Quiz 1 Introduction to Java

Released: Friday, 31st March 2023
Due: 11.59pm , Sunday, 9th April 2023

Instructions

1. Copy the class file and complete the code (at TODO section).
2. Several test cases are provided in the main function to help you verify your output. However, you should write your own test case for testing. We will use other test cases for marking.
3. Copy and paste the "submission code" into the submission form provided in your lecture session. The submission code is different for every question. It is stated in the provided code and in the output after you have successfully compile and execute the program.

   For example, in Q1 output:
   ```
   The question is done!
   Submission code: multiply function
   ```

   You should copy and paste ONLY the multiply function to the submission form for Q1 as below:
   ```
   public static int multiply(int a, int b) {
           // TODO
           …… your code here
   }
   ```

4. Failure to follow the submission instructions will be granted ZERO mark.
5. You can submit multiple submission before the due date. Only the latest submission will be recorded so do not submit again after the quiz is due. (See point below)
6. Submission after the due date will be given ZERO mark.

Programming with assertions

An assertion is a statement in the Java™ programming language that enables you to test your assumptions about your program. For example, if you write a method that calculates the speed of a particle, you might assert that the calculated speed is less than the speed of light.

Each assertion contains a boolean expression that you believe will be true when the assertion executes. If it is not true, the system will throw an error. By verifying that the boolean expression is indeed true, the assertion confirms your assumptions about the behavior of your program, increasing your confidence that the program is free of errors.

Read more from here.
https://docs.oracle.com/javase/7/docs/technotes/guides/language/assert.html

Question 1

This is an introduction question, the purpose of which is to explain how to solve questions in Quizzes.

This question is the easiest one. Write a function that will receive 2 numbers as input and it should return the multiplication of these 2 numbers.

Input: Two arguments. Both are of type integer.

Output: integer.

Examples:

```
assert multiply(3, 2) == 6
assert multiply(0, 1) == 0
```

Note:
Assertion is disabled by default in Java. To enable it, you need to run your code by -ea or -enableassertions option.

You are encouraged to test your code with test cases of your own. All submitted code will be tested against other test cases during marking.

```
public class Q1 {
    // BEGIN ANSWER Q1 <-- Copy From This Line
    // Submission code: multiply function
    public static int multiply(int a, int b) {
        // TODO

    }
    // END  ANSWER Q1  <-- Copy Until This Line

    public static void main(String[] a) {
        System.out.println("Example: " + multiply(1, 2));

        assert multiply(3, 2) == 6: "Failed 3 x 2";
        assert multiply(0, 1) == 0: "Failed 0 x 1";

        System.out.println("The question is done!");
        System.out.println("Submission code: multiply function");
    }
}

/* Node: To enable assertion in Java:
   Compile by javac Q1.java
   Execute by java -ea Q1
   Refer to Lecture01 if you do know how to compile and execute by command lines
*/
```

Question 2

"king Kong" is a word game we will use to teach the robots about pattern matching. You should write a function that will receive a positive integer and return:

"King Kong" if the number contains both digit 2 and 3;
"King" to replace digit 2 in the number;
"Kong" to replace digit 3 in the number;;
The number as a string for other cases.

Input: A number as an integer.

Output: The answer as a string.

Example:
```
assert kingKong(23).equals("King Kong");
```

```
public class Q2 {
    // BEGIN ANSWER Q2 <-- Copy From This Line
    // Submission code: kingKong function
    public static String kingKong(int num) {
        // TODO

    }
    // END  ANSWER Q2  <-- Copy Until This Line

    public static void main(String[] a){
        System.out.println("Example: " + kingKong(23));

        // These "asserts" are used for self-checking
        assert kingKong(23).equals("King Kong"): "Failed 23";
        assert kingKong(2).equals("King"): "Failed 2";
        assert kingKong(3).equals("Kong"): "Failed 3";
        assert kingKong(12).equals("1King"): "Failed 12";
        assert kingKong(30).equals("Kong0"): "Failed 30";

        System.out.println("The question is done!");
        System.out.println("Submission code: kingKong function.");
    }
}

/* Node: To enable assertion in Java:
   Compile by javac Q2.java
   Execute by java -ea Q2
   Refer to Lecture01 if you do know how to compile and execute by command lines
*/
```

Question 3

You are given a string of text and two markers (the initial one and final). You have to find the text enclosed between these two markers. But there are a few important conditions.

- The initial and final markers are always different.
- The initial and final markers always exist in the text and go one after another.

Input: Three arguments. All of them are strings. The second and third arguments are the initial and final markers.

Output: A string.

Example:
```
assert extractWord("What is [apple]", "[", "]").equals("apple");
```

How it is used: *For text parsing.*

Precondition: *There can't be more than one final and one initial markers.*

```
public class Q3 {

    // BEGIN ANSWER Q3 <-- Copy From This Line
    // Submission code: extractWord function
    public static String extractWord(String text, String start, String end) {
        // TODO
    }
    // END  ANSWER Q3  <-- Copy Until This Line

    public static void main(String[] a){
        System.out.println("Example: " + extractWord("What is >apple<", ">", "<"));

        // These "asserts" are used for self-checking
        assert extractWord("What is </orange>", "</", ">").equals("orange"): "Failed
orange";
        assert extractWord("What is [apple]", "[", "]").equals("apple"): "Failed
apple";
        assert extractWord("What is ><", ">", "<").equals(""): "Failed empty";
        assert extractWord("[an apple]", "[", "]").equals("an apple"): "Failed an
apple";

        System.out.println("The question is done!");
        System.out.println("Submission code: extractWord function.");
    }
}
/* Node: To enable assertion in Java:
   Compile by javac Q3.java
   Execute by java -ea Q3
   Refer to Lecture01 if you do know how to compile and execute by command lines
*/
```