University of Victoria
Faculty of Engineering: Computer Science

CSC 115: Fundamentals of Programming II

Spring 2016

Assignment 1

Objectives: Upon completion of this assignment you need to be able to:

- Write Java code following the Coding Convention defined for the course.
- Write (Implement) Java classes, instantiate and test objects
- Use arrays that contain either basic or complex data types.
- Read and understand specifications
- Use a developed testing system.

Resources:

- CSC 115 Java Coding Conventions.pdf (found on conneX, under Resources)
- Specs.html
- Skier.java
- SkierList.java
- Lesson.java

Introduction:

The local ski hill has just been bought and the new owners really need a new system to track people learning to ski in their lesson programs. The system will model individual learners (called skiers), recording name and current ski level (0-4), lists of skiers, and lessons that use the lists to form groups of skiers who are in the same level. The result will be an initial prototype that will be used in negotiation with the owners to (hopefully) earn a contract to develop a more fulsome system.

In this assignment you will:

- Implement the `Skier` class that will be used by a customer service representative when creating or managing a learner's record. It will record both a name and a ski level (0-4) associated with the learner. i.e. Emma Jackson is a currently a level 3 skier.
- Create and implement the `SkierList` class, which will store a possibly-changing list of skiers.
- Create and implement the `Lesson` class, containing a `SkierList` object that is subsequently made up of `Skier` objects associated with the lesson.

The Java source code files have been started for you, but they need to be completed. Internal to each java file, you will see some requirement comments preceded by "Programmer note: …". Once you have completed the code, you can remove the programmer note comments.

Quick Start

1) If you haven't done so already, download all the necessary documents for this assignment from conneX into a specific folder for CSC115 assignment one, `assn1` is a recommended name.
2) Double-click on the `Specs.html` file to open a webpage that links to the three document files for `Skier`, `SkierList` and `Lesson`. Look at these documents frequently, noting the method details and relationship of the classes to each other. It is recommended that these pages be open on your browser whenever you are implementing the java files.
3) Each one of these files can be compiled without errors. In the freshly created directory that contains all the files type

   ```
   javac *.java
   ```

   and all the java files will compile. If you want to compile just one of the files, then substitute the name for the wildcard '*' character. A successful compilation produces no output to the console, but new byte code files are created, namely the `*.class` files.
4) Each of the files is also able to run after the initial compilation. For example, type

   ```
   java Skier
   ```

   and the following message will appear on the console:

   ```
   Testing the Skier class.
   Failed at test one.
   ```

   This is hardly surprising, since you haven't written any code yet! As you progress in the writing of each of the methods in each of the files, the single error messages will progress until you see the following message:

   ```
   Testing the Skier class.
   All tests passed.
   ```

   The `main` method in each of the files is calling the methods and producing the output. Don't change this method; look at the code so you know what is being tested. It is recommended that you implement each method in the order that `main` tests them.

Detailed Instructions

You will be implementing each of the java classes, one in each of the separate java source files. It is important you complete them in the following order: `Skier.java`, `SkierList.java`, or `Lesson.java`. Complete the requirements and fully test each class before moving on to the next one. You will find that each subsequent class becomes easier to implement, so pay most attention to getting the first one completed properly.

Hint: Each progressive class will have references to the previous class you completed. There will be `Skier` objects referred to in the `SkierList` class and in the Lesson class. There is a `SkierList` attribute in the Lesson class. For example, the size of the `SkierList` is the same as the number of students in the class. So call the size method of the `SkierList` class when determining the number of students in the `Lesson` class.

For each class file, correctly **comment** and **implement** the each of the methods. Follow these guidelines:

- The associated html file contains a detailed explanation of each method. Read each method specification carefully and implement the method. Note the specification provides excellent method header commenting. Use them.
- Do not change the instance variables or the main method.
- The main method is a fully functional tester that tests each method sequentially. We recommend you implement the methods in the order they are tested in the main method.
- Compile and test your work frequently. Experienced programmers all do this, knowing that a single syntax error or logical error is easier to fix when it you've only made a few changes to the document.

Submission

Submit the following completed files to the Assignment folder on conneX. You will be shown how to do this in the first week of labs:

- `Skier.java`
- `SkierList.java`
- `Lesson.java`

Please make sure you have submitted all the files, and conneX has sent you a confirmation email. Do not send `[.class]` (the byte code) files. Also make sure you *submit* your assignment, not just save a draft. All draft copies are not available to the instructors, so we cannot mark them.

Note about Academic Integrity: It is OK to talk about your assignment with your classmates, and you are encouraged to design solutions together, but each student must implement (code) their own solution.
We will be using plagiarism detection software on your assignment submissions.

Grading

You will receive no marks for java files that do not compile.
The marker will be looking for:

- Proper programming style as per the code conventions on CSC115 conneX Resources.
- All methods are implemented exactly to the specifications, accessed via Specs.html.
- The proper expansion of the array in `SkierList` does not include any calls to methods in the `java.util` package. In other words, the contents of the smaller array are transferred to a larger array using a for-loop.