**ADVANCED DATA STRUCTURE**

**GROUP G**

**ASSIGNMENT 13**

**YEAR: 2016-17**

**COLLEGE: VIIT**

**Date Of Completion : 30/03/2018**

**Title:**

The scope rules of the programming mechanism

**Problem Statement:**

Any application defining scope of Formal parameter, Global parameter, Local parameter accessing mechanism and also relevance to private, public and protected access. Write a Java program which demonstrates the scope rules of the programming mechanism.

**Objective:**

To use different access modifiers in JAVA.

**Software and Hardware Requirement:**

* 1. 64-bit Open source Linux or its derivative.
  2. Eclipse IDE

**Theory:**

**Access Modifiers**

There are two types of modifiers in java: access modifiers and non-access modifiers.

The access modifiers in java specifies accessibility (scope) of a data member, method, constructor or class.

There are 4 types of java access modifiers:

1. private
2. default
3. protected
4. public

There are many non-access modifiers such as static, abstract, synchronized, native, volatile, transient etc. Here, we will learn access modifiers.

* 1. private access modifier

The private access modifier is accessible only within class.

* 1. default access modifier

If you don’t use any modifier, it is treated as default bydefault. The default modifier is accessible only within package.

* 1. protected access modifier

The protected access modifier is accessible within package and outside the package but through inheritance only.

The protected access modifier can be applied on the data member, method and constructor. It can’t be applied on the class.

* 1. public access modifier

The public access modifier is accessible everywhere. It has the widest scope among all other modifiers.



**Code 1 :**

public class Scope

{

static int x = 11; private int y = 33; protected int a=100; public static void main(String args[])

{

Scope t = new Scope(); t.method1(5);

System.out.println("protected a : "+t.a);

}

public void method1(int x)

{

Scope t = new Scope(); this.x = 22; y = 44;

this.a=200;

System.out.println("Scope.x: " + Scope.x);

System.out.println("t.x: " + t.x);

System.out.println("t.y: " + t.y);

System.out.println("Local Variable y: " + y);

System.out.println("Local Variable x: " + x);

System.out.println("protected a : "+t.a);

}

/\*int a = 5; for (int a = 0; a < 5; a++)

{

System.out.println(a); //Error due to Re defining of int a

}\*/

}

**Output 1:**

Scope.x: 22

t.x: 22

t.y: 33

Local Variable y: 44

Local Variable x: 5 protecteda:100 protected a: 200

**Code 2:**

class Test

{

int x, y;

Test()

{

x = 10; y = 20;

} }

class Scope2 {public static void main (String [] args)

{

Test ob1 = new Test();

System.out.println(ob1.x + " " + ob1.y);

// Creating a new reference variable ob2

// pointing to same address as ob1

Test ob2 = ob1;

// Any change made in ob2 will be reflected

// in ob1 ob2.x= 100;

System.out.println(ob1.x+" "+ob1.y);

System.out.println(ob2.x+" "+ob2.y);

}

}

**Output 2:**

20

100 20

100 20

**Code 3 :**

// A Java program to demonstrate shallow copy

// using clone()

// An object reference of this class is

// contained by Test2 class Test

{

int x, y;

}

// Contains a reference of Test and implements // clone with shallow copy.

class Test2 implements Cloneable

{ int a; int b;

Test c = new Test();

public Object clone() throws CloneNotSupportedException

{ return super.clone();

} }

public class Scope3

{ public static void main(String args[]) throws CloneNotSupportedException

{

Test2 t1 = new Test2();

t1.a = 10;

t1.b = 20;

t1.c.x = 30;

t1.c.y = 40;

Test2 t2 = (Test2) t1.clone();

// Creating a copy of object t1 and passing

// it to t2 t2.a = 100;

// Change in primitive type of t2 will not

// be reflected in t1 field t2.c.x = 300;

// Change in object type field will be

// reflected in both t2 and t1(shallow copy)

System.out.println(t1.a + " " + t1.b + " " + t1.c.x + " " + t1.c.y);

System.out.println(t2.a + " " + t2.b + " " + t2.c.x + " " + t2.c.y);

}

}

**Output 3 :**

10 20 300 40

100 20 300 40

**Conclusion:**

After completing this assignment, we learnt to use different access modifiers in Java.