Central Washington University
College of the Sciences
Department of Computer Science

CS-301 Data Structures   Fall 2016

Lab Practice 01

In this lab, we are going to practice the use of inheritance, exception catching and the concept of abstract classes.

Normally you, will find the source and data files in `/home/cs-301/Labs/Lab01`The files needed are

```
lb01.pdf
FinalExamDemo.java
FinalExam.java
GradedActivity.java
AbstractPolygon.java
```

The first one is the *.pdf file that you are reading.

1. In the list of programs above are three programs related to grading. Compile and run the programs, using the main in `FinalExamDemo()`

2. Create another class `ClientGraded` that in addition to invoke the method in `FinalExam` also invokes `Project`, also derived from `GradedActivity`. `Project` consist of 3 questions, each with 3 parts
   – Correcteness (70%)
   – Style (15 %)
   – Documentation(15%)
   Include proper constructors and methods to read all the items and provide the letter grade using the base class `GradedActivity`.

3. After investigating the methods and classes using the Java API site, of the abstract class `AbstractPolygon.java`, extend the abstract class into `Hex`, `Quad`, and `Triangle` classes, representing polygons of 6, 4, and 3 sides. Bear in mind that in addition to the points, the individual classes must have a variable called `color` (alas, it is not part of `AbstractPolygon`.). Write constructors for the form:

   ```
   ConcretePolygon( int [] x, int [] y, Color color)
   ```

   You may leave `paint()` just a stub.

4. Your are going to write a classes `ExHandn` that will contain include `main` as well as

   ```
   private static void myread()
   ```

We are going to experiment with propagating and catching exceptions ocurring inside myread(). In your, class create a string static field, where things are going to be read into. To instantiate a BufferedReader field inb use the layering:

```
private static InputStreamReader streamIn =
    new InputStreamReader(System.in);
private static BufferedReader inb =
  new BufferedReader(streamIn);
```

Now find out about the method readLine(), which is going to be used by myread() to read into the buffer. The main must simply prompt the user and call myread(). Ignore completely the possibility of checked exceptions, but *attempt* to compile.

5. Write a version which propagates (throws mechanism) the exception up to main. This first working version is class ExHand1

6. Write two more versions that catch the exception: The first one catches the exception in myread() (ExHand2), the other one in main() (ExHand3).

7. Suppose that in the program above, you want to read integers. To convert a string buffer to an integer, Integer.parseInt(buff) is commonly use. What exceptions does it throw?. Do you *have* to handle them? Include a try-catch block in you program that exits gracefully if input is not an integer.