Lab 6: Project Interfaces

Due: 10/3/12 – 11:59PM

Overview

For our semester project, we are going to build a library (that is, a package) of reusable code for generating and manipulating data. To do this, we are going to define two interfaces – DataProvider and DataProcessor – and implement some core functionality in terms of these abstractions. Classes that implement these interfaces will communicate with each other using the LookupTable you developed in Lab 3.

This code snippet demonstrates the use of these interfaces:

```java
public static void run(DataProvider in, DataProcessor out)
{
    while (in.hasData())
    {
        LookupTable lt = in.nextData();
        out.process(lt);
    }
}
```

As the semester goes on, we will create interesting implementations of these interfaces. Today, we just want to get all the machinery in place.

Tasks

1. Write the source code for the interfaces listed above, so that the provided run() method will compile.

2. Provide an implementation class of DataProvider that loads a LookupTable with arbitrary data, like the main() method from Lab 3. Note: you will need to involve a new instance variable so that hasData() returns true at least once, but eventually returns false so we don’t have an infinite loop.

3. Provide an implementation of DataProcessor whose process() method simply calls printAll() on the incoming LookupTable.

4. Write a Driver class that instantiates your classes from steps 2 and 3, and calls the run() method provided above. (Make run() a static method in your Driver class for now).
5. Write a second class that implements DataProvider. It can be exactly like the first such class, just put different arbitrary data in it. Modify your Driver code to use this new class as the provider argument to run(), instead of the one developed in step 2.

Summary

You have written a program where key pieces of functionality can be replaced without changing the rest of the code. Hopefully you can appreciate how interfaces make this possible. Now try to think of real world applications that might fit this provider $\rightarrow$ processor model.

Turn in all source code in a directory called YourName_1110_Lab6 by the due date.