// Java program implementing Singleton class

// with using getInstance() method

// Class 1

// Helper class

class Singleton {

// Static variable reference of single\_instance

// of type Singleton

private static Singleton single\_instance = null;

// Decl;aring a variable of type String

public String s;

// Constructor

// Here we will be creating private constructor

// restricted to this class itself

private Singleton()

{

s = "Hello I am a string part of Singleton class";

}

// Static method

// Static method to create instance of Singleton class

public static Singleton getInstance()

{

if (single\_instance == null)

single\_instance = new Singleton();

return single\_instance;

}

}

// Class 2

// Main class

class GFG {

// Main driver method

public static void main(String args[])

{

// Instantiating Singleton class with variable x

Singleton x = Singleton.getInstance();

// Instantiating Singleton class with variable y

Singleton y = Singleton.getInstance();

// Instantiating Singleton class with variable z

Singleton z = Singleton.getInstance();

// Printing the hash code for above variable as

// declared

System.out.println("Hashcode of x is "

+ x.hashCode());

System.out.println("Hashcode of y is "

+ y.hashCode());

System.out.println("Hashcode of z is "

+ z.hashCode());

// Condition check

if (x == y && y == z) {

// Print statement

System.out.println(

"Three objects point to the same memory location on the heap i.e, to the same object");

}

else {

// Print statement

System.out.println(

"Three objects DO NOT point to the same memory location on the heap");

}

}

}

2. Hiearchy of an organisation (Employee.java,Manager.java,Labour.java)

Employee.java

import java.io.\*;

import java.util.\*;

public class Employee

{

void salary()

{

int sal=25000;

System.out.println("The salary of an Employee is Rs:" +sal);

}

public static void main(String[] args)

{

Employee e1 = new Employee();

e1.salary();

Manager m1=new Manager();

m1.salary();

Labour l1=new Labour();

l1.salary();

}

}

Manager.java

public class Manager extends Employee

{

int incentives=6000;

void salary()

{

int sal;

sal=25000+incentives;

System.out.println("The salary of a Manager is Rs:" +sal);

}

}

Labour.java

public class Labour extends Employee

{

int overtime=3;

void salary()

{

int sal;

sal=8000+(3\*200);

System.out.println("The salary of a Labour is Rs:" +sal);

}

}

3. create bank accounts and use polymorphism (BankAccounts.java)

import java.io.\*;

import java.util.\*;

public class BankAccounts

{

public static void main(String[] args) {

int a1,a2,TotalCash;

BankAccounts b1=new SavingsAccount();

BankAccounts b2= new CurrentAccount();

a1=((SavingsAccount) b1).cash();

a2=((CurrentAccount) b2).cash();

TotalCash=a1+a2;

System.out.println("Total cash in the bank is Rs."+TotalCash);

}

}

class SavingsAccount extends BankAccounts

{

int cash()

{

int FixedDeposit=100000;

return FixedDeposit;

}

}

class CurrentAccount extends BankAccounts

{

int cash()

{

int CashCredit=200000;

return CashCredit;

}

}

5. To draw shapes and use abstract.(Shape.java,Line.java,Rectangle.java,Cube.java)

Shape.java

import java.io.\*;

import java.util.\*;

public abstract class Shape

{

abstract void draw();

public static void main(String[] args)

{

Shape l=new Line();

l.draw();

Shape r=new Rectangle();

r.draw();

Shape c=new Cube();

c.draw();

}

}

Line.java

public class Line extends Shape

{

@Override

void draw() {

System.out.println("The shape is LINE");

}

}

Rectangle.java

public class Rectangle extends Shape

{

@Override

void draw()

{

System.out.println("The shape is RECTANGLE");

}

}

Cube.java

public class Cube extends Shape

{

@Override

void draw()

{

System.out.println("The shape is CUBE");

}

}

6.Abstract class persistence with file and database as its subclasses and a client class(Persistence.java,

FilePesistence.java,DatabasePersistence.java,Client.java)

Persistence.java

import java.io.\*;

import java.util.\*;

public abstract class Persistence

{

abstract void persist();

}

FilePesistence.java

public class FilePesistence extends Persistence{

@Override

void persist() {

System.out.println("The data is saved in the file...");

}

}

DatabasePersistence.java

public class DatabasePersistence extends Persistence {

@Override

void persist() {

System.out.println("The data in not saved in database yet....");

}

}

Client.java

import java.io.\*;

import java.util.\*;

public class Client

{

public static void main(String[] args)

{

Persistence fp=new FilePesistence();

fp.persist();

Persistence dp=new DatabasePersistence();

dp.persist();

}

}