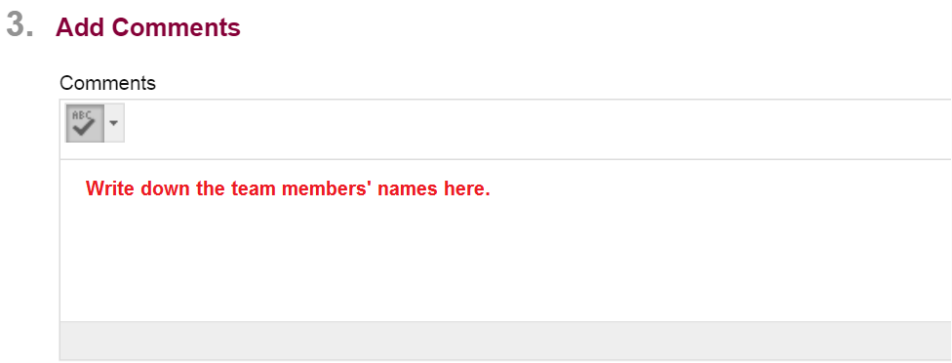
CSC 295 Selected Topics in Computer Science: Software Development (S17)

Project 2 (Deadline: May 18, 2017)

Project Instruction:

1. Please only create one Java project and then create one package for each problem. For example, we have two problems here, then create two packages. Give each package a meaningful name, such as problem1, problem2, etc., then put all java files of one problem inside the package of that problem. **Export the entire project and only submit one single zip file. Your need to create a word or txt or pdf file for the second problem, also put your document in the zip file.**

2. This can be done through team work. Each team can have up to two students. Both students will receive the same grade. One group only needs to upload one submission. Please write down the team members' names in "3. Add Comments" area as shown below:



Again, the purpose of letting you finish the project in groups is not to reduce the work load, it is to give you an opportunity to learn from each other and improve your collaboration skills.

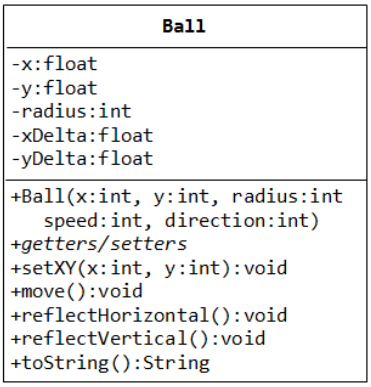
Project Problems:

1. Bouncing Balls (35 points)

(1). A class called Ball is designed as shown in the class diagram.

The Ball class contains the following private instance variables:

- x, y and radius, which represent the ball's center (x, y) co-ordinates and the radius, respectively.
- xDelta (Δx) and yDelta (Δy), which represent the displacement (movement) per step, in the x and y direction respectively.

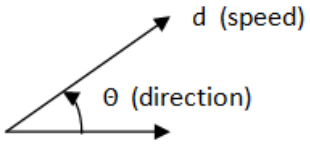


The Ball class contains the following public methods:

- A constructor which accepts x, y, radius, speed, and direction as arguments. For user friendliness, user specifies speed (in pixels per step) and direction (in degrees in the range of (-180 °, 180 °]).

$\Delta x = d \times \cos(\theta)$

$\Delta y = d \times \sin(\theta)$



User's Polar Co-ordinates

- Getter and setter for all the instance variables.
- A method move() which move the ball by one step.

  $x \mathrel{+}= \Delta x$

  $y \mathrel{+}= \Delta y$

- reflectHorizontal() which reflects the ball horizontally (i.e., hitting a vertical wall)

  $\Delta x = -\Delta x$

  Δy no changes

- reflectVertical() (the ball hits a horizontal wall).

  Δx no changes

  $\Delta y = -\Delta y$

- toString() which prints the message "Ball at (x, y) of velocity (Δx, Δy)".

Write the Ball class. Also write a test program to test all the methods defined in the class.

(2). A class called Container, which represents the enclosing box for the ball, is designed as shown in the class diagram. It contains:

Container
-x1:int -y1:int -x2:int -y2:int
+Container(x:int,y:int,width:int,height:int) *+getters/setters* +collidesWith(ball:Ball):boolean +toString():String

- Instance variables (x1, y1) and (x2, y2) which denote the bottom-left and top-right corners of the rectangular box.
- A constructor which accepts (x, y) of the bottom-left corner, width and height, which denote the width and height of the box, respectively.
- A toString() method that returns "Container at (x1,y1) to (x2, y2)".
- A boolean method called collidesWith(Ball), which check if the given Ball is outside the bounds of the container box. If so, it invokes the Ball's reflectHorizontal() and/or reflectVertical() to change the movement direction of the ball, and because of the change of the direction, ball will soon come inside of the container box.

Use the following statements to test your program:

```
Ball ball = new Ball(50, 50, 5, 10, 30);
Container box = new Container(0, 0, 100, 100);
for (int step = 0; step < 100; ++step) {
  ball.move();
  box.collidesWith(ball);
  System.out.println(ball); // manual check the position of the ball
}
```

2. Special Numbers (65 points)

Create a Java class named SpecialNumbers containing the following three int instance variables: nonPrimeNum, primeNum and perfectNum. In the constructor, initialize nonPrimeNum and primeNum to two random numbers within [1, 10000], and the probablities for primeNum to be a random number within the range [1, 50], [50, 450] and [451, 500] are 30%, 50%, and 20%, respectively.

Create an array containing 10000 objects of SpecialNumbers class. Use multithread programming and load balancing techiques to find out the object(s) satisfying the following constraints:

- nonPrimeNum is not a prime number
- primeNum is a prime number
- perfectNum is a perfect number
- Variance of above three numbers is the smallest among all objects.

If there is no such object existing, print out 0; if there are multiple objects satisfying above constraints, print out the values of the instance variables of each object.

Fill in the following table.

	100	200	300	400	500
2					
4					
6					
8					

The first column is the number of threads used in your program while the first row is the number of objects in each subtask. For example, the intersection of the second row and the second column mean that you should create 2 threads and each subtask contains 100 objects of SpecialNumbers. For each case, run your program three times and find out the average program execution time, which should be filled in the corresponding intersection place in the table. After filling the entire table, mark out the case (in red) with the minimum execution time.