PRESIDENCY UNIVERSITY

CSE 2001 - Data Structures and Algorithms

LAB SHEET1

Local variables

The variables declared inside a method or a block are known as local variables. A local variable is visible within the method in which it is declared. The local variable is created when execution control enters into the method or block and is destroyed after the method or block execution is completed.

/* Java program to demonstrate Local variables */

```
public class LocalVariables {

        public void show() {
                int a = 10;
                //static int x = 100;
                System.out.println("Inside show method, a = " + a);
        }
        public void display() {
                int b = 20;
                System.out.println("Inside display method, b = " + b);
                // trying to access variable 'a' - generates an ERROR
                System.out.println("Inside display method, a = " + a);
        }
        public static void main(String args[]) {
                LocalVariables obj = new LocalVariables();
                obj.show();
                obj.display();
        }
}
```

Instance variables or member variables or global variables

The variables declared inside a class and outside any method, constructor or block are known as instance variables or member variables. These variables are visible to all the methods of the class. The changes made to these variables by method affects all the methods in the class. These variables are created separate copy for every object of that class.

/* Java program to demonstrate Global variables */

```
public class ClassVariables {

	int x = 100;

	public void show() {
		System.out.println("Inside show method, x = " + x);
		x = x + 100;
	}
	public void display() {
		System.out.println("Inside display method, x = " + x);
	}

	public static void main(String[] args) {
		ClassVariables obj = new ClassVariables();
		obj.show();
		obj.display();
	}
}
```

Static variables or Class variables

A static variable is a variable that is declared using **static** keyword. The instance variables can be static variables but local variables can not. Static variables are initialized only once, at the start of the program execution. The static variable only has one copy per class irrespective of how many objects we create.

/* Java program to demonstrate Static variables */

```
public class StaticVariablesExample
{
	int x, y;  // Instance variables
	static int z;  // Static variable

	StaticVariablesExample(int x, int y)
	{
		this.x = x;
		this.y = y;
	}
```

```
	public void show() {
		int a;  // Local variables
		System.out.println("Inside show method,");
		System.out.println("x = " + x + ", y = " + y + ", z = " + z);
	}

	public static void main(String[] args) {
		StaticVariablesExample obj_1 = new StaticVariablesExample(10, 20);
		StaticVariablesExample obj_2 = new StaticVariablesExample(100, 200);
		obj_1.show();
		StaticVariablesExample.z = 1000;
		obj_2.show();
	}
}
```

Final variables

A final variable is a variable that declared using **final** keyword. The final variable is initialized only once, and does not allow any method to change it's value again. The variable created using **final** keyword acts as constant. All variables like local, instance, and static variables can be final variables.

/* Java program to demonstrate Final variables */

```
public class FinalVariableExample
{
	final int a = 10;
	void show() {
		System.out.println("a = " + a);
		a = 20;  //Error due to final variable cann't be modified
	}

	public static void main(String[] args) {

		FinalVariableExample obj = new FinalVariableExample();
		obj.show();

	}

}
```

ARRAYS in Java

Creating an array

Syntax :

dataType[] arr; (or)
dataType []arr; (or)
dataType arr[];

Instantiation of an Array in Java:

arrayRefVar=new datatype[size];

//Java Program to illustrate how to declare, instantiate, initialize and traverse the Java array.

```
class Testarray
{
        public static void main(String args[])
        {
                int a[]=new int[5];    //declaration and instantiation
                a[0]=10;               //initialization
                a[1]=20;
                a[2]=70;
                a[3]=40;
                a[4]=50;
        //traversing array
        for(int i=0;i<a.length;i++)          //length is the property of array
                System.out.println(a[i]);
        }
}
```

```
//Java Program to illustrate the use of declaration, instantiation
//and initialization of Java array in a single line
class Testarray1
{
        public static void main(String args[])
        {
        int a[]={33,3,4,5};     //declaration, instantiation and initialization
        //printing array
                for(int i=0;i<a.length;i++)   //length is the property of array
                        System.out.println(a[i]);
        }
}
```

//Java Program to print the array elements using for-each loop

```
class Testarray1
{
	public static void main(String args[])
	 {
		int arr[]={33,3,4,5};
	//printing array using for-each loop
		for(int i:arr)
			System.out.println(i);
	}
}
```

//Java Program to demonstrate the way of passing an array to method.

```
public class TestArray
{

	//creating a method which receives an array as a parameter
	static void printArray(int arr[])
	{
		for(int i=0;i<arr.length;i++)
			System.out.println(arr[i]);
	}

	public static void main(String args[])
	{
		printArray(new int[]{10,22,44,66});	//passing array to method
	}
}
```

PRACTISE PROBLEM FOR STUDENTS

1. Program to print the duplicate elements of an array
2. Program to print the second largest element in an array
3. Ms. White needs to analyze students' test performance. She needs a program to display a name and let her enter the test score. Then it should compute and display the average. Finally it should give a report with names, scores, and deviation from the average.