**IT1013: Programming in Java**

**Program manual**

**UNITIII**

**Practice Programs to Illustrate the concepts**

**Program1: Inheritance, Abstract Class, Finalize, Constructor Inheritance**

abstract class Inheritance{

abstract void display();

}

public class Inheritance\_Demo extends Inheritance{

void display(){System.out.println("Happy Holi");}

public static void main(String a[]){

Inheritance\_Demo ob2=new Inheritance\_Demo();

Inheritance\_Demo ob1=new Inheritance\_Demo();

ob1.display();

}

}

**Program1a: Constructor Inheritance**

class A{

A(){System.out.println("A's");}

}

class B extends A{

B(){

System.out.println("B's");

} }

public class Constructor\_Inheritance\_Demo {

public static void main(String a[]){

B ob1=new B(); } }

**Program1b: Interface Demo**

interface Interface{

int a=25;

void display();

}

public class Interface\_Demo implements Interface {

public void display(){System.out.println("Java");}

public static void main(String a[]){

Interface\_Demo ob1=new Interface\_Demo();

ob1.display();

System.out.println(ob1.a);

} }

**Program1c: Finalize Demo**

public class Finalize\_Demo\_WithNull {

static Finalize\_Demo\_WithNull d1, d2 ;

public void show( ) {

System.out.println("Hello 1"); }

protected void finalize( ) throws Throwable {

if(d1 != null) {

System.out.println("d1 object is not eligible for garbage collection and is still active");

d1 = null;

if (d1 == null) System.out.println("d1 is not referenced and getting removed from memory"); }

if(d2 != null) { System.out.println("d2 object is not eligible for garbage collection and is still active");

d2 = null;

if(d2 == null) System.out.println("d2 is not referenced and getting removed from memory"); }

super.finalize( ); }

public static void main( String args[]) {

d1 = new Finalize\_Demo\_WithNull( );

d2 = new Finalize\_Demo\_WithNull( );

d1.show( );

d2.show( );

System.runFinalization();} }

**Program2: Using super**

public class Superclass {

public void printMethod() {

System.out.println("Printed in Superclass.");

}

}

Here is a subclass, called Subclass, that overrides printMethod():

public class Subclass extends Superclass {

// overrides printMethod in Superclass

public void printMethod() {

super.printMethod();

System.out.println("Printed in Subclass");

}

public static void main(String[] args) {

Subclass s = new Subclass();

s.printMethod(); }}

**Program3: Method Overriding and Dynamic Method Dispatch**

class A

{

void callme()

{

System.out.println("Inside A's callme method");

}

}

class B extends A

{

void callme()   // override callme()

{

System.out.println("Inside B's callme method");

}

}

class C extends A

{

 void callme() // override callme()

{

 System.out.println("Inside C's callme method");

}

}

public class Dynamic\_disp

{

public static void main(String args[])

{

 A a = new A(); // object of type A

 B b = new B(); // object of type B

C c = new C(); // object of type C

A r; // obtain a reference of type A

r = a; // r refers to an A object

r.callme(); // calls A's version of callme

r = b; // r refers to a B object

r.callme(); // calls B's version of callme

r = c; // r refers to a C object

r.callme(); // calls C's version of callme

}

}

**Program6: Package Demo with Access specifications**

**//Please refer to Java Complete Reference Book**