**Lab 7**

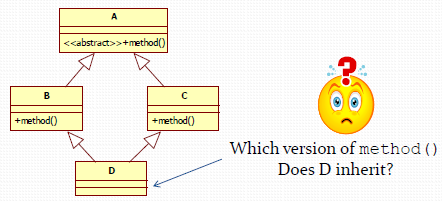
1. Short Answer
2. In an earlier lesson, it was mentioned that Java’s ArrayList implements 6 interfaces and extends one class. What are they?

Parts B – D of this Problem refer to code in package lesson7.labs.prob1, in which you are trying to remove duplicates from a List and then test that your output is correct. All three attempts to solve this problem are incorrect in some way (when you run the code, output message indicates that the procedure fails). Explain, in each case, what is wrong with the solution. Place each of your answers in a text file in the relevant package.

This question is trying to help you understand that it’s important to implement equals() and hashcode() methods in correct way. If you’re not familiar with those two methods, please run the questions in debug mode in Eclipse, it’ll help you find out what’s wrong with the code.

E. Lesson 5 introduced the Diamond Problem that must be handled by any language that supports multiple inheritance. Java SE 8 now supports “behavioral” multiple inheritance (but not “data” multiple inheritance). Explain how features of Java 8 handle the Diamond Problem by considering two scenarios:

i. When the type D is a class  
 ii. When the type D is an interface.



Since Java doesn’t support multiple inheritance,

a. If D is a class, that means B or C, one of them is class, the other must be interface. Both B and C has method(), how Java solve the conflict?

b. If D is a class, B and C can both be interface, how Java solve the conflict?

c. If D is an interface, B and C can must be interface. Because interface cannot extends/implements class. Both B and C has method(). How Java solve the conflict if you think it has conflict?

1. The code for Lab7 Prob3 includes the following classes/interfaces: Cache, StaticStorage, along with a driver class Main and a Customer class. StaticStorage is intended to store data that becomes available during the execution of the application, and this data needs to be accessible throughout the application for a certain period of time. It is reasonable to make StaticStorage a singleton. Since StaticStorage is going to play the role of a cache, it is also natural for StaticStorage to inherit from Cache. For simplicity, we have only one method in Cache: timeout(). This tells how long items will be allowed to stay in the cache. For this problem, refactor Cache and StaticStorage so that

a. StaticStorage is a singleton (by making it an enum)

b. The method timeout() can be accessed by StaticStorage through “inheritance” (and the static keyword is removed)

This one is to practice Enum is the threadsafe and best way to create Singleton.

1. In the lesson7.lab4.prob4 package, there is a class called ForEachExample that specifies, in its main method, a list of Strings. Use the Java 8 forEach method within the main method to print out the list so that all Strings are in upper case. To do this, you will need to define your own implementation of the Consumer interface.

Java 8 provides a default method forEach(Consumer c) which asked for Consumer Object. Consumer is an interface. If you don’t know Lambda expression, think about pre-java 8, how to get an instance which type is an interface. There are many ways to do that, we also reviewed during the class. During Java FX lesson, when you implement setOnAction() method, which technique did you use?

1. In the lesson7.lab4.prob4 package, there is a class called ForEachExample that specifies, in its main method, a list of Strings. Use the Java 8 forEach method within the main method to print out the list so that all Strings are in upper case. To do this, you will need to define your own implementation of the Consumer interface.
2. Rework the Duck Application of Lab 5, Problem 1 so that Flyable and Quackable interfaces are used after all, but now use Java 8 interfaces. Rewrite your code with this approach.

In class, we made progress toward a class diagram for the DuckApp, reproduced below. How will the Duck class use the FlyBehavior and QuackBehavior interfaces? Implement the diagram in Java, and make sure the answer to this question is clear in your code. To implement the methods like fly() and quack(), just print a phrase to the console, like “Flying with wings” or “Quack by squeaking.”

In this class diagram, the FlyBehavior and QuackBehavior are interfaces, the actual fly behavior is implemented in the subclasess. The same for QuackBehavior.

This question is to practice default methods in Java 8. Since interface can have default method, you can leave all the default behavior(fly() methods in FlyWithWings, CannotFly ) in the interface which means you can create two interface Flyable, UnFlyable which contains default fly, cannot fly default behavior. Then The Duck class can implements the some interface which have default behavior. For each subclass of Duck, it can implements the interface which contains the correct behavior for them.

