COMP 2150 – Spring 2014
Homework 4: Polymorphism
(Posted Feb. 26 – Due Mar. 5 by 12:40 pm)

Remember that as stated on the syllabus, this assignment should be an individual effort (contact me or visit the Computer Science Learning Center, www.cs.memphis.edu/cslc, if you need help).

Submission: Please zip your source code into a single file (you can zip the entire project folder if you're using BlueJ) and upload it to the proper folder in the eCourseware dropbox at https://elearn.memphis.edu. The dropbox will cut off all submissions after the indicated deadline, so don't wait until the very last minute to submit your work!

Grading: This assignment will be graded by the TA Rahul Vemuri (rvemuri@memphis.edu). If you have questions or concerns about your grade, please contact him first. I'll be happy to look over your assignment myself if he's not able to resolve the situation to your satisfaction. Also remember that as stated on the syllabus, all submissions MUST compile and run to receive credit. The TA does not have time to find and correct your syntax errors!

Coding Style: Be sure to follow the good coding practices that were discussed in COMP 1900:

- Use descriptive variable and method names. Avoid using single-character names unless it's very obvious what the variable's being used for (like a loop counter).
- Follow standard Java programming conventions for **variableAndMethodNames**, **ClassNames**, **CONSTANT_NAMES**.
- Consistently indent your code.
- Use comments judiciously. Every source code file should include a comment block at the top that lists your name and the assignment number. Every non-trivial method should have a comment preceding it that specifies what the method does, what its parameters are, and what it returns. For simple methods like accessors or mutators this isn't necessary, but it doesn't hurt to put in a simple comment like `// accessor methods`. I also expect to see comments throughout your code explaining what actions are being taken!

Suppose you own a computer store that sells two different types of products: desktop computers and laptops. These products can be described using the following attributes:

- Desktops have a processor, RAM, hard drive, and video RAM. The cost for a desktop[1] is calculated like this: base price of $150 + $6.50*(RAM in GB) + $0.15*(hard drive space in GB) + $0.48*(VRAM in MB)

- Laptops have a processor, RAM, hard drive, and screen size. The cost for a laptop is calculated like this: base price of $300 + $8.00*(RAM in GB) + $0.19*(hard drive space in GB) + $13.83*(screen size in inches)

1. **(15 pts)** Design and implement the following class hierarchy. All classes should have appropriate constructors.

    a. An abstract class **Computer** that contains as many common elements as possible for both desktops and laptops. It should also contain an abstract **getCost()** method.
    b. Two concrete subclasses of **Computer** named **Desktop** and **Laptop**

    Both **Desktop** and **Laptop** should have **toString** methods that return information on their type (desktop or laptop), individual components, and cost. **Desktop** and **Laptop** should also override the abstract **getCost()** method with the appropriate calculation.

[1] I can already hear the screams from PC hardware aficionados saying that this pricing scheme is full of fail. It's true, but do you really want to code a bunch of individual prices to account for processor architecture, cache size, clock speed, motherboard, PSU, memory speed/latency, HDD access time, solid state vs. magnetic vs. hybrid, DirectX 10/11 support, GPU architecture, custom cooling systems, etc.? I didn't think so ☺

2. **(10 pts)** Write a client program that creates an array of at least 5 **Computer** objects. The array should include both desktops and laptops. The program should then sort this array by cost (take advantage of the **Comparable** interface and the sort method for **Comparable** objects that I showed in class!), and display the final list of computers sorted by ascending cost. Note that there should be one array containing all of the computers, not two arrays with the desktops and laptops separately!

   Here's a sample of what your program's output might look like:

```
Laptop:
-------
CPU: Intel Core 2 Duo T7700
RAM: 2 GB
HDD: 96 GB
Screen: 15.1"
Cost: $543.073

Laptop:
-------
CPU: AMD A10-4600M
RAM: 8 GB
HDD: 800 GB
Screen: 13.0"
Cost: $695.79

Desktop:
--------
CPU: AMD FX-8150
RAM: 12 GB
HDD: 1024 GB
VRAM: 1296 MB
Cost: $1003.68

Laptop:
-------
CPU: Intel Core i5 4200M
RAM: 12 GB
HDD: 2048 GB
Screen: 17.0"
Cost: $1020.23

Desktop:
--------
CPU: Intel Core i7 2600K
RAM: 8 GB
HDD: 900 GB
VRAM: 3072 MB
Cost: $1811.56
```