I understand and will uphold the ideals of academic honesty as stated in the Honor Code.

CS 312
Assignment #2
This little Armie
Code Kata II
Due 9/22/17, in class

1. Goal
The goal of this assignment is to grok Java’s notions of inheritance, overloading, overriding, casting, and the @Override annotation. Note that Patrick’s version of “This little piggy went to market, …” is “This little armie …,” but, thankfully, the versus are all the same!

2. Problem Statement
[Rather than mimicking the extraction of requirements from a client, for this assignment, just follow these steps.]

1. Create a class LeftArm such that when a LeftArm object is printed the output is “I am a right arm”. Hint, override the toString method that gets inherited from class Object.

2. Create a class GrippingArm such that when a GrippingArm object is printed the output is “I am a gripping arm”.

3. Create a class StarFish that overloads the single-parameter method arm five times. Each of the five methods should print “A StarFish’s type arm” followed by its single parameter. The five values of type are int, double, String, LeftArm, and GrippingArm.

4. Create a subclass of StarFish named PisasterBrevispinus (according to Mr. Internet these are pink star fish). [A subclass such as PisasterBrevispinus inherits (get’s for free) all the non-private attributes and services from its superclass.] In the subclass override the arm methods for type int, String, GrippingArm to print something different. Then add the method leg(String s) that prints out “must be Patrick … only Patrick …” followed by the string s.

5. Create the class Tester with the method main. When coding the following, if the code for any step produces an error, comment out just the code causing the error and then add the error message from the compiler or the Java runtime system as a comment on the following line.

The remaining steps all involve adding code to main.

6. First create a StarFish object, a PisasterBrevispinus object, an int variable, a double variable, a String object, a LeftArm object, and a GrippingArm object.

7. Next have main print a heading and then send all five arm messages to the StarFish object. [Some OO terminology. An OO programmer views objectName.methodName() as sending the message methodName to the object objectName.]

8. Print a heading and then send all five arm messages and the leg message to the PisasterBrevispinus object.

9. Send the leg message (with the String of your choice) to the StarFish object.

10. Add the annotation @Override before each method in class PisasterBrevispinus. Predict which should work and which should not, then give it a try. [I’ll wait … were you right?]

11. Add the annotation @Override before the toString() methods of classes LeftArm and GrippingArm. If the midterm asks “provide a pro and a con of adding @Override to a toString method” what answer will you give?

12. A cast (attempts to) change the type of a value or an object. Java supports four kinds of casts:
    Widening Casts - byte → short → int → long → float → double,
    Narrowing Casts - byte ← short ← int ← long ← float ← double,
Upcasting - casting an object towards Object, and
Downcasting - casting an object away from Object.

(13) Create a Byte object initialized to 8 and a float variable initialized to 2.718. Then, send the arm message to the PisasterBrevispinus object using the Byte object as the parameter. Add a comment predicting the kind of casting used here. Run the code ... were you correct? How can you tell?

(14) Send the arm message to the PisasterBrevispinus object using the float variable as the parameter. Add a comment predicting the kind of casting used here. Run the code ... were you correct?

(15) Send the arm message to the PisasterBrevispinus using the float variable as the parameter, but prefix the actual parameter with (int). [Here the “(int)” is referred to as an explicit cast.] Add a comment predicting the kind of casting used here. Run the code ... were you correct?

(16) Declare a StarFish variable “StarFish iWasCastUp”. Then implicitly and explicitly assign iWasCastUp the value of the PisasterBrevispinus object you created in Step (6).

(17) Declare a PisasterBrevispinus variable “PisasterBrevispinus iWasCastDown”. Then implicitly and explicitly assign iWasCastUp the value of the StarFish object you create in Step (6).

(18) Add a method arm(short s) to class PisasterBrevispinus.

(19) Add the @Override annotation to your arm(short s) method.

(20) Finally, if the proceeding steps have not raised any questions in your mind, revisit them until a few pop up :) Then conduct two experiments of your own that address two questions that you have regarding overloading, overriding, casting, and the @Override annotation. Explain each in README.md including its motivation, the code changes and additions, and what you learned from the experiment.

What to hand in

(1) A well-formatted 2-up printout of your source code. Continue to exploit the useful printing incantations given with Assignment 1!

(2) A GitHub repo that includes (you must use these names as the grading script assumes them!)
  • README.md,
  • mycode.pdf (ensure that you are happy with the formatting by using xpdf), and
  • your Java source code with main in Tester.java.

Assignment Requirements  [ part of being a course rather than part of software development ]

• Create your code as part of the package asn2.
• Use clear documentation and careful formatting. Be consistent in indentation and alignment of braces. Each open brace “{” must be on its own line.
• Each source code file must start with
  // This is my code
  // <Your Name>
  // CS312
• Use a tab stop (indent) of either 2 or 4 spaces • no more!

Notes

• Reread the general notes regarding style, braces, and header comments.
• The @Override annotation informs the compiler that the element is meant to override an element declared in a superclass.

  @Override
  int overriddenMethod() { }