CMPS 12M
Introduction to Data Structures Lab
Lab Assignment 5

This is the third (and last) in a series of labs to teach you Java assuming you know C. All of the labs in this series are drawn from "Java for C Programmers" found at http://davin.50webs.com/research/1999/tsj4cp.html.

The goal of this assignment is to learn about basic inheritance and polymorphism.

Read
http://davin.50webs.com/research/1999/egs/q11.pdf,
http://davin.50webs.com/research/1999/egs/q12.pdf, and
http://davin.50webs.com/research/1999/egs/q13.pdf.
You do not need to answer the questions, although that would be a good idea, you should at least read them. If you can answer them easily it won't take much time, and if you can't then you will know what you need to study or where you should get some help from the TA during lab or in office hours. Answers to the questions in the reading are available at that same web site.

Inheritance and Polymorphism

In this lab you will combine concepts from each of the three exercises above, inheritance, runtime type information, and method overriding to implement a variation of the StarWars.java battle simulator presented in exercise 11 above. To get started, from the Labs/lab5 link on the class web page:

1. First download a copy of StarWars.java, XWing.java, and Tie.java, which are the same as those from the exercise above but split into three separate files, my preferred style (one class per file). The file StarWars.java will not be part of your final program and should not be part of your submission. It is provided only as a starting point.
2. Download the two files StarWars2.java (derived from StarWars.java with an incomplete battle() method) and StarWarsBattle.java (also derived from StarWars.java – this is the new main()).

In the version you will complete you will:

1. Create an abstract SpaceShip class per the instructions in exercise 11 that removes the redundant code from the XWing and Tie classes. **Place this class in its own file, not in the StarWars.java file.**
2. Complete the modified StarWars2.battle() method so that it produces the same output as the original StarWars.battle() method. Note that in your version, there is just a single array of SpaceShip[] that contains both types of spaceships, in no particular order.
3. Your modified battle() method must produce the exact same output as the original given the same collection of spaceships. Although the different types of spaceships can be arbitrarily interleaved in the array, all of one type (e.g. XWing) will be in the same order as they are in the example. Although you are not required to use them, as examples of using runtime type information, and as a possible aid in your solution, StarWars2.java contains two helper methods nextXWing() and nextTie().
4. All three of SpaceShip, XWing, and Tie, should contain toString() methods that are used to generate the output from StarWars2.duel() which must be used by battle() (as in the original) and **must not be modified**. Furthermore, the toString() methods in XWing and Tie should make no direct mention of shields or weapon.

Note that main() is now in a separate class, StarWarsBattle, which calls StarWars2.battle() passing it the array of fighters (SpaceShip objects) to use in the battle. You may of course modify StarWarsBattle for testing purposes, but your final submission should include the unmodified StarWarsBattle. Your program will be tested with different arrays of fighters.

What to turn in

You should turn in a lab5.zip file containing a single directory, lab5. The directory should contain five .java files: StarWarsBattle.java (unmodified), StarWars2.java (method duel() should be unmodified), SpaceShip.java, XWing.java, and Tie.java, and a pair programming log from logTemplates.txt named log.txt.