Homework 2 (HW2)

EE 363 (Fall 2018)

Department of Electrical & Computer Engineering

Clarkson University

Due: October 11, 2018 at 11 pm.

Instructions

Please read the instructions carefully before submitting your work.

Note: Solve all problems and upload your answers to Moodle. Whenever you write solutions on paper, you need to scan it and upload the file(s).

Note: *user* stands for your login ID on *polaris.clarkson.edu*. This should be the exact same as your CU ID.

Note: Make sure any code you write works on *polaris.clarkson.edu* before uploading your files. You may lose many points if your code doesn't even compile.

Note: Please do not upload any executable or intermediate files as answers to problems, unless specifically asked to do so.

Note: Read Chapters 1 through 4 of the textbook and answer the questions that follow.

Total points: **40**

Problem 1 (10 points)

Study Student.java and studentAPI.txt (these were discussed in class).

Modify function *updateCourseGrades* in class *Student* so that if any of the elements inside the argument are negative, it throws an *InvalidGradeException*. You should define *InvalidGradeException* as a subtype of Java's *RuntimeException* class. Note that *updateCourseGrades* accepts an array of grades for a student, and adds them to the existing list of grades for this student; it does not remove or overwrite any previous point grade.

Then, write a client class *UseStudent* that attempts to do the following:
--create two *Student* objects named ms1 and ms2; ms2 is to be initialized with grades {10.0, 5, 3.5, 6, 4}
--update grades of ms1 with values {10.5, 40}
--update grades of ms2 with these values {8, -6.5, 20.0}
--print out the total number of students by calling the appropriate function in class *Student*
--list the grades for students ms1 and ms2 by calling the appropriate function; note that the intention of the client is that ms1 should have grades {10.5, 40} and ms2 should have grades {10.0, 5, 3.5, 6, 4, 8, -6.5, 20.0}

Deliverable: Upload UseStudent.java, your (modified) version of Student.java, and the exception class source file to Moodle.

Problem 2 (10 points)

Write a class *BasicNum* that contains a function called *gcd* with the following header (shown along with specifications):

```
//@ requires: ! (x==0 && y == 0)
//@ signals_only:  ZeroException (a user-defined, checked
 exception)
//@ ensures: throws ZeroException if both x and y are zero,
//@              otherwise returns the gcd of x and y.
public static int  gcd(int x, int y) throws ZeroException
```

You will need to define the *ZeroException* class.

The method *gcd* computes the greatest common divisor (GCD) of two numbers. The GCD of two numbers, both of which cannot simultaneously be zero, is the greatest positive number that divides both evenly. For example GCD(2, 4) is 2; GCD (2, 9) is 1, GCD(75, 50) is 25, GCD(8, 1) is 1, GCD(3, 0) = 3, GCD(0, 8) = 8.

Note that GCD(a, b) = GCD(b, a), and GCD(a, 0) = a.

Write a client class *CheckGCD* that tests your implementation (*BasicNum.gcd*) by calling it with all the 6 pairs of numbers shown above, and an additional 4 times (so a total of 10 calls).

Deliverable: Upload BasicNum.java, ZeroException.java and CheckGCD.java to Moodle.

Problem 3 (10 points)

Write a class *ArrSum* containing a standalone procedure *sum* with the following header:

```
public static int sum(int[] a)
```

sum takes an integer array as an argument, and returns the sum of all array elements back to the caller.

Write appropriate specifications (precondition, postcondition, and frame properties using assignable/modifies) for *sum* that describe its behavior to all potential clients. Also provide an implementation of *sum* (that meets your specifications).

Write a driver class *TestSum* with at least 3 different calls to *sum*.

Note: You must *explicitly* write all relevant specifications, i.e. the precondition, postcondition, and the frame condition. For example, even if the precondition for your implementation is 'true', you must have a requires clause that says this.

Deliverable: Upload the file with the source for *ArrSum* and also the client source/driver file to Moodle.

Problem 4 (10 points)

Solve problem 4.3 from the textbook. Write the solution in a plain text file called user_hw2_prob4.txt.

Deliverable: Upload user_hw2_prob4.txt to Moodle.

End