**Q1. Write a singleton class. Confirm that singleton class cannot be inherited.**

**Singleton Class:-**

The singleton design pattern is used to restrict the instantiation of a class and ensures that only one instance of the class exists in the JVM. In other words, a singleton class is a class that can have only one object (an instance of the class) at a time per JVM instance.

**Theory:**

**Singleton Class in Java**

In object-oriented programming, a singleton class is a class that can have only one object (an instance of the class) at a time.

After first time, if we try to instantiate the Singleton class, the new variable also points to the first instance created. So whatever modifications we do to any variable inside the class through any instance, it affects the variable of the single instance created and is visible if we access that variable through any variable of that class type defined.

Remember the key points while defining class as singleton class that is while designing a singleton class:

* Make constructor private.
* Write a static method that has return type object of this singleton class. Here, the concept of Lazy initialization is used to write this static method.

**Program for valid accessing of Singleton Class:**

**class** SingletonInheritanceCheck{

**private** **static** SingletonInheritanceCheck *X* = **null**;

**private** SingletonInheritanceCheck(){

System.***out***.println("I am a statement inside the singleton class");

}

**public** **static** SingletonInheritanceCheck getInstance(){

**if** (*X*== **null**)

*X* = **new** SingletonInheritanceCheck();

**return** *X*;

}

}

**public** **class** Assignment2Q1 {

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

SingletonInheritanceCheck x = SingletonInheritanceCheck.*getInstance*();

// Instantiating Singleton class with variable y

SingletonInheritanceCheck y = SingletonInheritanceCheck.*getInstance*();

// Instantiating Singleton class with variable z

SingletonInheritanceCheck z = SingletonInheritanceCheck.*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(

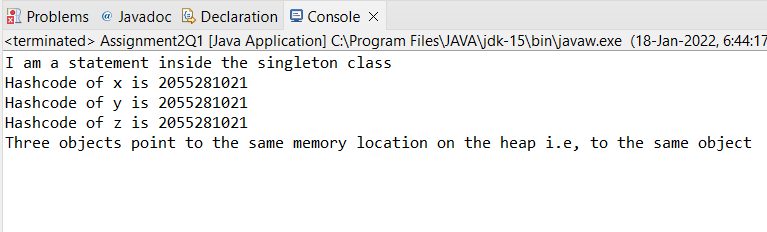
"Three objects DO NOT point to the same memory location on the heap");

}

}

}

**Output:**



**Q2. Write a program that describes the hierarchy of an organization. Here we need to write 3 classes Employee, Manager & Labour where Manager & Labour are the sub classes of the Employee. Manager has incentive & Labour has over time. Add the functionality to calculate total salary of all the employees. Use polymorphism i.e. method overriding.**

**Description:-**

Write a program to calculate the total salaries given to all the employees by an organization. In this organization there are employees category under which Manager and Labour are working where manager will receive incentive and labourer will receive overtime benefits.

**Code:**

**class** Employee{

**int** getSalary(**int** salary){

**return** salary;

}

}

**class** Manager **extends** Employee{

@Override

**int** getSalary(**int** salary){

**int** incentive = 2000;

**return** (incentive + salary);

}

}

**class** Labour **extends** Employee{

@Override

**int** getSalary(**int** salary){

**int** overtime = 500;

**return** (overtime + salary);

}

}

**public** **class** Assignment2Q2 {

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

**int** salary = 8000;

//Creating an instance for the Manager class.

Manager manager = **new** Manager();

//Creating an instance for the Labour class.

Labour labour = **new** Labour();

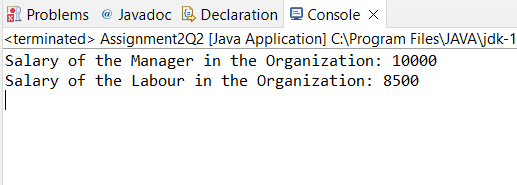
System.***out***.println("Salary of the Manager in the Organization: "+manager.getSalary(salary));

System.***out***.println("Salary of the Labour in the Organization: "+labour.getSalary(salary));

}

}

**Output:**



**Q3. Write a program to consider saving & current account in the bank. Saving account holder has ‘Fixed Deposits’ whereas Current account holder has cash credit. Apply polymorphism to find out total cash in the bank.**

**Description: -**

Write a program to calculate the total cash available in the bank. In this bank, bank will provide two types of accounts one is savings accounts and another is current account where current account has cash credits and savings account has fixed deposit options.

**Code:**

**class** Bank{

**public** **int** getCash(**int** cash){

**return** cash;

}

}

**class** SavingAccount **extends** Bank{

**int** totalDeposits = 10000;

**int** fixedDepositAmount = 5000;

@Override

**public** **int** getCash(**int** cash) {

**return** totalDeposits+cash;

}

**public** **int** getTotalDeposits() {

**return** totalDeposits;

}

**public** **int** getFixedDepositAmount() {

**return** fixedDepositAmount;

}

}

**class** CreditAccount **extends** Bank{

**int** totalDeposits = 10000;

**int** cashCredit = 2000;

@Override

**public** **int** getCash(**int** cash){

**return** totalDeposits + cash;

}

**public** **int** getTotalDeposits() {

**return** totalDeposits;

}

**public** **int** getCashCredit() {

**return** cashCredit;

}

}

**public** **class** Assignment2Q3 {

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

//Creating an instance for the credit account class.

CreditAccount credit = **new** CreditAccount();

//Creating an instance for the saving account class.

SavingAccount saving = **new** SavingAccount();

**int** cash = 10000;

System.***out***.println("Minimum cash limit in credit account: "+credit.getCashCredit());

System.***out***.println("Fixed deposit amount in saving account: "+saving.getFixedDepositAmount());

System.***out***.println("Cash Deposited in credit account: "+credit.getTotalDeposits());

System.***out***.println("Cash Deposited in saving account: "+saving.getTotalDeposits());

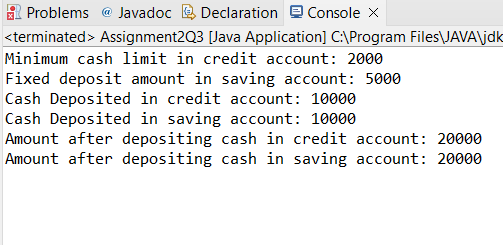
System.***out***.println("Amount after depositing cash in credit account: "+credit.getCash(cash));

System.***out***.println("Amount after depositing cash in saving account: "+saving.getCash(cash));

}

}

**Output:**



**4. Test the following principles of an abstract class:**

**• If any class has any of its method abstract then you must declare entire class abstract.**

**• Abstract class cannot be instantiated.**

**• When we extend an abstract class, we must either override all the abstract methods in sub class or declare subclass as abstract.**

**• Abstract class cannot be private.**

**• Abstract class cannot be final.**

**• You can declare a class abstract without having any abstract method.**

**Description:-**

Write a program in such a way that all the conditions above for abstract class should satisfy.

**Code:**

**abstract** **class** AbstractClass{

**public** **int** a;

**public** **int** b;

**public** AbstractClass(){

System.***out***.println("I am a constructor of Abstract class with an Abstract method.");

}

**public** **int** getA() {

**return** a;

}

**public** **void** setA(**int** a) {

**this**.a = a;

}

**public** **int** getB() {

**return** b;

}

**public** **void** setB(**int** b) {

**this**.b = b;

}

**abstract** **int** Sum();

}

**abstract** **class** Base1{

**public** Base1(){

System.***out***.println("I am a constructor of an Abstract class without any Abstract method");

}

}

**class** Derived **extends** AbstractClass{

@Override

**int** Sum() {

**return** a+b;

}

}

**class** Derived1 **extends** Base1{

}

**public** **class** Assignment2Q4 {

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

Derived derived = **new** Derived();

System.***out***.println("Enter the values of a and b: ");

derived.setA(20);

derived.setB(30);

System.***out***.println("Values of a and b are: ");

System.***out***.print("a = "+derived.getA());

System.***out***.println("b = "+derived.getB());

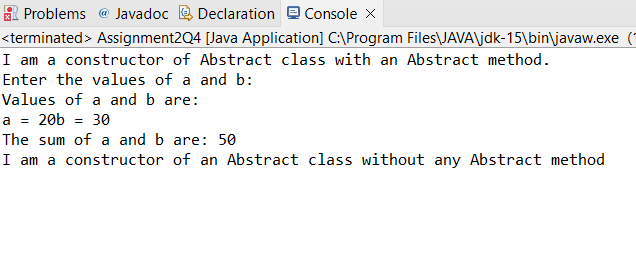
System.***out***.println("The sum of a and b are: "+derived.Sum());

Derived1 derived1 = **new** Derived1();

}

}

**Output:**

****

**Q5. Write the classes Line, Rectangle, Cube etc. & make the Shape as their base class. Add an abstract draw() method in the class Shape & draw all shapes.**

**Description:-**

Write a java class named Shape which has abstract draw() and also classes Line, Rectangle, Cube etc. extends Shape to implement the draw method.

**Code:**

**abstract** **class** Shape{

**abstract** **public** String draw();

}

**class** Line **extends** Shape{

Line(){

System.***out***.println("I am constructor of Line Class");

}

@Override

**public** String draw() {

**return** "Draw a line...";

}

}

**class** Rectangle **extends** Shape{

Rectangle(){

System.***out***.println("I am constructor of Rectangle Class");

}

@Override

**public** String draw() {

**return** "Draw a rectangle...";

}

}

**class** Cube **extends** Shape{

Cube(){

System.***out***.println("I am constructor of Cube Class");

}

@Override

**public** String draw() {

**return** "Draw a cube...";

}

}

**public** **class** Assignment2Q5 {

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

//Instance of line class has been instantiated.

Line line = **new** Line();

//Instance of rectangle class has been instantiated.

Rectangle rectangle = **new** Rectangle();

//Instance of cube class has been instantiated.

Cube cube = **new** Cube();

System.***out***.println(line.draw());

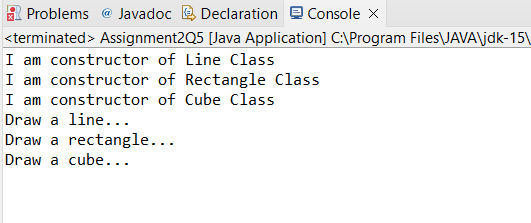
System.***out***.println(rectangle.draw());

System.***out***.println(cube.draw());

}

}

**Output:**



**Q6. Write an abstract class ‘Persistence’ along with two sub classes ‘FilePersistence’ & ‘DatabasePersistence’. The base class with have an abstract method persist() which will be overridden by its sub classes. Write a client who gets the Persistence object at runtime & invokes persist() method on it without knowing whether data is being saved in File or in Database.**

**Description:-**

Write a program having an abstract class "Persistence" which have two child classes "FilePersistence","DatabasePersistence".The base class will have a persist() method and it is overridden by sub classes.wite a seperate class of client which will get persistence object and invoke persist method on it.(Polymorphism)

**Code:**

**abstract** **class** Persistence{

**abstract** **public** String Persist();

}

**class** FilePersistence **extends** Persistence{

@Override

**public** String Persist() {

**return** "This is the Persist method of File Persistence Method.";

}

}

**class** DatabasePersistence **extends** Persistence{

@Override

**public** String Persist() {

**return** "This is the Persist method of Database Persistence Method.";

}

}

**public** **class** Assignment2Q6 {

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

//Creating an instance of File Persistence class.

FilePersistence filePersistence = **new** FilePersistence();

System.***out***.println(filePersistence.Persist());

//Creating an instance of Database Persistence class.

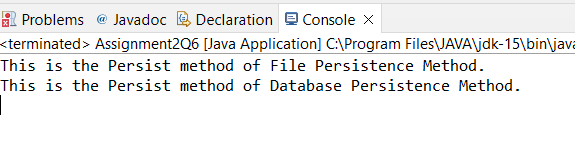
DatabasePersistence databasePersistence = **new** DatabasePersistence();

System.***out***.println(databasePersistence.Persist());

}

}

**Output:**



**Q7. Develop an application for Dessert shop. The application should allow owner to add items like Candy, Cookie or Ice Cream in the shop storage. Also, customers should be able to place an order.**

**Dessert Item is an abstract class having an abstract method getCost(). Every dessert item has tax associated. Candy item is sold in dollar currency, Cookie in Euro currency & Ice Cream in Rupees currency. The sub classes are supposed to override these methods. When we run the application, it should ask us our role i.e. owner or customer. If role is owner, we should be able to add dessert items in our storage. If role is customer, then we should be able to place an order. The currency conversion rates are:**

**1 dollar = 60 rupees.**

**1 euro = 70 rupees.**

**Code:**

**import** java.util.Scanner;

**abstract** **class** DesertItem{

**abstract** **public** **int** getCost();

}

**class** Candy **extends** DesertItem{

**public** **int** cash;

**int** addCandy;

Candy(){

addCandy = 0;

}

**public** **void** setAddCandy(**int** candy){

**this**.addCandy += candy;

}

**public** **int** getAddCandy() {

**return** addCandy;

}

**public** **void** setCash(**int** cash) {

**this**.cash = cash;

}

@Override

**public** **int** getCost() {

**return** cash\*addCandy\*60;

}

**public** **void** Order(**int** order){

addCandy -= order;

}

**public** **int** TotalAmount(**int** order){

**return** cash\*order\*60;

}

}

**class** Cookie **extends** DesertItem{

**int** addCookie;

**public** **int** cash;

**public** Cookie() {

**this**.addCookie = 0;

}

**public** **int** getAddCookie() {

**return** addCookie;

}

**public** **void** setAddCookie(**int** addCookie) {

**this**.addCookie += addCookie;

}

**public** **void** setCash(**int** cash){

**this**.cash = cash;

}

@Override

**public** **int** getCost() {

**return** cash\*addCookie\*70;

}

**public** **void** Order(**int** order){

addCookie -= order;

}

**public** **int** TotalAmount(**int** order){

**return** cash\*order\*70;

}

}

**class** IceCream **extends** DesertItem{

**int** addIceCream;

**public** **int** cash;

**public** IceCream() {

addIceCream = 0;

}

**public** **void** setCash(**int** cash) {

**this**.cash = cash;

}

**public** **int** getAddIceCream() {

**return** addIceCream;

}

**public** **void** setAddIceCream(**int** addIceCream) {

**this**.addIceCream += addIceCream;

}

@Override

**public** **int** getCost() {

**return** cash\*addIceCream;

}

**public** **void** Order(**int** order){

addIceCream -= order;

}

**public** **int** TotalAmount(**int** order){

**return** cash\*order;

}

}

**public** **class** Assignment2Q7 {

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

Scanner sc = **new** Scanner(System.***in***);

**int** choice;

Candy candy = **new** Candy();

Cookie cookie = **new** Cookie();

IceCream iceCream = **new** IceCream();

**do** {

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

System.***out***.println("\t\t\*\* SELECT ROLE \*\*\t\t");

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

System.***out***.println("\t\t\*\* 1) OWNER \*\*\t\t");

System.***out***.println("\t\t\*\* 2) CUSTOMER \*\*\t\t");

System.***out***.println("\t\t\*\* 3) EXIT \*\*\t\t");

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

System.***out***.print("\t\tENTER YOUR CHOICE: ");

choice = sc.nextInt();

**switch** (choice) {

**case** 1:

**int** desertitem;

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

System.***out***.println("\t\t\*\* ADD DESERTS \*\*\t\t");

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

System.***out***.println("\t\t\*\* 1) ADD CANDY \*\*\t\t");

System.***out***.println("\t\t\*\* 2) ADD COOKIE \*\*\t\t");

System.***out***.println("\t\t\*\* 3) ADD ICE CREAM \*\*\t\t");

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

System.***out***.print("\t\tENTER YOUR CHOICE: ");

desertitem = sc.nextInt();

**if** (desertitem == 1) {

**int** m;

**do** {

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

System.***out***.println("\t\t 1. ENTER THE NUMBER OF CANDIES ");

System.***out***.println("\t\t 2. DISPLAY THE NUMBER OF CANDIES ");

System.***out***.println("\t\t 3. ENTER THE PRICE OF CANDIES ");

System.***out***.println("\t\t 4. DISPLAY THE PRICE OF CANDIES ");

System.***out***.println("\t\t 5. GO BACK");

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

System.***out***.print("\t\tENTER YOUR CHOICE: ");

m = sc.nextInt();

**if** (m == 1) {

**int** candies;

System.***out***.print("\t\tENTER THE NUMBER OF CANDIES: ");

candies = sc.nextInt();

candy.setAddCandy(candies);

} **else** **if** (m == 2) {

System.***out***.println("\t\tNUMBER OF CANDIES = " + candy.getAddCandy());

} **else** **if** (m == 3) {

**int** price;

System.***out***.print("\t\tENTER THE PRICE OF CANDIES (in Dollars): ");

price = sc.nextInt();

candy.setCash(price);

} **else** **if**(m==4){

System.***out***.println("\t\tPRICE OF CANDIES (in Rupees) = " + candy.getCost());

}

**else**{

**break**;

}

}**while** (m!=5);

} **else** **if** (desertitem == 2) {

**int** m;

**do** {

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

System.***out***.println("\t\t 1. ENTER THE NUMBER OF COOKIES ");

System.***out***.println("\t\t 2. DISPLAY THE NUMBER OF COOKIES ");

System.***out***.println("\t\t 3. ENTER THE PRICE OF COOKIES ");

System.***out***.println("\t\t 4. DISPLAY THE PRICE OF COOKIES ");

System.***out***.println("\t\t 5. GO BACK");

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

System.***out***.print("\t\tENTER YOUR CHOICE: ");

m = sc.nextInt();

**if** (m == 1) {

**int** cookies;

System.***out***.print("\t\tENTER THE NUMBER OF COOKIES: ");

cookies = sc.nextInt();

cookie.setAddCookie(cookies);

} **else** **if** (m == 2) {

System.***out***.println("\t\tNUMBER OF COOKIES = " + cookie.getAddCookie());

} **else** **if** (m == 3) {

**int** price;

System.***out***.print("\t\tENTER THE PRICE OF COOKIES (in Euros): ");

price = sc.nextInt();

cookie.setCash(price);

} **else** **if**(m==4){

System.***out***.println("\t\tPRICE OF COOKIES (in Rupees) = " + cookie.getCost());

}

**else**{

**break**;

}

}**while** (m!=5);

} **else** {

**int** m;

**do**{

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

System.***out***.println("\t\t 1. ENTER THE NUMBER OF ICE CREAM ");

System.***out***.println("\t\t 2. DISPLAY THE NUMBER OF ICE CREAM ");

System.***out***.println("\t\t 3. ENTER THE PRICE OF ICE CREAM ");

System.***out***.println("\t\t 4. DISPLAY THE PRICE OF ICE CREAM ");

System.***out***.println("\t\t 5. GO BACK");

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

System.***out***.print("\t\tENTER YOUR CHOICE: ");

m = sc.nextInt();

**if** (m == 1) {

**int** icecreams;

System.***out***.print("\t\tENTER THE NUMBER OF ICE CREAM: ");

icecreams = sc.nextInt();

iceCream.setAddIceCream(icecreams);

} **else** **if** (m == 2) {

System.***out***.println("\t\tNUMBER OF ICE CREAM = " + iceCream.getAddIceCream());

} **else** **if** (m == 3) {

**int** price;

System.***out***.print("\t\tENTER THE PRICE OF ICE CREAM (in Rupees): ");

price = sc.nextInt();

iceCream.setCash(price);

} **else** **if**(m==4){

System.***out***.println("\t\tPRICE OF ICE CREAM (in Rupees) = " + iceCream.getCost());

}

**else**{

**break**;

}

}**while**(m!=5);

}

**break**;

**case** 2:

**int** order;

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

System.***out***.println("\t\t\*\* SELECT DESERT \*\*\t\t");

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

System.***out***.println("\t\t\*\* 1) CANDY \*\*\t\t");

System.***out***.println("\t\t\*\* 2) COOKIE \*\*\t\t");

System.***out***.println("\t\t\*\* 3) ICE CREAM \*\*\t\t");

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

System.***out***.print("\t\tENTER YOUR CHOICE: ");

order = sc.nextInt();

**if**(order == 1){

**int** qty;

System.***out***.print("\t\t ENTER THE QUANTITY YOU WANT TO PURCHASE: ");

qty = sc.nextInt();

candy.Order(qty);

System.***out***.println("\t\t ORDER RECEIPT: ");

System.***out***.println("\t\t QUANTITY ORDER = "+qty);

System.***out***.println("\t\t TOTAL AMOUNT (in Rupees) = "+candy.TotalAmount(qty));

}

**else** **if**(order==2){

**int** qty;

System.***out***.print("\t\t ENTER THE QUANTITY YOU WANT TO PURCHASE: ");

qty = sc.nextInt();

cookie.Order(qty);

System.***out***.println("\t\t ORDER RECEIPT: ");

System.***out***.println("\t\t QUANTITY ORDER = "+qty);

System.***out***.println("\t\t TOTAL AMOUNT (in Rupees) = "+cookie.TotalAmount(qty));

}

**else**{

**int** qty;

System.***out***.print("\t\t ENTER THE QUANTITY YOU WANT TO PURCHASE: ");

qty = sc.nextInt();

iceCream.Order(qty);

System.***out***.println("\t\t ORDER RECEIPT: ");

System.***out***.println("\t\t QUANTITY ORDER = "+qty);

System.***out***.println("\t\t TOTAL AMOUNT (in Rupees) = "+iceCream.TotalAmount(qty));

}

**break**;

**case** 3:

System.*exit*(0);

**break**;

**default**:

System.***out***.println("\t\t PLEASE ENTER THE CORRECT CHOICE!!!");

}

}**while** (choice != 3) ;

}

}

**Output:**

