Computer Programming I   Instructor: Greg Shaw
COP 2210

Programming Assignment #2

*Using an Existing Class: Creating Objects
and Calling Accessor and Mutator Methods*

I. The Assignment

This assignment is to write a "test" class (aka: "client code") that creates and manipulates objects of the class *Balloon.java* (aka: the "domain class"). The Balloon class implements an astounding merger of technologies ancient and modern – a hot air balloon controlled by software.

To access the Balloon class, download it and store it in the **src** folder of your **NetBeans** project as *Balloon.java*.

The best way to learn how to use the Balloon class – or any other Java class - is to consult the documentation for the class. For *Balloon.java*, that would be *Balloon.html*.

These web pages tell us everything we need to know to
(1) create objects - the number and types of arguments required by the class *constructor*
(2) call methods - what methods are available, whether each returns a value and, if so, what type, and the number and types of arguments required by each

Feel free to examine the Balloon class code but don't worry if you don't understand it. You will in the near future and there is nothing in there that can be used in your test class. Remember: it is never necessary to know *how* a method does what it does. We just have to know how call it.

☞ Review declaring variables, creating objects, calling methods that return a value vs. "void" methods, and *accessor* and *mutator* methods before beginning.

To receive credit for this assignment, you must not modify the Balloon class in any way!

II. Your *BalloonTester* Class

Your *BalloonTester* class will have only a single method – main – and will perform each of the following operations, in the exact order listed below. Each operation may be done in one or two statements. Make sure you follow directions faithfully, and note that once you have done step 3, you can copy and paste it to do steps 6, 9, and 12.

1. Create a Balloon object with a name of your own choosing and an altitude of 100 meters.

2. Create a second Balloon object with a name of your own choosing, and specify an initial altitude of -50 meters.
3. Call the *accessor* methods of the Balloon class to get the name and altitude of each Balloon object. Print the data, one object per line.
4. Call the *ascendTo* method to move the object you created in step 1 to 200 meters.
5. Call the *adjustAltitude* method to increase the altitude of the object you created in step 2 by 300 meters.
6. Call the *accessor* methods of the Balloon class to get the name and altitude of each object. Print the data, one object per line.
7. Call the *adjustAltitude* method to decrease the altitude of the object you created in step 2 by 75 meters.
8. Make the object you created in step 1 ascend to the same altitude as the other object. *You may assume that the other object is at a higher altitude.*

   To get credit for step 8., the statement(s) you write must always work, regardless of the actual altitude of the second object. It cannot depend on you knowing the altitude of the second object. Hint: Since every Balloon knows its altitude, call a "get" method. In other words, if you use a literal to set the altitude, it is not correct.

9. Call the accessor methods to get the name and altitude of each object. Print the data, one object per line.
10. Move the object you created in step 1 to an altitude that is *twice* its current altitude. As in step 8, the statement(s) you write must work for *any* altitude and must not depend on you "figuring out" the new altitude beforehand.
11. Attempt to move the object you created in step 2 to an altitude that is 300 meters below its current altitude.
12. Call the accessor methods to get the name and altitude of each object. Print the data, one object per line.