Name: ______________________________ Left class at: ___________

CPSC 233 – Coding Challenge 1 – Practice 1 and 2– Winter 2019

This is a practice coding challenge. You may ask other students and your TA any questions you wish. Note that you are expected to complete the actual coding challenge independently. Having a solution for this coding challenge will not help you come up with your own solution for the actual coding challenge. Only if you can solve this practice independently, you can be confident that you can complete the actual coding challenge successfully.

A coding challenge is similar to a test or exam: you get a limited amount of time to complete it and you must do so independently. But you'll need different skills for this type of a test than you would for a written or MCQ test. The main challenge is that you must have code that compiles and that compiles against the provided tests. If this is not the case, you will receive an F for the test, even if the compile error is minor. This means you're your first priority must be to have code that compiles. See the resources provided (both videos and text) to see how you can successfully ensure you always have code that compiles ready for submission.

Also remember that for the actual coding challenge you complete in the tutorial, the following rules apply:

- To complete the exercise, you may use your text, **code and other resources posted on D2L** and Java resources on-line.
- But the solution must be your own. Do **not** communicate with another person. Do not contact someone on-line. Do no talk with anyone in the class about the exercise.
- You can ask the TA questions. But consider the following limits on the help that TAs are allowed to provide.
  - TAs may help with technical issues in D2L or WebCAT or lab computer. This includes not finding the coding challenge on D2L or WebCAT, not being able to log into required resources. It does NOT include help with test or compile errors.
  - You may ask your TA for help with an uncommon compile or testing error. Your TA may determine that you should know how to manage such an error and may not be able to provide further information.
  - Your TA is **not** allowed to help explaining basic Java concepts and how to use testing tools. You are expected to know how to use these.
  - Your TA is **not** allowed to help when you have problems with your own computer or the configuration on your own computer. In those cases, you are expected to use the computers provided in the classroom.

Coding challenges must be completed during the tutorial and in the classroom. (Contact the instructor if you are registered with Student Accessibility Services and you need accommodations for the coding challenges.)

- If you arrive late, you must sign in with the TA, who has taken attendance at the start of class.
- If you leave early, you must hand-in this instruction sheet with the time you are leaving directly to the TA.

Name: ______________________________ Left class at: ___________

- All submissions must take place when you are in the tutorial. If there are submissions timed before sign-in or after sign-out, you will receive an F for this coding challenge.

It is NOT allowed to communicate with others about this coding challenge. Breaking any of the following rules will result in an automatic F for the Coding challenge:

- Do not have any communication devices visible.
- Do not have any communication software open on your computer. (They should all be closed, minimizing the software is NOT sufficient.)
- Do NOT communicate with others students about the coding challenge until the following week. Some students are writing a similar coding challenge later in the week.

Requirements

Remember to always create a 'skeleton' first and compile and test this using the provided JUnit test. If you are using Eclipse, make sure your submission pass the test cases on WebCaT. WebCAT will always be used to grade your solution. Use of Eclipse is not recommended for coding challenge. **Make sure you submit frequently!**

For a first practice, create a class called *BasicJavaP1* that contains the following functions:

1. *public static long floor(double num)*
   It should take the floor of the argument (round it down to nearest integer) and return the result. You may assume the number is non-negative.
2. *public static double conversion(double fahr)*
   The argument is a temperature in degrees Fahrenheit. The method should convert this to degrees Celsius and return the result.
3. *public static boolean willRoundUp(double num)*
   The method should return true if this number (the double) would be rounded up. (Recall that a number is rounded up if the decimal value is .5 or higher.) The method should return false otherwise. Do not use an if statement in your method. You can accomplish this using only arithmetic and boolean expressions.
4. *public static int sumRange(int start, int end)*
   This method should return the sum of all numbers starting at *start* and ending at *end*. The last number should be excluded in the computation. For example, if *start* is 2 and *end* is 7 then the result should be 2 + 3 + 4 + 5 + 6 = 20.
5. *public static int countChar(String str, char c)*
   Count the number of times the character stored in variable c appears in string str. An upper case and lower case character should be considered different characters (and your solution should be case sensitive).
6. *public static int addDigits(int num)*
   Your method should add the digits in the argument and return the result. For example 12345 result in the computation 1 + 2 + 3 + 4 + 5 = 15.

For a second practice, create a class called *BasicJavaP2* that contains the following functions:

1. *public static boolean isUpper(char aChar)*
   Return true if aChar is an upper case character and false otherwise.

Name: ______________________________ Left class at: ___________

2. *public static double computePolynomial(double x)*
   Solve the polynomial $(3-x)^2 + 4(7 + x) - 9$ for the *x* provided as an argument.
3. *public static long floorAfterMult(int num1, double num2)*
   Return the result of multiplying the two number (*num1* and *num2*) and return the floor of the result (rounded down to the nearest integer).
4. *public static boolean containsAllChars(String str, String chars)*
   Return true if *str* contains all the characters in the string *chars* and false otherwise. For example, if *str* is 'Hello' and *chars* is 'eo' then *containsAllChars* should return true but if *str* is 'Hello 'and *chars* is 'eoa' then *containsAllChars* should return false.

Do NOT use: Math, Integer, Character or StringBuilder classes in any of the practice functions or in the actual coding challenge. The tests will fail if you do so.

The descriptions give the basic expected functionality. The tests themselves provide additional information on functionality expected.

Finally, your actual coding challenge will be submitted using WebCAT. A WebCAT assignment is available for you to practice submissions with. Remember, WebCAT will automatically grade your submission and give feedback on errors. You can submit as many times as you wish until the deadline for the assignment has passed. WebCAT will use the grade of the LAST submission so make sure the last submission is the best one.

For the coding challenge, your submission is due at the end of tutorial. WebCAT remains open an additional 5 minutes after the deadline to ensure you can resubmit an earlier, better version. Do NOT use these 5 minutes to do additional work: it likely leads to code that will NOT work. Use this only to submit your best version.