JAVA ASSIGNMENT 1 STREAM: MCA

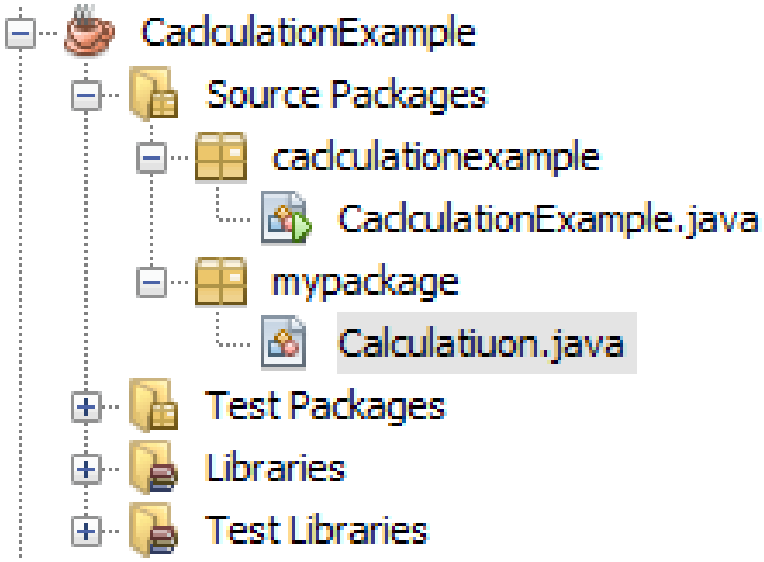
OBJECTIVE: To learn

- how to define multiple class in an application.
- How to use real numbers properly
- How to handle very large numbers in your application

1. Write a java program that finds the factorial value, check the prime number, and calculate the gcd value of two numbers.

   Consider the following implementation code/ driver code. Then complete the application by defining the class and functions properly.

   You should follow the following class hierarchy.



   The Calculation class contains the code for calculating factorial, prime number and gcd. This class contains the code for normal range integer value and also for BigInteger value.

   [Note: A prime number is a positive integer greater than 1 that has no positive integer divisors other than 1 and itself. In other words, a prime number is a number that is only divisible by 1 and itself.

   For example, the first few prime numbers are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, and so on.]

The implementation class (Driver code, i.e. main function) is shown below:

```
public class CaclculationExample {
    public static void main(String[] args) {
        // TODO code application logic here
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number = ");
        int num = sc.nextInt();
        BigInteger factorialValue = Calculatiuon.factorial(num);
        System.out.println("Factorial value for "+num+" = "+factorialValue);


        System.out.print("Enter the number to check Prime =  ");
        long n = sc.nextLong();
        boolean val = Calculatiuon.isPrime(n);
        System.out.println("Check the Prime Value = "+val);

        BigInteger nextPrime = Calculatiuon.nextPrime(n);
        System.out.println("Next Problable prime = "+nextPrime);

        System.out.println("GCD value =
"+Calculatiuon.gcdCalculation(6786540, 4587655));
    }//end of MAIN
}//end of CLASS
```

Take a good look at the following case studies before designing the class

Case 1:

```
run:
Enter the number = 5
Factorial value for 5 = 120
Enter the number to check Prime =  151
Check the Prime Value = true
Next Problable prime = 157
GCD value = 5
```

Case 2:

run:

Enter the number = 15
Factorial value for 15 = 1307674368000
Enter the number to check Prime = 779
Check the Prime Value = false
Next Problable prime = 787
GCD value = 5
BUILD SUCCESSFUL (total time: 27 seconds)

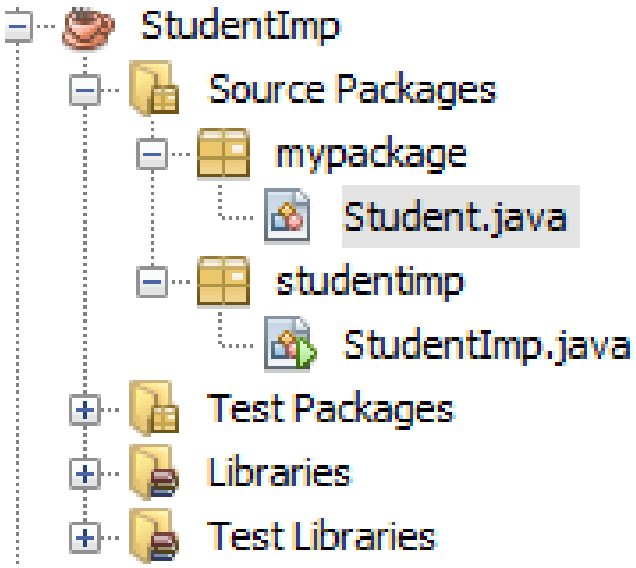
Case 3:

run:
Enter the number = 100
Factorial value for 100 =
93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000
Enter the number to check Prime = 67893452323
Check the Prime Value = true
Next Problable prime = 67893452339
GCD value = 5
BUILD SUCCESSFUL (total time: 30 seconds)

Case 4:
run:
Enter the number = 258
Factorial value for 258 =
56878465411882525024019869106095810443711861483651573510414327553915228320989004061260555958946091088710623453508969559004059093321200238492719713563110330058714032972946730753868022316902240107823463759719680928160493952481819610819007374844441824697652051115339179081141100212631179271854890626131975300539844183321097520081102320544748815063511124295784925759777654006636479482711740126764956099108357637301664918401416624491645555527002584252416000000000000000000000000000000000000000000000000000000000000000000
Enter the number to check Prime = 56789324567
Check the Prime Value = false
Next Problable prime = 56789324597
GCD value = 5

BUILD SUCCESSFUL (total time: 14 seconds)

2. Write a Java Application that creates a student class and have the following class hierarchy.



Create a class Student. Student class has the following attributes:

Student Name (dtype: String): stName_,
Student Roll (dtype: integer type): stroll_
Marks obtained by Student (dtype: integer type):stMarks_
And Student Year (dtype: String). :stYear_

Assign the values into the attributes using the constructor (use the constructor overloading). You can also use the getter/ setter methods. But before assigning the value your application should check the student roll number must belong between 1 to 60. The marks value should be less than 100.

The following attributes are initialized by the initializer list with following value:
Name and year is initialized with NULL. Roll and Marks is initialized by 0 and -1.

Some other functions are:

Function name: display
Return type: void.
Purpose: It displays the attributes of the object and each information is printed in the new line. The function declaration should assure that it does not change any attributes.

Function Name: displayRollName
Return type: Student
Purpose: It prints the roll and name of the student.

You may add any other functions in your representation as required.

Create Three students ([“Sutapa Sen”, 25, 76, “2010”] , [“Amal Basu”, 15, 85, “2010”], [“Hitesh Bagchi”, 31, 66,”2010”]).
Print the student information.

Now print the student name and roll number who scored highest marks.

3. Consider the following implementation class or driver class, write a Java application that defines the class Bank properly.

```
public class StudentImp {
   public static void main(String[] args) {
      Bank B1 = new Bank("01634001300000452","Biman Sen", 'S',
"55000.67");
      B1.display();
      System.out.println("After Deposit ...");
      B1.deposit("45987.59");
      B1.display();
      System.out.println("After With draw ...");
      B1.withDraw("10000");
      B1.display();
   }//end of main
}//end of class
```

The Bank constructor takes account number, Customer name, account type (S for savings or C for current), and amount. All these parameters are string type.
The display function displays the customer details.

The deposit function takes the amount in the string format and updates the balance amount.
Similarly, the withdraw function takes the withdrawal amount in string format. Before withdrawal the system checks the balance amount should be larger than 5000.

4. Write a Java class Complex for dealing with complex number. Your class must have the following features:

- Instance variables :
  realPart for the real part of type double
  imaginaryPart for imaginary part of type double.
- Constructor:
  public Complex (): A default constructor, it should initialize the number to 0, 0)
  public Complex (double realPart, double imaginaryPart): A constructor with parameters, it creates the complex object by setting the two fields to the passed values.
- Instance methods:
  public Complex add (Complex otherNumber): This method will find the sum of the current complex number and the passed complex number. The method returns a new Complex number which is the sum of the two.
  public Complex subtract (Complex otherNumber): This method will find the difference of the current complex number and the passed complex number. The method returns a new Complex number which is the difference of the two.
  public Complex multiply (Complex otherNumber): This method will find the product of the current complex number and the passed complex number. The method returns a new Complex number which is the product of the two.
  public void setRealPart (double realPart): Used to set the real part of this complex number.
  public void setImaginaryPart (double realPart): Used to set the imaginary part of this complex number.
  public double getRealPart(): This method returns the real part of the complex number
  public double getImaginaryPart(): This method returns the imaginary part of the complex number

public String toString(): This method allows the complex number to be easily printed out to the screen

Write a separate class **ComplexDemo** with a main() method and test the Complex class methods.

5. Write a Java program to find the k largest elements in a given array. Elements in the array can be in any order. (assume that k = 3) Complete the following code

```
public class KLargestArrayElements {

  public static void main(String[] args) {
    int[] arr = { 12, 45, 1, -1, 45, 16, 97, 100 };

    //Take input for k

    int[] kLargest = findKLargest(arr, k);

   System.out.println("The " + k + " largest elements in the array are: " +
Arrays.toString(kLargest));

  } //end of main function

  public static int[] findKLargest (int[] arr, int k) {
    // sort the array in ascending order

    // create a new array (named kLargest) to store the k largest elements

    // iterate over the last k elements in the sorted array


    return kLargest;
  } //end of findKLargest
}//end of class
```