CS 211, ALL SECTIONS

LAB EXERCISE-3

DUE MONDAY, FEBRUARY 11TH AT MIDNIGHT

The objective of this lab is to be able to build simple classes, including designing the needed data structure and implementing the requested behaviors; also to be able to effectively use Java methods for processing input (via the `Scanner` class).

OVERVIEW:

1. Write two Java classes which implement the requested functionality. The first one will implement a student grade, while the second one will implement an input filter which splits intput into floating points, integers, and ordinary strings.
2. Download and use a tester module to ensure that your programs are correct. The programs can also be tested using Dr Java's interactive window.
3. Prepare the assignment for submission and submit it

This is a lab exercise on which collaboration (including discussing the solution on Piazza or searching online sources) is explicitly permitted. It is still in your interest to understand and prepare your final submission yourself.

GRADE TASK:

(*4pts*) First, write a class named `Grade`. This class should keep track of a student's grade. The grade itself will be stored as a floating point number, but it should also be able to produce a corresponding letter grade. You may not use any import statements. Any data elements should be declared `private`, although this will not be graded. If you think it is necessary you may include helper methods of your own. The class should implement the following `public` methods:

- `public Grade()` the constructor will initialize the grade to `100.0`.
- `public Grade(double grade)` the constructor will initialize the grade to the given value. If a grade less than zero is passed in, then the value should default to zero instead.
- `double getGrade()` this getter method will return the student's grade.
- `void setGrade(double grade)` this setter method will set a new value of the grade. If a grade less than zero is passed in, then the value should be ignored and the original kept instead.
- `public String toString()` returns the letter grade corresponding to the numerical grade. The scale is: `"A"` for `90` and above; `"B"` for grades `80` and above but below A; `"C"` for grades `70` and above but below B; `"D"` for `60` and above but below C; `"F"` for all other grades.

All method signatures should match the ones shown exactly.

INPUT SPLITTER TASK:

(*5pts*) Now, write a class named `InputSplitter`. This class will take input from the keyboard (using a `Scanner`) and split the incoming tokens into integers, floating point numbers, and strings. The incoming tokens will be added to one of three accumulators which start out at zero, and which keep a running total.

Thus, the class will have an accumulator for integers. It starts out with the value zero. Every time another integer is read in, it is added to the integer accumulator. The accumulator keeps a running total of all the integers which have been seen so far. Likewise, there is an accumulator for floating point values. It starts at zero, but adds every floating point value which is seen. Finally, there is a string accumulator. It starts out as an empty string, but appends to the string every time another string token is read. Your class will need to be able to decide the type of each input token (hint: a `Scanner` has the methods `hasNextInt()`, `hasNextDouble()`, and `hasNext()`) as well as to keep track of the running totals (hint: you can use instance variables).

The only import statement you may use in this implementation is the `Scanner` class. Any data elements should be declared `private`, although this will not be graded. If you think it is necessary you may include helper methods of your own. The class should implement the following `public` methods:

- `public InputSplitter()` this constructor will initialize the three accumulators as well as the `Scanner` which is used for input.
- `public void next()` read the next token, and print its type and value. This method will use the input `Scanner` object which was initialized earlier to read the next token from the input. Depending on the type of the input token, method will print the value of the token in one of the following three formats:

```
integer: 1
```

```
double: 2.0
```

```
string: abc
```

In addition to the printed output, the value will be added (appended, in the case of a string) to the accumulator for the specific type.

- `public int getIntTotal()` retrieves the total value summed in the integer accumulator.
- `public double getDoubleTotal()` retrieves the total value summed in the floating point accumulator.
- `public String getStringTotal()` retrieves the complete value concatenated in the string accumulater.
- `public String toString()` returns the current contents of each of the three accumulators as a single string in the following format:

```
integer: 1 double: 2.0 string: abc
```

Below is a sample run demonstrating how the class would function using the following input:

```
1 abc 2.0 3 def 4.0 6 7 xyz 5.5
```

```
> InputSplitter i = new InputSplitter();
> System.out.println(i);
integer: 0 double 0.0 string:
> i.next();
integer: 1
> System.out.println(i);
integer: 1 double 0.0 string:
> i.next();
string: abc
> i.next();
double: 2.0
> System.out.println(i);
integer: 1 double 2.0 string: abc
> i.next();
integer: 3
> System.out.println(i);
integer: 4 double 2.0 string: abc
> i.next();
string: def
> System.out.println(i);
integer: 4 double 2.0 string: abcdef
> i.next();
double: 4.0
> System.out.println(i);
integer: 4 double 6.0 string: abcdef
> i.next();
integer: 6
> i.next();
integer: 7
> i.next();
string: xyz
> i.next();
double: 5.5
> System.out.println(i);
integer: 17 double 11.5 string: abcdefxyz
> System.out.println(i.getIntTotal());
17
> System.out.println(i.getDoubleTotal());
11.5
> System.out.println(i.getStringTotal());
abcdefxyz
```

TESTING:

Download the following (if you don't already have it):

- https://mason.gmu.edu/~iavramo2/classes/cs211/junit-cs211.jar
- https://mason.gmu.edu/~iavramo2/classes/cs211/s19/E3tester.java

This is a unit tester which is what we will use to test to see if your classes are working correctly.

When you think your classes are ready (you can try them even without the tester), do the following from the command prompt to run the tests.

On Windows:

```
javac -cp .;junit-cs211.jar *.java
java -cp .;junit-cs211.jar E3tester
```

On Mac or Linux:

```
javac -cp .:junit-cs211.jar *.java
java -cp .:junit-cs211.jar E3tester
```

In Dr Java:

- open the files, including `E3tester.java`, and click the *Test* button.

SUBMISSION:

Submission instructions are as follows.

1. Let *xxx* be your lab section number and let *yyyyyyy* be your GMU userid. Create the directory *`xxx_yyyyyyyy_E3/`*
2. Place all of the `.java` files that you've created into the directory.
3. Create the file `ID.txt` in the format shown below, containing your name, userid, G#, lecture section and lab section, and add it to the directory. Your directory should contain three files total.

   Full Name: Donald Knuth
   userID: dknuth
   G#: 00123456
   Lecture section: 004
   Lab section: 213

4. compress the folder and its contents into a .zip file, and upload the file to Blackboard.