CS-433 Advanced Programming

Homework # 1

Due: Monday, 24th September 2018 11:55 PM

1. Objective of assignment

- Understanding basics of java
- Basic usage of Eclipse IDE
- Understanding difference between mutable and immutable
- Understanding difference between primitive and non primitive data types.

2. Instructions

- Homework can only be done individually.
- Start as early as possible. Schedule your work accordingly.
- Follow some modular and good software engineering approach.
- It is advised that don't search for the solutions. They will influence you to implement in some specific way, which will be caught.
- While submitting assignment make a **RollNum_Name_Homework_1.zip** file including only the java source code files.
- Submit this zip file on **slate**
- Email me in case of any queries @ abdulmaroof.nuces@gmail.com
- For each passing day after deadline, 50% of the marks will be deducted

There are 3 exercises in this homework, each 10 points.
For this homework and for all the upcoming homeworks, follow the commenting and coding convention, unless otherwise specified explicitly. Name your files as specified in each of the exercises below.

1. **Write a Java program to experiment String and StringBuffer classes.** See their documentation at https://docs.oracle.com/javase/10/docs/api/java/lang/String.html and https://docs.oracle.com/javase/10/docs/api/java/lang/StringBuffer.html. Have a class named Homework_1 with a main function. The main function should expect exactly two command-line arguments. If the number of command-line arguments is not two, the main function should return immediately. Otherwise, it should print out the command-line arguments and continue as follows:
   - Create two String objects, say s1 and s2, initializing them with the first and the second command-line arguments. Experiment and print out the results of the following: s1.length(), s1.charAt(i) for all i for String s1, s1.equals(s2), s1.equalsIgnoreCase(s2), s1.compareTo(s2), s1.regionMatches(int toffset, s2, int offset, int len) for some offset and len, s1.regionMatches(boolean ignoreCase, int toffset, s2, int offset, int len) for some offset and len, s1.indexOf(c, i) for some c and i, s1.concat(s2), s1.replace(c1, c2), s1.upperCase(), s1.lowerCase().
   - Create two StringBuffer objects, say sbuf1 and sbuf2, initializing them with the first and the second command-line arguments. Experiment and print out the results of the following: sbuf1.length(), sbuf1.delete(int start, int end), sbuf1.deleteCharAt(int index), sbuf1.reverse() methods. Invoke sbuf1.replace() with proper arguments. Call the sbuf1.append().append() methods in a chain of method calls by passing primitive data types and sbuf2 as the parameters. Additionally, introduce a new class named MyClass in the same file but outside of StringTest class. MyClass should define nothing but the toString method that returns “This is my object”. Invoke sbuf1.append() with an object of MyClass. Invoke sbuf1.insert() method just like you did with the append().

2. **Write a Java program have the following steps. Note that in each step the StringBuffer is modified, and those modifications are retained in future steps. Also, the StringBuffer is printed out after each of the steps 5-10.**

- The user needs to enter two strings: one long string (say, 10 or so characters at a minimum) and a shorter string that is contained within the longer string. This input should be obtained via thenextLine() method, as using the next() method will not read in a string that contains spaces.
- Create a StringBuffer object from the longer string -- this is the StringBuffer that you will manipulate for the rest of the homework. There are two ways to do this: create a default constructred StringBuffer, and append() the long string to that, or use the StringBuffer with the appropriate specific constructor.
- Include, as a comment in your program, the code for creating the StringBuffer in the other way from step 2.
- Find the position of the small string within the StringBuffer, and save that position.
- Delete the small string from the StringBuffer, and print out the result.
- Insert "CS433" into the position of the StringBuffer where the small string was originally found (from step 3), and print out the result.
- Remove the last word from the string. You can assume that everything from the last space (found via lastIndexOf()) to the end of the String is the last word. Print out the result.
- Append " rocks" to the end of the StringBuffer, and print out the result. Note that there is a space before the work 'rocks'.
- Delete the character at position n/2, where n is the length of the StringBuffer. Print out the result.
- Reverse the StringBuffer, and print out the result.

Sample execution

Note that the text in red is what was input by the user. Your program needs to print out similar information, but the format does not have to be the exact same.
This program demonstrates the use of the StringBuffer class.

Enter a long string:
I think that this course really rules

Enter a shorter string within the first string:
This course

I think that really rules
I think that CS433 really rules
I think that CS433 really
I think that CS433 really rocks
I think that CS33 really rocks
skcor yllaer 33SC taht kniht I

3. Define a Student class with instance variables name, id, midterm1, midterm2 and final. Name is a string whereas others are all integers`. Also, add a static variable nextId, which is an integer and statically initialized to 1. Have some overloaded constructors. In each of them, the id should be assigned to the next available id given by nextId. The default constructor should set the name of the student object to "StudentX" where X is the next id. Add a calculateGrade() method which returns a string for the letter grade of the student, like "A", "B", "C", "D" or "F", based on the overall score. Overall score should be calculated as (30% of midterm1 + 30% of midterm2 + 40% of final). The letter grade should be calculated according to the absolute grading in FAST.

   Your test class, to be named as Homework1, should create 25 student objects with default constructor and invoke the setter methods for midterm1, midterm2 and final with random numbers ranging from 50 to 100 inclusive. After that, it should print the student information via the toString() method. Student information provided by toString() should include name, midterm1, midterm2, final and the letter grade given by the calculateGrade() method.

Good programming practices

The good programming practices given in (GoodProgrammingPractices.pdf) need to be followed in all homework's and assignments.

Good Luck! ☺