1. Write a Singleton class. Confirm that singleton class cannot be inherited.

public class Singleton

{

private static Singleton singleton = null;

//default constructor

private Singleton()

{

s =”hello I am string part of Singleton class”;

}

public static Singleton getInstance()

{

if (singleton == null)

{

//object is created only Once in singleton

singleton = new Singleton();

return singleton;

}

}

}

Main.java

public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

Singleton instance1=Singleton.getInstance();

System.out.println(instance1);

}

}

Output

Singleton@6504e3b2

2. Program to describe the hierarchy of an organization.

class Employee{

float salary = 30000;

}

class Manager extends Employee{

double incentive = 5;

double sal=salary\*incentive;

public void salary() {

System.out.println("Manager salary is :" +salary);

System.out.println("Incentive for Manager:" +incentive);

}

}

class Labour extends Employee{

double overtime = 10;

double sal=salary\*overtime;

public void salary() {

System.out.println("Labour salary is :" +salary);

System.out.println("Overtime for Labour :" +overtime);

}

}

public class Hierarchy

{

public static void main(String args[]){

Manager m = new Manager();

Labour l = new Labour();

// All objects of inherited classes can access the variable of class Employee

m.salary();

l.salary();

//summing salary

salary(m.sal,l.sal);

}

static void salary(double a, double b) {

double addsal=a+b;

System.out.println("Sum of all employees salary :" +addsal);

}

}

Output

Manager salary is: 30000.0

Incentive for Manager: 5.0

Labour salary is: 30000.0

Overtime for Labour: 10.0

Sum of all employees salary: 450000.0

3.write program to consider saving and current bank holder.

import java.util.Scanner;

class Savings{

public void show(int fd) {

System.out.println("Fixed Deposit "+fd);

}

}

class Current{

public void show(int cc) {

System.out.println("Cash Credit "+cc);

}

}

class Totalcash{

public void show(int ssal,int csal) {

int sum;

sum=ssal+csal;

System.out.println("Total Cash : "+sum);

}

}

public class Bank {

public static void main(String[] args) {

// TODO Auto-generated method stub

int fdep,ccd;

Savings s = new Savings();

Current c = new Current();

Totalcash tc= new Totalcash();

Scanner scanner = new Scanner(System.in);

System.out.println("Enter Fixed deposit for savings account");

fdep = scanner.nextInt();

System.out.println("Enter Cash credit for Current account");

ccd = scanner.nextInt();

s.show(fdep);

c.show(ccd);

tc.show(fdep,ccd);

}

}

Output

Enter Fixed deposit for savings account

40000

Enter Cash credit for Current account

50000

Fixed Deposit: 40000

Cash Credit: 50000

Total Cash: 90000

4. Test the following principles of an abstract class:

i. if any class has any of its method abstract then you must declare entire class abstract

abstract.

public class demo extends call {

public void dispaly2()

{

System.out.println("HI");

}

public static void main(String[] args)

{

demo d= new demo();

d.dispaly();

}

}

**package** abstarct;

**abstract class** call {

**void** display()

{

System.***out***.println("hello");

}

}

Output: Not necessary that abstract class have abstract method and methods do not require implementation for declaration

ii. When we extend an abstract class ,we must override all the abstract method in subclass or declare sub class as abstract

package abstract;

public class demo extends call {

public void dispaly2()

{

System.out.println("HI");

}

Public static void main(String[] args)

{

demo d= new demo();

d.dispaly2();

}

}

package abstract;

abstract class call {

void display()

{

System.out.println("hello");

}

}

Output: HI

iii. Abstract class cannot be private

Exception in thread "main" java.lang.Error: Unresolved compilation problem:

Implicit super constructor call() is not visible for default constructor. Must define an explicit constructor

Iv . Abstract class cannot be final

Yes it can’t be a final because final is not a access specifier

v. you can declare abstract class without abstract method

yes we can create abstract class without a method.

5. write the classes line, rectangle, cube etc. and make the shape as their base class. Add an abstract draw() method in the class shape and draw all shapes.

abstract class Shape{

abstract void draw();

}

class Line extends Shape{

void draw(){System.out.println("drawing Line");}

}

class Square extends Shape{

void draw(){System.out.println("drawing square");}

}

class Rectangle extends Shape{

void draw(){System.out.println("drawing rectangle");}

}

class Circle extends Shape{

void draw(){System.out.println("drawing circle");}

}

class Paint{

public static void main(String args[]){

Shape s1=new Circle();

Shape s2=new Rectangle();

Shape s3=new Line();

Shape s4=new Square();

s1.draw();

s2.draw();

s3.draw();

s4.draw();

}

}

Output

drawing circle

drawing rectangle

drawing Line

drawing square

6. write an abstract class persistence along with two sub classes.

abstract class Persistence{

abstract void persist();

}

class File Persistence extends Persistence{

@Override

void persist() {

// TODO Auto-generated method stub

System.out.println("Data Saved in File");

}

}

class Database Persistence extends Persistence{

@Override

void persist() {

// TODO Auto-generated method stub

System.out.println("Data saved in Database");

}

}

public class Client extends Persistence{

public static void main(String[] args) {

// TODO Auto-generated method stub

Persistence obj=new Client();

obj.persist();

}

@Override

void persist() {

// TODO Auto-generated method stub

System.out.println("Persist method from client");

}

}

Output

Persist method from client

7. Develop an application for dessert shop. The application should allow the owner to add items like candy, cookie, ice cream in shop storage. Also customer should be able to place an order.

abstract class Dessertitem{

abstract void getCost();

}

class Candy extends Dessertitem{

@Override

void getCost() {

// TODO Auto-generated method stub

double val=0;

int orderqty, amt[]= {5,10,50};

double tax=0.3;

System.out.println("Tax on Candy dessert : "+tax);

Scanner sc = new Scanner(System.in);

System.out.println("Enter quantity to place order :");

orderqty=sc.nextInt();

val = amt[0]\*orderqty\*60\*tax;

System.out.println("Thank you for placing order!! It costs :"+val+" rupees");

}

}

class Cookie extends Dessertitem{

@Override

void getCost() {

// TODO Auto-generated method stub

double val=0;

int orderqty, amt[]= {5,10,50};

double tax=0.2;

System.out.println("Tax on Cookie dessert : "+tax);

Scanner sc = new Scanner(System.in);

System.out.println("Enter quantity to place order :");

orderqty=sc.nextInt();

val = amt[1]\*orderqty\*70\*tax;

System.out.println("Thank you for placing order!! It costs :"+val+" rupees");

}

}

class Icecream extends Dessertitem{

@Override

void getCost() {

// TODO Auto-generated method stub

double val=0;

int orderqty, amt[]= {5,10,50};

double tax=0.5;

System.out.println("Tax on Icecream dessert : "+tax);

Scanner sc = new Scanner(System.in);

System.out.println("Enter quantity to place order :");

orderqty=sc.nextInt();

val = amt[2]\*orderqty\*tax;

System.out.println("Thank you for placing order!! It costs :"+val+" rupees");

}

}

public class DessertShop extends Dessertitem{

public static void main(String[] args) {

// TODO Auto-generated method stub

String desserts[]= {"Candy","Cookie","Ice cream"};

int qty[]= {10,20,15},amt[]= {5,10,50};

Dessertitem obj=new DessertShop();

Dessertitem can=new Candy();

Dessertitem ck=new Cookie();

Dessertitem ice=new Icecream();

Scanner sc = new Scanner(System.in);

char ch;

int own;

do {

System.out.println("\n \*\*\*Welcome to Dessert Shop Application\*\*\*");

System.out.println("a. Owner \n b. Customer\n c. Exit ");

System.out.println("Enter your choice: ");

ch = sc.next().charAt(0);

switch (ch) {

case 'a':

System.out.println("\n \*\*\*Owner Module\*\*\*");

System.out.println("\n \*\*\*Dessert you want to add\*\*\*");

System.out.println("1. Candy\n2. Cookie\n3.Ice cream ");

System.out.println("Enter your choice: ");

own=sc.nextInt();

if(own==1 || own==2 || own ==3) {

System.out.println(desserts[own-1]+" Available quantity in stock : "+qty[own-1]);

System.out.println("Enter the quantity to add:");

qty[own-1]=qty[own-1]+sc.nextInt();

System.out.println("Dessert added successfully");

System.out.println("Candy : "+qty[0]+"\nCookie : "+qty[1]+"\nIce cream : "+qty[2]);

System.out.println("See you soon...Have a great day!!");

}

break;

case 'b':

System.out.println("\n \*\*\*Customer Module\*\*\*");

System.out.println("\n \*\*\*Dessert you want to place an Order for\*\*\*");

System.out.println("1. Candy\n2. Cookie\n3.Icecream ");

System.out.println("Enter your choice: ");

own=sc.nextInt();

switch(own) {

case 1:

System.out.println("\*\*Candy\*\*");

System.out.println("quantity Available: "+qty[own-1]);

System.out.println("Each Candy costs : "+amt[own-1]+" in dollars");

can.getCost();

System.out.println("See you soon...Have a great day!!");

break;

case 2:

System.out.println("\*\*Cookie\*\*");

System.out.println("quantity Available: "+qty[own-1]);

System.out.println("Each Cookie costs : "+amt[own-1]+" in Euros");

ck.getCost();

System.out.println("See you soon...Have a great day!!");

Break;

case 3:

System.out.println("\*\*Ice cream\*\*");

System.out.println("quantity Available: "+qty[own-1]);

System.out.println("Each Ice cream costs : "+amt[own-1]+" in Rupees");

ice.getCost();

System.out.println("See you soon...Have a great day!!");

break;

}

case 'c':

//System.out.println("See you soon...Have a great day!!");

break;

}

}

while (ch != 3);

}

@Override

void getCost() {

// TODO Auto-generated method stub

System.out.println("From Main class");

}

}

Output

\*\*\*Welcome to Dessert Shop Application\*\*\*

a. Owner

b. Customer

c. Exit

Enter your choice:

a

\*\*\*Owner Module\*\*\*

\*\*\*Dessert you want to add\*\*\*

1. Candy

2. Cookie

3. Icecream

Enter your choice:

1

Candy Available quantity in stock: 10

Enter the quantity to add:

5

Dessert added successfully

Candy: 15

Cookie: 20

Ice cream: 15

See you soon...Have a great day!!

\*\*\*Welcome to Dessert Shop Application\*\*\*

a. Owner

b. Customer

c. Exit

Enter your choice:

a

\*\*\*Owner Module\*\*\*

\*\*\*Dessert you want to add\*\*\*

1. Candy

2. Cookie

3. Ice cream

Enter your choice:

3

Ice cream Available quantity in stock: 15

Enter the quantity to add:

3

Dessert added successfully

Candy: 15

Cookie: 20

Ice cream: 18

See you soon...Have a great day!!

\*\*\*Welcome to Dessert Shop Application\*\*\*

a. Owner

b. Customer

c. Exit

Enter your choice:

b

\*\*\*Customer Module\*\*\*

\*\*\*Dessert you want to place an Order for\*\*\*

1. Candy

2. Cookie

3. Ice cream

Enter your choice:

2

\*\*Cookie\*\*

Quantity Available: 20

Each Cookie costs: 10 in Euros

Tax on Cookie dessert: 0.2

Enter quantity to place order:

2

Thank you for placing order!! It costs: 280.0 rupees

See you soon...Have a great day!!