**CSIS2175 – Assignment 2**

***Adv Integrated Software Development***

**Name: Thiago Lyra Ganem\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Student No.: 300370930 \_\_\_\_\_\_**

**1 ) LeapYear**

Three methods were created in this java project.

A **getUserInput method** which asks for the user to enter an Integer as an input and return the Integer. However, if a non-integer is entered, the method will use a recursion to re ask the user to input a valid integer.  
  
**The leapCheck method** is checking if the integer number is a Leap Year or not, by using two if else statements. Being the first one a nested if.  
  
The firs if checks if it is a year that ends in 00, and if so it check if it is perfectly divisible by 400.   
The third if

**The testCode method** is just to check if the code is working. It is calling the leapCheck method with 1900 and 2016 as arguments. If 1900 returns false and 2016 returns true, the code is working.

**2) Extract Digit**

The project contains 4 methods.

A **getUserInput method** which asks for the user to enter an Integer as an input and return the Integer. However, if a non-integer is entered, the method will use a recursion to re ask the user to input a valid integer.

**The getArrayOfInt method** will return an Integer Array List that will be the reverse of the number entered.

The method will loop through the number entered until it is lower than 0. Inside each iteration, it will get the last number entered by the user by using the modulus operator by 10 and add the remainder into to Array List. Then, it will divide the number by 10 until it reaches 0.

**Loop Steps Ex:**  
1st Step 23%10 = 3 -> add 3 to Array  
2nd Step 23/10 = 2

3rd Step 2%10 ->2 add 2 to array  
4th Step 2/10 =0 quit loop.  
Final Array List = [3,2]

**The getArrayOfInt Method** takes an Array List as a parameter and uses a string builder to create the final string that will be printed in the console by looping through the Array and appending each number of the reversed Array List

If the digit is not the last one, it will also append a space.

**The testCode Method** is just to check if the code is working. It is calling both createString and getArrayOfInt methods and passing 123 as a parameter.

Then, it compares with the string literal “ 3 2 1” and will return Code is Working if it is working.

**3) Decimal Digits**

**a) Division**

Starting with the Division which was easier than the multiplication part. The idea was to replicate each step of the division.

I noticed that I had to approach all the division situations separately. The first one when a division has a remainder of 0, and the second one when the division has a remainder. Therefore, I build the flow below to better try to explain my thought.  
  
A diagram of a flowchart

Description automatically generated with low confidence

I built the code for the flow chart and used static variables so each recursion could be in the same state. The CalculateDivisionNumber method do the entire calculation, and the Division method just reset the static variables to the original state in case more than one division should be executed in the same code.

To test the method, I created a testDivison method, that compares if 20/3 = 6.666666666666666666 and 21/3 = 7.00000000000000000000 and 2/3 = 0.666666666666666666. If the comparison work the division code is working.

**B) Multiplication**

The idea was to replicate all steps of a multiplication. I’ve created the flow below to try to explain every step made in the code.  
The main goal of the multiplication was to create a 2d with the products of each multiplication, and then sum all the numbers in the array.

To test if the code was working, I’ve created a testCode method with 789\*592 and if the result was equal to 459198, the code is working.

**A diagram of a flowchart

Description automatically generated with low confidence**

**4) Perfect Java**

The Project has 3 methods.  
  
**The getEvenlyDividedNumbers** receives an int as a parameter and returns an array list of all items that can be divided into.

To do so, it loops from 0 until it reaches itself, and do a modulus operation, if the remainder of the operation is equal to 0 the number is added to the list.

**The getListSum** receives the array list as a parameter and return the sum of all numbers of the array.

**The testCode will** call both methods using 6 as an argument and compare with 6. If the result Is true, the code is working.

To display all the perfect numbers, I created a loop in the main method to loop from one to 1000 and add to an array list of perfect numbers all the numbers that if compared with itself returns a true value.

**5) Palindrome**

The project has 6 methods.

A **getUserInput method** which asks for the user to enter an Integer as an input and return the Integer. However, if a non-integer is entered, the method will use a recursion to re ask the user to input a valid integer.

**The getLettersInPhrase** receives a string as a parameter iterate into every character in the string and uses the isLetter phrase to add each letter into the character array list and returns this array list of characters.

**The build String** is a method that receives an array of characters, it loops through the array list and generates the String that will later be compared with the palindrome.

**The reverseArray** method receives an array of characters, and loops it from the last character to the first one and generates another array of characters with a reverse order.

**The palindromeChecker** method receives two strings that will be compared using the .equals(). If strings are a match, it will return true confirming the string is a palindrome.

**The testCode method**, call all the other methods with the String “Madam, I’m Adam” as an argument. If the code is working, it will print Code is working in the console.