CMSC141: Object-Oriented Programming

Lab 6: Objects

due: October 26th or 27th, 2017

Design, implement and share a new kind of Processing object.

1. Choose one specific everyday object, preferably one you can see, to simulate in Processing.
2. Write a ***blueprint*** of the object: describe the attributes of the object (i.e. nouns) and its methods (i.e. verbs, or actions). Use specific names for each method and attribute, and give a brief english language summary of each (see the table on page 2). The method descriptions should include any input parameters or output values. Each attribute should have appropriate getter/setter methods.
3. The instructor will give you someone else's design document.
4. Assuming your partner's class is complete---all methods are ready to be called—come up with a plan for using both your objects. **Interaction between the two objects should occur outside of any class** (e.g., in **`setup, draw, mousePressed`**). Start coding this part of the lab first.
5. When both of you have completed your implementations, share only your class definitions (e.g. **`Ball.pde`**, not **`setup & draw`**), and see how your programs function together!

Learning Objectives

- ❏ Write a Java class
- ❏ Design an interface
- ❏ Implement a class

Deliverables

- ❏ Your program should start with a comment that includes your name, email, date, assignment description, collaboration statement, and reflection.
- ❏ Bring a hardcopy of your program (the source code, not the graphics) to your next lab period.
- ❏ Also turn-in the original textual design document.
- ❏ Be prepared to run the Processing sketch and demonstrate your "Theory of the Program."

Example Documentation for our `Ball`

Attributes	
`pos: PVector`	Position of the ball
`r: float`	Radius of the ball
Constructors	
`Ball(pos: PVector, r:float)`	Create a new ball at specified position pos with radius `r`.
`Ball()`	Create a new ball at a random location with radius of 15.
Methods	
`move(speed: PVector)`	Change the position of the ball by `speed`.
`display()`	Display the ball as a red circle.
`bounce()`	Bounce the ball off of the left, right, top and bottom edges of the window.
`getR(): float`	Return the radius of the circle
`setR(r: float)`	Set the radius of the circle, the size can't be smaller than 1
`getPos(): PVector`	Return the position of the ball
`setPos(p: PVector)`	Sets the position of the ball to `p`
`getX(): float`	return the `x` position of the ball
`getY(): float`	return the `y` position of the ball
`setX(x: float)`	set the `x` position of the ball, keeps the ball within the window
`setY(y: float)`	set the `y` position of the ball, keeps the ball within the window