Lab 04: Heisenberg

CO225: Software Construction

July 5, 2016

1 Goals

By the end of this lab, you should be able to:

- Write programs in Java using the concepts of Object-Oriented Programming
- Understand the difference between class variables and instance variables and make use of them in the right context

2 Introduction

In this laboratory class, you are expected to create a class called Ball to model the behaviour of a ball on a 2-D plane. Each ball has a position and a velocity (speed and a direction) at a given time and moves on a 2-D plane without friction that is to say the velocity does not change, but the position does. To simplify the mathematics we will assume balls are of negligible radius.

In your system you can have a number of balls and there is a global clock. These balls will be traveling at some velocity. New balls can be introduced to the system after sometime these new balls can have an arbitrary starting point and a velocity. For example at $t = 0$ the system can have two balls at positions (x_1, y_1) and (x_2, y_2) traveling at v_1 and v_2 respectively. Then when $t = 10$ we can introduced a 3rd ball with initial position (x_3, y_3) with a velocity v_3.

You should implement a Java Class called Ball to capture this behaviour. The class should provide the following functions (you should think of write keywords like *static*, *boolean* etc. for each function). You may use the provided skeleton code for your implementation. **But you will have to modify the return types access etc.**

2.1 Description

```
Ball(double x, double y, double speed, double angleOfSpeedWithX)
```

This is used to add a new ball to the simulation. Note that this is the constructor. The ball can be added after sometime of starting the simulation. So the system should keep track of the simulation time. Note that velocity is specified by specifying the speed and the angle it makes with the positive direction of x-axis of the 2-D plane.

```
updateTime(double time)
```

Run the simulation for the time specified. For now this would mean increase the system clock by that amount of time. You should use the correct return type and access modifier for this function ($static, public, ...$).

```
showAll(void)
```

This function should display all the balls at the given instance of time. The balls can be given an identity (some integer value) and its coordinates (x, y) should be displayed. Example: $(Ball1 : (19.2, 12.3)$. Note that you do not know the number of balls that would be there in the system at the start. (hint: ArrayLists from the lecture might be useful). Be careful about the type of this function.

```
willCollide(Ball b1)
```

This method can be called on a ball and will return whether or not they would collide with the given *ball* $b1$. You should not be able to call this method without an object (only $b2.willCollide(b1)$). Note here that two balls might be introduced into the system at two different times. so your function should consider that.

3 What to turn in?

Task

Submit your *Ball.java* file to feels before the deadline. We will test for correct use of Java keywords (public, static, return types etc.) and good programming practices. You will need some mathematics from your A/L as well. We will test your code with a main program similar what is given in the skeleton code.