CSE 271 – Spring 2017 Thursday Labs – Due Monday 11:59pm
Friday Labs – Due Tuesday 11:59pm

Lab 06- 30 points

For today's lab you will be doing one exercise pertaining to interfaces. It is just one exercise, and small so you can focus and finish your project, which is due this week!

Updated TA Office Hours

You are welcome to go to office hours for TAs from any section!

- Dustin Riley, rileydm3@miamioh.edu , Tuesday 730-830pm Benton 002
- Emily Pantuso, pantusen@miamioh.edu , Monday 700-800pm Benton 002
- Gage Laufenberg, laufengd@miamioh.edu , Monday 9-10pm Benton 002
- Michael Gentile, gentilm5@miamioh.edu , Tuesday 830-930 Benton 006
- Brandan Nofs, nofsbm@miamioh.edu , Sunday 6-7PM , Benton 006
- Rob Koch, kochrt@miamioh.edu , Sunday 7-8pm, Benton 006
- Thomas Burlett, burlette@miamioh.edu , Sunday 8-9pm, Benton 006
- Sarah Edmonds, edmondsh@miamioh.edu , Sunday 9-10pm, Benton 006

Lab Requirements

Suppose that we have a set of objects with some common behaviors: they could move up, down, left or right. The exact behaviors/implementations (such as how to move and how far to move) depends on the objects themselves. One common way to model these common behaviors is to define an *interface*. In this case, an you will define an interface called `Movable`, with `interface` methods `moveUp()`, `moveDown()`, `moveLeft()`, and `moveRight()`, `all of which return void`. (5 points)

Write two concrete classes - `MovablePoint` and `MovableCircle` - that implement the Movable interface. These classes that implement the `Movable` interface will provide actual implementation to the `(abstract) interface` methods.

MovablePoint Class (10 points)

For the `MovablePoint` class, declare the instance variables `x`, `y`, `xSpeed` and `ySpeed`; `which represent the x location, y location, speed in the x direction, and speed in the y direction`. Make getters and setters for the speed variables only, and one constructor that sets all the (4) instance variables. XSpeed and YSpeed can be positive numbers only, so make sure you program them accordingly (smart setters). Override toString() to an appropriate representation. Implement the movable interface with "logical" implementations that utilize these instance variables.

For the MovablePoint, you should be using your ySpeed and xSpeed values for the moveUp/Down and moveLeft/Right methods, respectively. And, you shouldn't be changing your speed in those methods

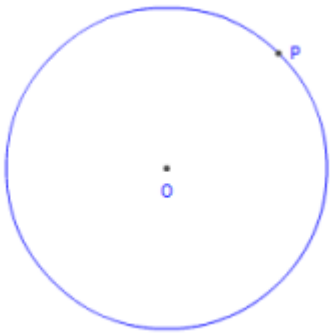
either (we are ignoring friction and momentum here). So, for example. If I have a MoveablePoint with the values

y = 4;
yspeed = 8;

Then, moveUp() would cause y to 12, and moveDown() would change y to -4.

MovableCircle Class (10 points)

For the `MovableCircle` class, use (have an instance variable of) `MovablePoint` to represent its center and an integer radius to represent the radius. Override toString() to an appropriate representation. Implement the Movable interface methods by moving the center MovablePoint using the move methods of that class. For this class, have a single constructor that configures a MovablePoint (4 values to pass to the MovablePoint constructer), and the radius as we demonstrate below in our test program.



Test Program and Demonstration (5 Points)

```
Write out a single test program, MovableTester, (class with a main method) that includes
these test statements and some of your own
Movable m1 = new MovablePoint(5, 6, 10,12);     // upcast
System.out.println(m1);
m1.moveLeft();
System.out.println(m1);
Movable m2 = new MovableCircle(2, 1, 2, 20,50); // upcast. Constructor takes in 4 point values and
//radius
System.out.println(m2);
m2.moveRight();
System.out.println(m2);
```

In addition, add one “public static” method called chaChaSlide(Movable[]) that takes in a collection of moveable things to your tester, in the same way we had an “average” method in our class example. This method returns nothing, but moves the objects in the following way listed below. In your main method, call that method with a single collection containing a variety of (at least one of each) points and circles,

1. Move to the left
2. Move to the right
3. Take it back now y'all (Move down)
4. One hop this time (Move up)
5. Move to the left
6. Take it back now y'all (Move down)
7. One hop this time (Move up)
8. To the right (Move right)
9. To the left (Move left)
10. Take it back now y'all (Move down)
11. One hop this time, one hop this time (Move up x 2)

Submission

Turn in a zip of your 4 files to Canvas by your respective deadline. Ensure you have actually submitted (received confirmation and checked online), since no late labs will be accepted. Grading will be based on conforming to the standards we reviewed in class as well as following the requirements of this lab.