8 Assignment

Etudes

8.1 Part 1

Finish all the lab for this week and include it in your portfolio.

8.2 Part 2

Do the first 4 problems below. For the *filter* method, add examples of its use in the context of the provided code.

Main Assignment: Abstracting with Object, Interfaces, and Function Objects

8.3 Problem

You will start this assignment with the given code. The code as written runs in ProfessorJ — separate it into files as needed to build a Java project. You should also use the *TestHarness* for testing of your program.

The code defines a list of objects, a class that represents a person, and its derived class that represents a student. There are no examples of data. Some methods are already defined, but do not include tests. By filling in the missing pieces of the design recipe, you will become familiar with the code. You will then add new methods to this class hierarchy.

1. Draw a class diagram for the given class hierarchy.
2. Make examples of data as required by the design recipe.
3. Design and run tests for all methods in the class hierarchy that represents a list of objects, using a list of students as the sample data.
4. Design the method *filter* that produces a list of all objects that satisfy the given predicate.
5. Design an *interface ICompare* that contains a method *betterThan* which takes as argument one *Object* and returns a *boolean* value.

6. In the *class Person* the basis for the *betterThan* comparison is the alphabetical ordering of the *name*s. In the *class Student* the ordering is determined by the *gpa*. Modify each class to implement the *ICompare* interface accordingly.

7. Add to the classes that represent a list of *Object*s the method(s) that implement insertion sort, the ordering determined by the *betterThan* method of the *ICompare* interface. Test is with both, lists of *Person*s and lists of *Student*s.

8. Design the method that verifies that a list is sorted according to the ordering determined by the *betterThan* method of the *ICompare* interface.

9. Design and implement quicksort for the list of *Object*s. (Refer to HtDP for the explanation of quicksort.) ∎

8.4 Problem

1. Define an interface *IObj2Obj* which contains a method *obj2obj* that consumes one *Object* and produces another *Object*.

2. Define a class that implements this interface by consuming an instance of the *class Student* and producing a *String* that contains student's id, name, credits, and gpa. For example, it may produce "1234, Jenny Smart, 34 credits, gpa 3.4".

3. Design the method that for the list of *Object*s that consumes an instance of a class that implements *IObj2Obj* and produces a new list of *Object*s where every element is the result of invoking the *obj2obj* method with the element of the original list as its argument.

4. Write a test case that produces a list of *String*s representing all honors students (with *gpa* greater than 3.5).

5. Write a test case that produces a list of *String*s that represent all students with more than 80 credits. ∎