A6 (30 marks)

***Focus**: methods, Debugging*

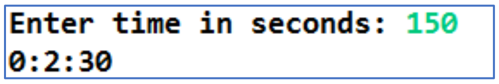
Q1. [10 marks] Create a method that transforms a time given in seconds to a String that shows the hours, minutes, and seconds like the time on a digital clock. The String returned from the method shows the time as hours:minutes:seconds. Use the following header:

public static String convertTime(int totalSeconds)

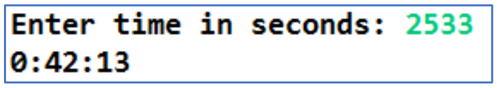
Write a test program that asks the user for the time in second then displays the time in hours:minutes:seconds.

Hint: Use the / operator to get the number of minutes, and the % operator to get the number of seconds left over.

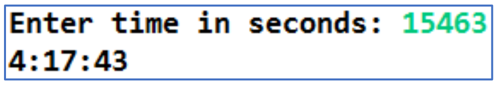
Sample runs



```
Enter time in seconds: 150
0:2:30
```



```
Enter time in seconds: 2533
0:42:13
```



```
Enter time in seconds: 15463
4:17:43
```

Q2. [8 marks] Write a class with 2 methods. The first method determines if any two sides added together are greater than the remaining side. The second method calculates the triangle's area.

// Method 1: Return true if the sum of every two sides is greater than the third.
public static boolean isValid(double sid1, double side2, double side3)

// Method 2: Return the triangle area
public static double area(double side1, double side2, double side3)

Develop a test program that takes in the three sides of a triangle and uses the *isValid( )*method to determine if the input is valid according to the above criteria. If the inputs are valid display the area. Otherwise display an error message and continue to ask for inputs until a valid set is received.

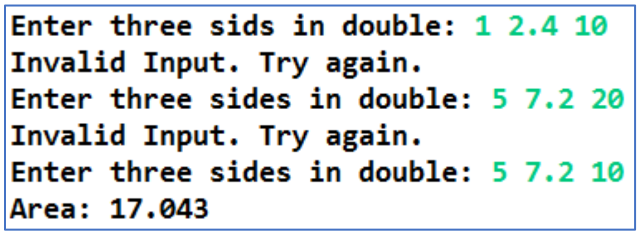
The formula for computing the area of a triangle is as follows:

$$s = (side1 + side2 + side3)/2$$

$$area = \sqrt{s(s - side1)(s - side2)(s - side3)}$$

Hint: use a while loop to check the validity of the user's input.

Sample run



```
Enter three sids in double: 1 2.4 10
Invalid Input. Try again.
Enter three sides in double: 5 7.2 20
Invalid Input. Try again.
Enter three sides in double: 5 7.2 10
Area: 17.043
```

Q3. [8 marks] Create two methods: one that will reverse the order of the digits in an integer and a second determines if a number is a palindrome (i.e. reads the same front to back) using the following method headers.

// Method 1: Takes in an integer and reverses it, e.g., reverse(456) returns 654
public static int reverse(int number)

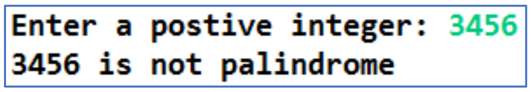
// Method 2: Takes in an integer and returns true if it is a palindrome. Use the reverse method you just wrote to implement this method.
public static boolean isPalindrome(int number)

Create a test program that asks the user to input an integer and provides feedback on whether the integer is a palindrome or not.

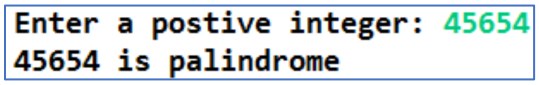
Hint: To find the reverse number, use one of the following approaches:

- Modify your numerical inputs to a string, then use a for loop to reverse the characters with string concatenation and the charAt() method.
- Retrieve individual digits and combine them in reverse order using the / and % operators, using ten as a divisor. Combine the digits with either mathematically or with string concatenation.

Sample runs



```
Enter a postive integer: 3456
3456 is not palindrome
```



```
Enter a postive integer: 45654
45654 is palindrome
```

Q4. [4 marks] This question has a quick tutorial on the Eclipse Debugger tool.

1. Copy the code below into Eclipse. Set a **breakpoint** on the line `int b = 20;`
   This is where you want to start to observe how your program proceeds.

```
public static void main(String[] args) {
    int a = 10;
    int b = 20;
    int count = countPerfectSquares(a, b);
    System.out.printf("Range [%d,%d] has %d of Perfect Squares.", a, b,
count);
}
public static int countPerfectSquares(int n1, int n2) {
    int count = 0;
    for (int i = n1; i <= n2; i++)
       if (isPerfectSquare(i))
          count++;
    return count;
}
private static boolean isPerfectSquare(int num) {
    int sq = (int)Math.sqrt(num);
    return (sq*sq == num);
}
```

2. Start the Eclipse debugger (or the debugger in your preferred IDE). As soon as Eclipse reaches the **breakpoint**, it will **pause** the program and wait for your instructions.

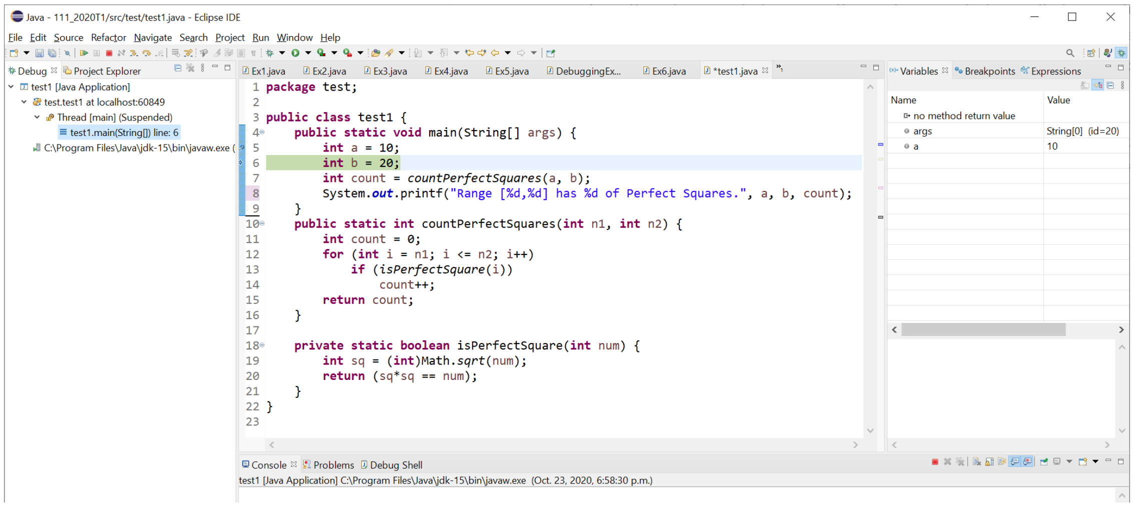
3. **Single-step** through your program, tracing the code and observing how the variables change.

Do the following as you trace your code:

1. Take **screenshots** right after each of the `main` method's variables, ***`b`*** **and** ***`count`***, have changed. For example, the screenshot below is taken once 'a' has changed (i.e. has been initialized). Your screenshots should be right after lines 6 and 7 have been finished executing.

2. Take **screenshots** right after you step into each of the methods **`countPerfectSquares()`** **and** **`isPerfectSquare()`**. Highlight in your screenshots the stack trace (the left part that shows the list of method calls).

3. You should now have four screenshots: 2 from step (1) above and 2 from step (2).
   Zip your screenshots into one file and submit it along with answers to Q1-3.



Java - 111_2020T1/src/test/test1.java - Eclipse IDE

```
test1 [Java Application]
  test.test1 at localhost:60849
    Thread [main] (Suspended)
      test1.main(String[]) line: 6
  C:\Program Files\Java\jdk-15\bin\javaw.exe (
```

```java
package test;

public class test1 {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        int count = countPerfectSquares(a, b);
        System.out.printf("Range [%d,%d] has %d of Perfect Squares.", a, b, count);
    }
    public static int countPerfectSquares(int n1, int n2) {
        int count = 0;
        for (int i = n1; i <= n2; i++)
            if (isPerfectSquare(i))
                count++;
        return count;
    }

    private static boolean isPerfectSquare(int num) {
        int sq = (int)Math.sqrt(num);
        return (sq*sq == num);
    }
}
```

Name	Value
no method return value	
args	String[0] (id=20)
a	10

```
test1 [Java Application] C:\Program Files\Java\jdk-15\bin\javaw.exe  (Oct. 23, 2020, 6:58:30 p.m.)
```

Submission Instructions

For this assignment, you need to do the following:

1. Create a Java project of which name consists of **your student number followed by the assignment number,** e.g., “1234567_A1”.
2. Create one class for each question and write your answer inside that class. Your classes should have the same name as the question number (e.g., Q1)
3. After solving all questions, open your file explorer.
4. Navigate to your Java project folder (can be found inside your Eclipse workspace folder).
5. Locate the “src” folder for this project (the folder that includes the source code for all questions).
6. Zip the “src” folder and rename the zipped file to match your project name (e.g., 1234567_A1.zip).
7. ***Don’t forget to include the screenshots from the last question in the same zip file you created above.***
8. Submit the zipped file **to Canvas.**

Note that you can resubmit an assignment, but the new submission overwrites the old submission and receives a new timestamp.