Coding Question

**Q1- Write a Java program to find how many numbers in the array are Armstrong’s numbers.**

**Sample input: 153**

**Calculation: 1\*1\*1 + 5\*5\*5 + 3\*3\*3 = 153**

**Sample output: Is an Armstrong’s number**

**Another Sample input: 123**

**Calculation: 1\*1\*1 + 2\*2\*2 + 3\*3\*3 = 36**

**Sample output: Is not an Armstrong’s number**

**package** p1;

**public** **class** q\_1 {

**public** **static** **void** main(String[] args) {

**int** number = 153, originalNumber, remainder, result = 0;

originalNumber = number;

**while** (originalNumber != 0)

{

remainder = originalNumber % 10;

result += Math.*pow*(remainder, 3);

originalNumber /= 10;

}

**if**(result == number)

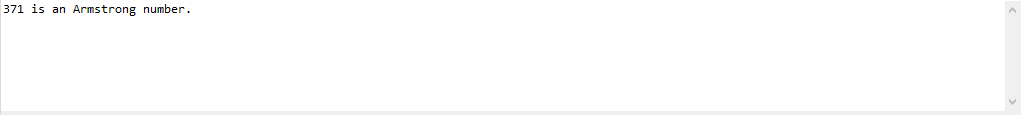
System.***out***.println(number + " is an Armstrong number.");

**else**

System.***out***.println(number + " is not an Armstrong number.");

}

}





**Q2- Write a Java program to find the total number of Tokens in the string.**

**You need to keep the track of following delimiters**

**‘@’, ‘#’, ‘ ’, ‘,’, ‘\*’**

**You need to make a Class named as StringTokenizer with private string data member and method int countTokens().**

**Sample Input: Hello@my#name\*is,ABC XYZ**

**Sample Output: Count = 6**

**package** p1;

**import** java.util.\*;

**import** java.util.Scanner;

**public** **class** q\_1

{

**public** **static** **class** StringTokenizer

{

**private** String str;

StringTokenizer ()

{

str=**null**;

}

StringTokenizer(String str)

{

**this**.str = str;

}

**public** **int** countTokens(String str)

{

**int** count=1;

**char**[] ch = **new** **char**[str.length()];

**for** (**int** i = 0; i < str.length(); i++)

{

ch[i] = str.charAt(i);

**if**(ch[i]==' ' || ch[i] == '#'|| ch[i] == '@'|| ch[i] == ','|| ch[i] == '\*')

{

count++;

}

}

**return** count;

}

}

**public** **static** **void** main(String []args)

{

String str = "Hello@my#name\*is,ABC XYZ";

System.***out***.println("Enter String: "+str);

StringTokenizer str1 = **new** StringTokenizer(str);

System.***out***.println("Count: "+str1.countTokens(str));

}

}



**Q3- Perform Encapsulation(getters) for the following members in RunEncapTest and write Output of the code?**

public class RunEncapTest {

private String Name;

private int Age;

private int id;

//write default constructors and parameterized constructor with three parameters

//write getters

//write setter

}

public static void main(String args[ ][ ])

{

EncaptulationTest encap = new EncapsulationTest();

//call setters for initialization

//call getter and print values

}

Sample Output:

This would produce following output: Name: John Age: 28 Id: 12345.

**package** p1;

**import** java.util.\*;

**import** java.util.Scanner;

**public** **class** q\_1

{

**public** **static** **class** RunEncapTest

{

**private** String Name;

**private** **int** Age;

**private** **int** id;

RunEncapTest()

{

Name=**null**;

Age=0;

id=0;

}

RunEncapTest(String Name,**int** Age,**int** id)

{

**this**.Name = Name;

**this**.Age= Age;

**this**.id = id;

}

**public** **void** setName(String Name)

{

**this**.Name = Name;

}

**public** **void** setAge(**int** Age)

{

**this**.Age = Age;

}

**public** **void** setId(**int** id)

{

**this**.id = id;

}

**public** String getName()

{

**return** Name;

}

**public** **int** getAge()

{

**return** Age;

}

**public** **int** getid()

{

**return** id;

}

**public** **void** data()

{

System.***out***.println("Name: "+Name+" Age: "+Age+" ID: "+id);

}

}

**public** **static** **void** main(String []args)

{

RunEncapTest encap=**new** RunEncapTest();

encap.setName("John");

encap.setAge(28);

encap.setId(12345);

System.***out***.println("Name: "+encap.getName()+" Age: "+encap.getAge()+" ID: "+encap.getid());

RunEncapTest encap1 = **new** RunEncapTest("Ahmer",24,123);

encap1.data();

}

}

**Q4-** **Solve the problem given below**

1. A class named ***Operation*** with a method named ***reverse*** that takes integers as parameters and returns an integer denoting their sum.
2. A class named ***Operator*** that inherits from a superclass named *Operation*.

Your classes should not be public.

**Input Format**

You are not responsible for reading any input from user; a locked code stub will test your submission by calling the *reverse* method on an *Operator* object and passing it **String** parameter.

**Output Format**

You are not responsible for printing anything. Your *reverse* method must return the reverse of the string and the position of asterisks should not change.

**Sample Input:**

* ABC\*\*bca\*
* \*hel\*\*l\*o

**Sample Output:**

The *main* method in the *Solution* class above should print the following:

My superclass is: Operation

* acb\*\*CBA\*
* \*oll\*\*e\*h

**Q5- Write a program with a parent class animal. Inside it define a name and an age variables, and set\_value() function.Then create two base variables Zebra and Dolphin which write a message telling the age, the name and giving some extra information (e.g. place of origin).**

**package** proj1;

**public** **class** test

{

**public** **static** **class** animal

{

**protected** String Name;

**protected** **int** Age;

animal()

{

Name=**null**;

Age=0;

}

**public** **void** setName(String Name)

{

**this**.Name = Name;

}

**public** **void** setAge(**int** Age)

{

**this**.Age = Age;

}

}

**public** **static** **class** Zebra **extends** animal

{

**public** **void** zebra()

{

System.***out***.println("Zebra Name"+ Name+" Age: "+Age+". The birth place is Sindh,Pakistan" );

}

}

**public** **static** **class** Dolphin **extends** animal

{

**public** **void** dolphin()

{

System.***out***.println("Dolphin Name:"+ Name+". Age: "+Age+" Years. The birth place is Indus River,Pakistan." );

}

}

**public** **static** **void** main(String[] args)

{

Zebra z = **new** Zebra();

z.setName("John Doe");

z.setAge(10);

Dolphin d = **new** Dolphin();

d.setName("John Doe");

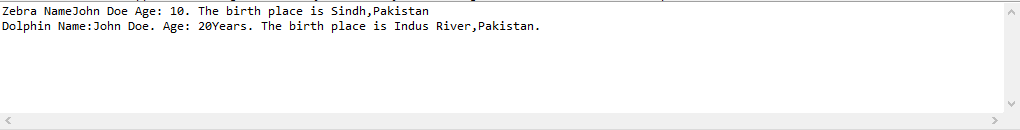
d.setAge(20);

z.zebra();

d.dolphin();

}

}



**Q- Run and observe the behavior of the code and fix the problem in the code given below.**

**Sample Output:**

Hello I am a motorcycle, I am a cycle with an engine.

My ancestor is a cycle who is a vehicle with pedals.

class BiCycle{

    String define\_me(){

        return "a vehicle with pedals.";

    }

}

class MotorCycle extends BiCycle{

    String define\_me(){

        return "a cycle with an engine.";

    }

    MotorCycle(){

        System.out.println("Hello I am a motorcycle, I am "+ define\_me());

        String temp=super.define\_me();

        System.out.println("My ancestor is a cycle who is "+ temp );

    }

}

class Solution{

    public static void main(String []args){

        MotorCycle M=new MotorCycle();

}

}

**package** p1;

**class** BiCycle{

String define\_me(){

**return** "a vehicle with pedals.";

}

}

**class** MotorCycle **extends** BiCycle{

String define\_me(){

**return** "a cycle with an engine.";

}

MotorCycle(){

System.***out***.println("Hello I am a motorcycle, I am "+ define\_me());

String temp=**super**.define\_me();

System.***out***.println("My ancestor is a cycle who is "+ temp );

}

}

**class** q\_1{

**public** **static** **void** main(String[] args){

MotorCycle M=**new** MotorCycle();

}

}

