EECS1022 -Summer 2022- Lab02

Due Date: Friday, June 3, 2022, before 22:00

Contents

EECS1022 -Summer 2022- Lab02

Due Date: Friday, June 3, 2022, before 22:00

Instructions:

Important Reminders

- You should attend your lab session (the one you are enrolled in). If you need to change your lab enrollment, you should contact the Undergraduate Office in the department. ***Instructors or TAs cannot change your enrollment***.
- You can submit your lab work in eClass any time before 22:00 on Friday (**June 3, 2022**) of the week the lab is due. Your last submission will overwrite the previous ones, **and only the last submission will be graded**.
- The deadline is strict with no excuses: **you receive 0 for not making your electronic submission in time**. ***Emailing your solutions to the instructors or TAs will not be acceptable.***
- To submit your work, you need to use the York eClass.
- **Your submission will be graded by JUnit tests given to you and additional JUnit tests covering some other input values. This is to encourage you to take more responsibility for the correctness of your code by writing more JUnit tests.**
- Developing and submitting a correct solution for this lab without compilation errors is essential. Hence, it's important you take a reasonable amount of time to test your code in different ways. If you submitted a solution with a small mistake in terms of syntax or do not comply with lab instructions, then you may receive 0 as a grade for the implementation of this lab
- There will be **a 25% penalty** on your lab final grade if your submitted code does not compile due to ***minor compilation errors***, given that TAs can fix these minor compilation errors. **You will receive a zero if your code contains major compilation errors that TAs can not fix**.

Academic Honesty

- Students are expected to read the Senate Policy on Academic Honesty. See also the EECS Department Academic Honesty Guidelines.
- **All labs are to be completed individually: no group work is allowed. Do not discuss solutions with anyone other than the instructor or the TAs. Do not copy or look at specific solutions from the net. If you are repeating the course, you are not allowed to submit your own solution developed in previous terms or for other purposes. You should start from scratch and follow the instructions.**

EECS1022 -Summer 2022- Lab02

Due Date: Friday, June 3, 2022, before 22:00

Learning Objectives

- To create a static (`Utility`) class where all methods are static
- To use Java controls structure (selection structures, repetition structures, and nested loops)
- To use JUnit Tests to verify your work
- To create methods based on a given API

Important pre-lab works you need to do before going to the lab

a. http://wiki.eclipse.org/The_Official_Eclipse_FAQs
b. Testing using Eclipse IDE
    a. Video: Java Unit Testing with JUnit - Tutorial - How to Create And Use Unit Tests
c. Read about debugging using Eclipse IDE
    a. https://www.eclipse.org/community/eclipse_newsletter/2017/june/article1.php
    b. Video: How to set breakpoints to debug code in Java using Eclipse Debug mode
    c. Video: How To Debug Java Code The Right Way - Eclipse Debugger Full Tutorial

Feel free to find online resources on these and share them with your classmate on the discussion forum

Style Rules

The style rules are not overly restrictive in EECS1022.

1. Your programs should follow the standard Java conventions (class names begin with an uppercase letter, variable names start with a lowercase letter, **`public static final`** constants should be in all caps, etc.).
2. In general, use short but descriptive variable names. There are exceptions to this rule; for example, traditional loop variables are often called i, j, k, etc.
3. Avoid very long names; they are hard to read, take up too much screen space, and easily mistype.
4. Use a consistent indentation size. Beware of the TAB vs SPACE problem: Tabs have no fixed size; one editor might interpret a tab as four spaces, and another might use eight spaces. If you mix tabs and spaces, you will have indenting errors when your code is viewed in different editors.
5. Use a consistent brace style:

```
// left-aligned braces
class X
{
  public void someMethod()
  {
    // ...
  }

  public void anotherMethod()
  {
    for (int i = 0; i < 1; i++)
    {
      // ...
    }
  }
}
```

or

```
// ragged braces
class X {
  public void someMethod() {
    // ...
  }

  public void anotherMethod() {
    for (int i = 0; i < 1; i++) {
      // ...
    }
  }
}
```

6. This one always causes problems for students. Insert a space around operators (except the period "."). The following

```
// some code somewhere
boolean isBetween = (x > MIN_VALUE) && (x > MAX_VALUE);
int someValue = x + y * z;
```

is much easier to read than this

```
// AVOID DOING THIS
// some code somewhere
boolean isBetween=(x>MIN_VALUE)&&(x>MAX_VALUE);
int someValue=x+y*z;
```

7. Avoid using "magic numbers". A magic number is a number that appears in a program in place of a named constant. For example, consider the following code:

```
int n = 7 * 24;
```

What do the numbers 7 and 24 mean? Compare the code above to the following:

```
final int DAYS_PER_WEEK = 7;
final int HOURS_PER_DAY = 24;
int n = DAYS_PER_WEEK * HOURS_PER_DAY;
```

In the second example, the meaning of 7 and 24 is now clear (better yet, rename n). Not all numbers are magic numbers. You can usually use 0, 1, and 2 without creating a named constant. If you ever find yourself doing something like:

```
final int TEN = 10;
```

Then you are probably better off using 10 and explaining its meaning in a comment.

8. A good IDE (integrated development environment) such as IntelliJ IDEA and eclipse will correct many style errors for you. In IntelliJ IDEA, you can select the code that you want to format, then choose *code -> Reformat Code* to format your code automatically.

EECS1022 -Summer 2022- Lab02

Due Date: Friday, June 3, 2022, before 22:00

Getting Started

1. Start eclipse.
2. **Download the starter code "Lab2.zip" from the eClass course site**
3. Import the test project by doing the following:
    1. Under the **File** menu, choose **Import...**
    2. Under **General,** choose **Existing Projects into Workspace** and press **Next**
    3. Click the **Select archive file** radio button, and click the **Browse...** button. You may have to wait about 10 seconds before the file browser appears.
    4. In the file browser that appears, navigate to your home directory.
    5. Select the file **Lab2.zip** and click **OK**
    6. Click **Finish**.
4. All files you need for this lab should now appear in eclipse.

Lab Structure

After successfully importing the starter code/project "Lab2.zip"

The lab folder/directory structure is as follows:

- **src/lab2**/: directory contains a Java file named **Utility.java**.
- **src/lab2**/: directory contains a Java file (JUnit test cases) named **JunitTest_Utility.java**. This file contains several JUnit test cases that can help to test your code.
  *It should be noted that you need to tun the JUnit tester **JunitTest_Utilities.java** after you complete the `Utility` class to check your work. Nonetheless, passing all given tests does not guarantee full marks for this lab. Therefore, you are required to write additional tests to ensure the correctness of your implementations.*
- **doc/**: directory contains Java documentations for lab2 in HTML format.

EECS1022 -Summer 2022- Lab02

Due Date: Friday, June 3, 2022, before 22:00

Lab Exercise

In this lab, you need to write Java code to implement four tasks (static methods).

Task1 Implementation:

```
public static String binaryRepresentation(short value)
```

Write a static method that receives a short value that can be stored in 16 bits and then convert this input value to 16 bits binary representation. The method returns the string value representing the binary value in 16 bits equivalent to the input value.

Use of arrays or any Java library class (e.g., array, ArrayList) is strictly forbidden. Violation of this will result in a 70% penalty on your marks. Try to solve this problem using Java Control Structures (selection structures, repetition structures, and nested Loops) only

Examples:

if value=15 then return "The decimal value (15) has binary representation [0000000000001111]"
if value=145 then return "The decimal value (145) has binary representation [0000000010010001]"
if value=255 then return "The decimal value (205) has binary representation [0000000011001101]"

Remember that double quotations are shown above to indicate the beginning and end of the string value, and these double quotations are not part of the return value.

Make sure the method compiles without errors and returns the correct result when invoked.

Parameters:

value input integer value of type short

Returns:

String value represents binary value in 16 bits. See the example above

Precondition: value is non-negative values

Task2 Implementation:

```
public static String countofMult357(int lowerLimit, int upperLimit)
```

Write a static method that counts the number of integers between a lower-limit integer and an upper-limit integer that are multiple of 3 or multiple of 5 or multiple 7.

Use of arrays or any Java library class (e.g., array, ArrayList) is strictly forbidden. Violation of this will result in a 70% penalty on your marks. Try to solve this problem using Java Control Structures (selection structures, repetition structures, and nested Loops) only

Examples:

if lowerLimit=0 and upperLimit= 5 then return "Between (1) and (5) there are (1) multiple of 3, (1) multiple of 5 and (0) multiple of 7"

if lowerLimit=10 and upperLimit= 10 then return "Between (10) and (10) there are (0) multiple of 3, (1) multiple of 5 and (0) multiple of 7"

if lowerLimit=10 and upperLimit= 22 then return "Between (10) and (22) there are (4) multiple of 3, (3) multiple of 5 and (2) multiple of 7"

if lowerLimit=7 and upperLimit= 5 then return "Error: lower limit (7) is not less than or equal to upper limit (5)"

Remember that double quotations are shown above to indicate the beginning and end of the string value, and these double quotations are not part of the return value.

Make sure the method compiles without errors and returns the correct result when invoked.

Parameters:
lowerLimit lower limit integer value
upperLimit upper limit integer value
Returns:
String value as shown in the examples above @pre.

Precondition: lowerLimit and upperLimit are non-negative values, and lowerLimit less than upperLimit

Task3 Implementation:

```
public static String perfectNumber(int n)
```

Write a static method named **perfectNumber** that accepts an integer maximum as its parameter and returns string of all **perfect numbers** up to and including that maximum. A perfect number is an integer that is equal to the sum of its proper factors, that is, all numbers that evenly divide it other than 1 and itself.

The smallest perfect number is 6, which is the sum of 1, 2, and 3.

Use of arrays or any Java library class (e.g., array, ArrayList) is strictly forbidden. Violation of this will result in a 50% penalty on your marks. Try to solve this problem using Java Control Structures (selection structures, repetition structures, and nested Loops) only

For example:

if n = 15 then return "Perfect numbers up to (15): [6]"
if n= 150 then return "Perfect numbers up to (150): [6 28]"
if n= 500 then return "Perfect numbers up to (500): [6 28 496]"

Remember that double quotations are shown above to indicate the beginning and end of the string value, and these double quotations are not part of the return value.
Make sure the method compiles without errors and returns the correct result when invoked.

Parameters:
n :int input integer
Returns:
String value as shown above

Precondition : You may assume that the integer n is No negative integer

Task4 Implementation:

```
public static String isPalindromeInt(int n)
```

Write a static method named isPalindromeInt to determine if the given input `n` integer is a palindrome integer.

For example:

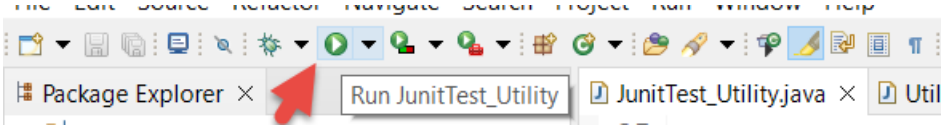
if n = 0 then return "Integer 0 is Palindrome"
if n= 10 then return "Integer 10 is NOT Palindrome"
if n= 3635363 then return "Integer 3635363 is Palindrome"
if n= 121121 then return "Integer 121121 is Palindrome"
if n= 112545214 then return "Integer 112545214 is NOT Palindrome"

Parameters:
n : int input value
Returns:
`n` is palindrome integer then return true. Otherwise, return false
Precondition : You may assume that the integer n is No negative integer

Verify and Test Your Implementation Using JUnit Test Cases

Step1. Inside the **JunitTest_Utility.java** file, you will find several JUnit test cases that can help to test your implementations.

Step2. You can run all the JUnit test cases by clicking on the green arrow.



EECS1022 -Summer 2022- Lab02

Due Date: Friday, June 3, 2022, before 22:00

Submit your work by using the course eClass

Check List:

Before submitting your files for this lab, you need to make sure you completed the following

	TASK1 implemented and tested
	TASK2 implemented and tested
	TASK3 implemented and tested
	TASK4 implemented and tested
	There is No compilation error generated from your implementation
	The **Utility.java** file contains the implementation for all the lab tasks.

Submit The Following File:

1) You need to submit the **Utility.java** file.