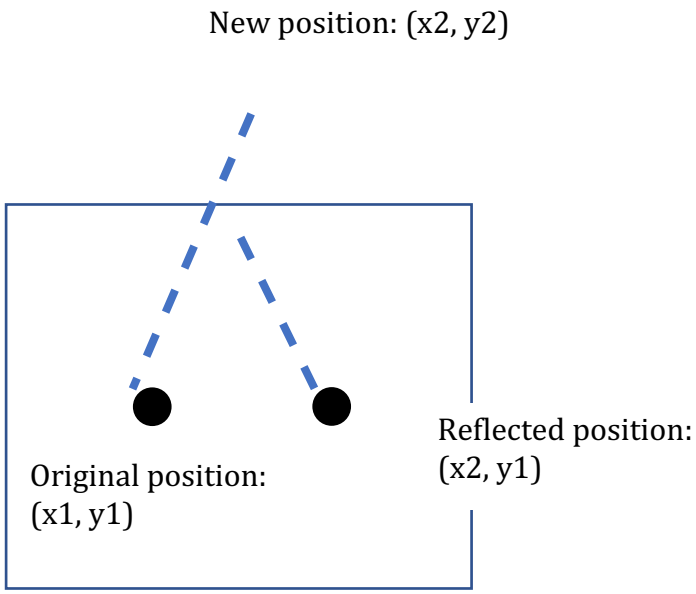
New position: (x2, y2)

New y value takes it outside the field but the new x value is still within bounds. So the ball gets reflected.
The reflected position will have the original y value and the new x value.
Change the x coordinate of the ball to the new x value but do not change the y coordinate.



New position: (x2, y2)

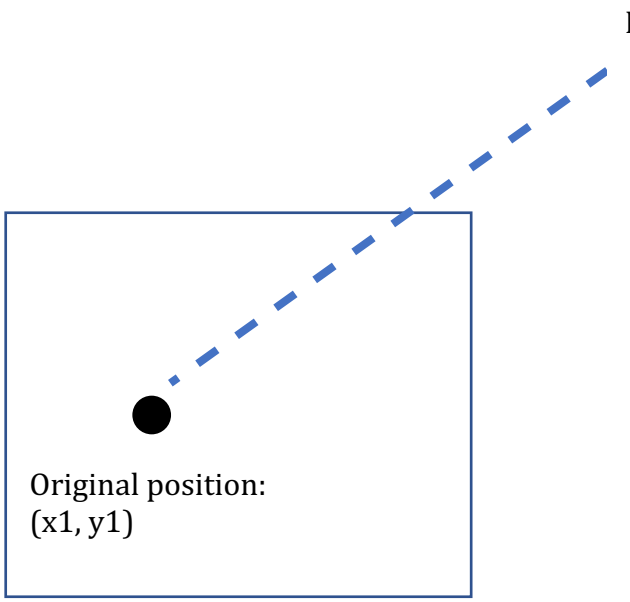
New x value takes it outside the field but the new y value is still within bounds. So the ball gets reflected.
The reflected position will have the original x value and the new y value.
Change the y coordinate of the ball to the new y value but do not change the x coordinate.



New y value takes it outside the field but the new x value is still within bounds. So the ball gets reflected.
The reflected position will have the original y value and the new x value.
Change the x coordinate of the ball to the new x value but do not change the y coordinate.

Special case assumption:

The above cases assume that one of the coordinates in the new position is within the field bounds. If both are outside the bounds, as in the example below, assume that the ball gets reflected back to its original position. That is, the x and y values remain the same. This is an over-simplification but serves for the purposes of this assignment.



Both new x and new y values are out of bounds. The ball gets reflected back to its original position, that is, do not change x1 and y1.
Don't change either the x or the y value of the ball.

Here's a simple Demo program that creates a Ball object, a Field object and a Player object and demonstrates a few kicks from the player.

```
public class Demo{
        public static void main(String[] args){
                Ball soccerBall;
                Field dalField;
                Player srini;

                System.out.println("SOCCER GAME SETUP!");
                srini = new Player("Srini");
                dalField = new Field(0.0, 0.0, 500.0, 400.0);
                soccerBall = new Ball(100.0, 200.0);
                System.out.println(srini + "\n" + dalField + "\n" +
soccerBall);
                System.out.println("\n");

                srini.kick(dalField, soccerBall, 300.0, 45.0);

                System.out.println(srini + " kicks the ball for a dis-
tance of 300.0 pixels at 45.0 degrees");
                System.out.println(soccerBall);
                System.out.println("\n");
                srini.kick(dalField, soccerBall, 500.0, 0.0);
                System.out.println(srini + " kicks the ball for a dis-
tance of 500.0 pixels at 0.0 degrees");
                System.out.println(soccerBall);
                System.out.println("\n");

                srini.kick(dalField, soccerBall, 100.0, 270.0);
                System.out.println(srini + " kicks the ball for a dis-
tance of 100.0 pixels at 270.0 degrees");
                System.out.println(soccerBall);

        }
}
```

Output:

```
SOCCER GAME SETUP!
Player Srini
Field: [0.0,0.0]500.0,400.0
Ball is at (100.0,200.0)


Player Srini kicks the ball for a distance of 300.0 pixels at 45.0 degrees
Ball is at (312.13203435596427,200.0)


Player Srini kicks the ball for a distance of 500.0 pixels at 0.0 degrees
Ball is at (312.13203435596427,200.0)


Player Srini kicks the ball for a distance of 100.0 pixels at 270.0 degrees
Ball is at (312.13203435596427,100.0)
```

In your assignment, write the demo program to do the following:
Create a Field object (use some reasonable values for the Field attributes)

Create a Ball object (set the x and y coordinates of the Ball object to be within the Field)
Create two Player objects
Generate random numbers to determine the kick distance and the direction for each kick. Assume reasonable bounds for the random number for the kick distance. The direction must be a number between 0.0 and 360.0.
Let each player take turns in kicking the ball 10 times using the random kick distance and direction.
Display the positions of the ball for each of the 20 kicks. Also display which of these kicks resulted in the ball being reflected.

What to submit: One zip file consisting of the following -

Field.java
Ball.java
Player.java
Demo.java
Text file showing a sample run from your program.

Note about comparing Doubles: As in your lab, in this assignment, you will be comparing doubles. If you use the == to compare two doubles, it may produce unexpected outcome. For example, (1.1+2.2)==3.3 returns false. It is common to define a small threshold value EPSILON (usually set to 10^-8 for comparing doubles. So you would write your equals method to return true if the result is + or – EPSILON.