**IT1013: Programming in Java**

**Program manual**

**UNITII**

**Practice Programs to illustrate the concepts**

**Program1:**

**//Illustrates the concept of Constructor and Constructor Overloading**

class **CallConstructorDemo**{

//Instance variables

float width;

float height;

float depth;

//Constructor for initialization and automatically called on object creation

//Constructor with no arguments

**CallConstructorDemo()**

{

width=20;

height=45;

depth=67;

}

**CallConstructorDemo(float w,float h,float d)**

{

width=w;

height=h;

depth=d;

}

**CallConstructorDemo(int w,int h,int d)**

{

width=w;

height=h;

depth=d;

}

void **volume()**

{

System.out.println("Volume is "+width\*height\*depth);

}

}

class **ConstructorOverloadDemo**{

public static void main(String args[])

{

**CallConstructorDemo obj=new CallConstructorDemo();**//Note () in CallConstructorDemo(). This calls the constructor "CallConstructorDemo()" along creation of object "obj"

**CallConstructorDemo obj1=new CallConstructorDemo(0.56F,0.3590F,0.878F);**

**CallConstructorDemo obj2=new CallConstructorDemo(33,56,67);**

obj.volume();

obj1.volume();

obj2.volume();

}

}

**Program2:**

**//Illustrates the concept of Method Overloading**

class **CallMethodOverloadingDemo**{

//Instance variables

float width;

float height;

float depth;

**CallMethodOverloadingDemo()**

{

width=45.67f;

height=56.87f;

depth=78.96f;

}

void **volume()**

{

System.out.println("Volume is "+width\*height\*depth);

}

float **volume(float w,float h,float d)/**/Overloaded method

{

width=w;

height=h;

depth=d;

return width\*height\*depth;

}

}

class **MethodOverloadingDemo**{

public static void main(String args[])

{

**CallMethodOverloadingDemo obj=new CallMethodOverloadingDemo();**

**obj.volume();**

float vol=obj.volume(2,4,5);

System.out.println("Volume through overloaded method: " + **vol**);

} }

**Program3:**

**//Program to illustrate Object Passing**

class **Test** {

int a, b;

**Test(int i, int j)** {

a = i;

b = j;

}

// return true if o is equal to the invoking object

**boolean equals(Test o)** {

if(o.a == a && o.b == b) return true;

else return false;

}

}

class **ObjPassing** {

public static void main(String args[]) {

**Test ob1 = new Test(100, 22);**

**Test ob2 = new Test(100, 22);**

**Test ob3 = new Test(-1, -1);**

System.out.println("ob1 == ob2: " +

ob1.equals(ob2));

System.out.println("ob1 == ob3: " +

ob1.equals(ob3));

}

}

**Program4:**

**//Illustrates the concept of Object return**

class **Test** {

int a;

**Test(int i)** {

a = i;

}

Test **incrByTen()** {

Test temp = new Test(a+10);

return temp;

}

}

class **RetOb** {

public static void main(String args[]) {

Test ob1 = new Test(2);

Test ob2;

ob2 = ob1.incrByTen();

System.out.println("ob1.a: " + ob1.a);

System.out.println("ob2.a: " + ob2.a);

ob2 = ob2.incrByTen();

System.out.println("ob2.a after second increase: "

+ ob2.a);

}

}

**Program 5:**

**//Program to illustrate static, staic variable and static method**

public class **StaticExample**{

**static** {

System.out.println("This is first static block");

}

public **StaticExample()**{

System.out.println("This is constructor");

}

public **static String staticString** = "Static Variable";

**static** {

System.out.println("This is second static block and "

+ staticString);

}

**static void display()**{

System.out.println("I am static method");}

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

StaticExample statEx = new StaticExample();

StaticExample.staticMethod2();

display();

System.out.println("I am a static variable: "+StaticExample.staticString);

}

**static** {

staticMethod();

System.out.println("This is third static block");

}

**public static void staticMethod()** {

System.out.println("This is static method");

}

**public static void staticMethod2()** { System.out.println("This is static method2");

}

}

**Program 6:**

**//Program to illustrate the concept of arrays**

import java.util.\*;

class **Arrays\_demo\_method**{

int r\_No[]=new int[2];

//static int c=89;

**void display(){**

Scanner s1=new Scanner(System.in);

r\_No[0]=s1.nextInt();

r\_No[1]=s1.nextInt();

System.out.println("Value at index0 of r\_No array " + r\_No[0]);

System.out.println("Value at index1 of r\_No array " + r\_No[1]);

System.out.println("length of r\_No array is" + r\_No.length);

}

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

{

Arrays\_demo\_method obj1=new Arrays\_demo\_method();

System.out.println("Value at index0 of args array " + args[0]);

System.out.println("Value at index1 of args array " + args[1]);

System.out.println("Value at index2 of args array " + args[2]);

System.out.println("length of args array is" + args.length);

obj1.display();

//System.out.println("Value of static variable in main because a static variable can be accessed from static context" +c);

}

}

**Program7:**

**//Program to illustrate the concept of finalize() and garbage**

**Collection**

/\*\* Example shows garbage collector in action Note that the finalize() method of object GC1 runs without being specifically called and that  
the id's of garbage collected objects are not always sequential.  
\*/  
class **TestGC** {  
public static void main(String[] args) {  
Runtime rt = **Runtime.getRuntime();**System.out.println("Available Free Memory: " + **rt.freeMemory()**);  
for(int i=0; i<10000; i++ ) {  
GC1 x = new GC1(i);   
}  
System.out.println("**Free Memory before call to gc():** " +   
rt.freeMemory());  
**System.runFinalization();**  
System.gc();  
System.out.println(" **Free Memory after call to gc():** " +   
rt.freeMemory());   
}  
}  
**class GC1** {  
String str;  
int id;  
**GC1(int i)** {  
this.str = new String("abcdefghijklmnopqrstuvwxyz");  
this.id = i;  
}  
**protected void finalize()** {  
System.out.println("GC1 object " + id + " has been finalized."); }   
}

**Program 8:**

**//Program to illustrate the concept of Nested and Inner Classes**

class Outer {

int outer\_x = 100;

void test() {

Inner inner = new Inner();

inner.display();

}

// this is an inner class

class Inner {

void display() {

System.out.println("display: outer\_x = " + outer\_x);

}

}

}

class InnerClassDemo {

public static void main(String args[]) {

Outer outer = new Outer();

outer.test();

} }

**Program8a: Nested Class**

package com.java2novice.nested.classes;

public class MyBasicStaticMemberClass {

    public static class MyStaticMemberExampleClass {

        public void printStatus() {

            System.out.println("Hey I am inside static member class");

        }

    }

    public static void main(String a[]) {

        StaticMemberTestClass smt = new StaticMemberTestClass();

        smt.testMemberClass();

    }

}

class StaticMemberTestClass {

    public void testMemberClass() {

        MyBasicStaticMemberClass.MyStaticMemberExampleClass msme

                    = new MyBasicStaticMemberClass.MyStaticMemberExampleClass();

        msme.printStatus();

    }

}

**Program 9:**

**//String and StringBuffer**

**Note: work out all tutorial Questions**

// Demonstrating some String methods.

class StringDemo2 {

public static void main(String args[]) {

String strOb1 = "First String";

String strOb2 = "Second String";

String strOb3 = strOb1;

System.out.println("Length of strOb1: " +

strOb1.length());

System.out.println("Char at index 3 in strOb1: " +

strOb1.charAt(3));

if(strOb1.equals(strOb2))

System.out.println("strOb1 == strOb2");

else

System.out.println("strOb1 != strOb2");

if(strOb1.equals(strOb3))

System.out.println("strOb1 == strOb3");

else

System.out.println("strOb1 != strOb3");

}

}

**Program 10: valueOf()**

public class ValueOfDemo {

public static void main(String args[]){

double d = 102939939.939;

boolean b = true;

long l = 1232874;

char[] arr = {'a', 'b', 'c', 'd', 'e', 'f','g' };

System.out.println("Return Value : " + String.valueOf(d) );

System.out.println("Return Value : " + String.valueOf(b) );

System.out.println("Return Value : " + String.valueOf(l) );

System.out.println("Return Value : " + String.valueOf(arr) );

}

}