CSCI-4448/5448: Homework 1
Java/OOP/OOD

Objectives

- Get familiar with Java.
- Understand, recognize, and able to use OOP.
- Understand, recognize, and able to use OOD.
- Able to create and use static methods. Understand when it is appropriate to use static variables and static methods.
- Able to create and use interfaces at the best level in the class hierarchy. Understand the differences of implementing at different levels in the hierarchy.
- Differentiate between getClass() vs instanceof. When would you use each?
- Recognize the best option to create a superclass, abstract superclass, or interface.
- Recognize what will happen in the future when new requirements come in based on various solution options.
- Introduce you to what this class will be covering through the semester.

---

1. Create a class named `UserUtility` that has a static method named `createUsername` that takes two parameters: their first name and their last name (in that order) and returns a username as the first 3 characters of their first name plus the last 3 characters of their last name.
   Then create a driver program that prints the result of calling your method statically with "Liz" and "Boese" as the parameters.

2. Using the code on the website, change only the `Media`, `Book` and `DVD` classes to implement the `Comparable` interface and use the driver program (without changing it) to sort them correctly.
   Expected output:

```
1999: Dare to go Solo [DVD]
2000: Dare to go Solo [DVD]
2006: Dare to go Solo [DVD]
2012: Dare to go Solo [DVD]
1970: Intro to Programming [DVD]
2011: Intro to Programming [DVD]
Dare to go Solo by Elizabeth Boese
Dare to go Solo by Wise Travelguy
Intro to Programming with Java Applets by Elizabeth Boe
Intro to Programming with Java Applets by Elizabeth Boese
```

Questions

1. Where should you designate the implements Comparable? Can you do it at the superclass level or should you do it at the subclasses level?
2. Do you need to retype the methods from an interface in an abstract superclass that implements that interface? Why or why not?
3. How in Java do you check to see if an object is an instance of another class type?
4. When you are dealing with an object variable of a superclass data type, can you call the methods on it from a sub-class if you know which subclass type it is? Why/why not? If not, how would you call those methods?

5. In a subclass, can you reference a variable in the superclass with the this keyword? How else can you reference it? What is the best practice? What about inside constructors?
6. How do you compare two objects' titles in this one method? e.g., where do you get the title from object 1 and the title from object 2? Do you understand how this is working when you send an object of the same class type into a method like we are doing?
7. If inside Media, we added code like that below to print out the class of each object, would it say Media for all of them or would it be Book/DVD?

```
...
public int compareTo(Media media)
{
      System.out.println(this.getClass());
      System.out.println(media.getClass());
```

8. What happens to your solution if we extend the program to include USB, MP3, iPhone and Android phones?
9. Provide definitions for the following terms.
    a. abstraction
    b. encapsulation
    c. cohesion
    d. coupling
    e. How does each of these terms apply to the object-oriented notion of a class? Provide examples of both good and bad uses of these terms in the design of a class or a set of classes.
    f. In your program give an example of abstraction (beyond "there's an abstract class").
10. How is your soution following the open-closed principle?
11. How is your solution following the Liskov Substitution principle?
12. What woud be an example implementation violating the Dependency Inversion principle?