**JAVA**

**Inheritance**

**Lab Exercise No:**34

**Exercise Objective(s):** *The concept of inheritance*

**Exercise:** *Create a class called Vehicle. Create subclasses like Truck, Bus, Car etc. Add common methods*

*in the base class and specific methods in the corresponding class. Create a class called Road*

*and create objects for the Truck, Car, Bus etc and display the appropriate message.*

**SOLUTION:**

*/\* Understanding the concept of inheritance through one parent class vehicle and Truck,Bus, Car as Subclass \*/*

package com.hsbc.pack;

class Vehicle {

public Vehicle() {

super();

System.out.println(" Vehicle");

}

}

class Car extends Vehicle { //Car is a subclass of Vehicle class

public Car() {

super(); //super calls the constructor of parent class

}

public void printDesc() {

System.out.println("Car");

}

}

class Truck extends Vehicle { //Truck is a subclass of Vehicle class

public Truck() {

super();

}

public void printDesc() {

System.out.println("Truck");

}

}

class Bus extends Vehicle { { //Bus is a subclass of Vehicle class

public Bus() {

super();

}

public void printDesc() {

System.out.println(" Bus");

}

}

public class Road {

public static void main(String[] args) {

Car c = new Car();

c.printDesc();

System.out.println();

Truck t = new Truck();

t.printDesc();

System.out.println();

Bus b = new Bus();

b.printDesc();

System.out.println();

}

}

--------------------------------------------------------------------------------------------------------------------------------------

**Lab Exercise No:**35

**Exercise Objective(s):***super keyword*

**Exercise:***In the Lab Exercise 34, in the Vehicle class constructor initialize few variables like color, no of*

*wheels, model etc. Give appropriate values for these variables from the invoking subclass.*

**SOLUTION:**

/\* Implementing inheritance and understanding the concept of super method to call the parent class methods, variables and constructors \*/

class Vehicle { //Vehicle is a parent class

String color;

int noOfWheels;

String modelName;

public Vehicle() {

super();

}

public Vehicle(String color, int noOfWheels, String modelName) { //parametrized constructor of Vehicle class

super(); //super calls parent class constructor which is Object Class

this.color = color;

this.noOfWheels = noOfWheels;

this.modelName = modelName;

}

}

class Bike extends Vehicle { //Bike is child class to Vehicle

public Bike() {

super("Blue",2,"Pulsar”, V4S"); //calling constructor of Vehicle class

}

public void details() { //displaying values

System.out.println("Bike Name : " + this.modelName + "\n" + "Bike Color : " + this.color + "\n" + "No. of Wheels : " + this.noOfWheels);

}

}

class Car extends Vehicle { //Car is child class to Vehicle

public Car() {

super("Blue",4,"Buggati Chiron"); // calling constructor of Vehicle class

}

public void details() {

System.out.println("Car Name : " + this.modelName + "\n" + "Car Color : " + this.color + "\n" + "No. of Wheels : " + this.noOfWheels);

}

}

class Truck extends Vehicle { //Truck is child class to Vehicle

public Truck() {

super("Black",2,"Tesla Cybertruck");

}

public void details() {

System.out.println("Truck Name : " + this.modelName + "\n" + "Truck Color : " + this.color + "\n" + "No. of Wheels : " + this.noOfWheels);

}

}

public class Solution35 {

public static void main(String[] args) {

System.out.println("Details of Bike");

Bike b = new Bike();

b.details();

System.out.println("Details of Car");

Car c = new Car();

c.details();

System.out.println("Details of Truck”);

Truck t = new Truck();

t.details();

}

}

--------------------------------------------------------------------------------------------------------------------------------------

**Lab Exercise No:**36

**Exercise Objective(s):** *protected access specifier*

**Exercise:***In the Lab Exercise 35, create another class called City which creates an object for the Car,*

*Truck and Bus class and displays the details through a display () method in the Vehicle class.*

*The other methods and data members should not be accessible by the City class.*

**SOLUTION:**

*/\** Demonstration of the use of access specifiers and how they affect the visibility of parent class data members in child class through inheritance \*/

class Vehicle {

protected String color ;

protected int noOfWheels;

protected String model;

public Vehicle() { //default constructor

super(); //calls the parent class constructor which is Object class

}

protected Vehicle(Truck truck, Bus bus, Car car) { //parametrized constructor of Vehicle class

truck.luggage();

bus.travel();

car.personalUse();

}

protected void display () {

}

protected String getColor() { //returns the color of vehicle

return color;

}

protected void setColor(String color) { //sets the color of vehicle

this.color = color;

}

protected int getNoOfWheels() { //returns the number of wheels in vehicle

return noOfWheels;

}

protected void setNoOfWheels(int noOfWheels) {

this.noOfWheels = noOfWheels; //returns the number of wheels in vehicle

}

protected String getModel() { //returns the model of vehicle

return model;

}

protected void setModel(String model) { //sets the model of vehicle

this.model = model;

}

protected void move() {

System.out.println("Every Vehicle Moves on the road");

}

}

class Truck extends Vehicle {

@Override

public String toString() { //prints the object in the form of a string

return "Truck [color=" + color + ", noOfWheels=" + noOfWheels + ", model=" + model + "]";

}

protected void luggage() {

System.out.println("Truck is mainly used for transport of luggage");

Truck truck = new Truck36();

truck.setColor("Black");

truck.setModel("T4");

truck.setNoOfWheels(8);

System.out.println(truck);

}

}

class Bus extends Vehicle {

@Override

public String toString() {

return "Bus [color=" + color + ", noOfWheels=" + noOfWheels + ", model=" + model + "]";

}

protected void travel() {

System.out.println("Inside travel method of Bus class");

Bus bus = new Bus();

bus.setColor("Black");

bus.setModel("TATA");

bus.setNoOfWheels(6);

System.out.println(bus);

}

}

class Car36 extends Vehicle {

@Override

public String toString() {

return "Car [color=" + color + ", noOfWheels=" + noOfWheels + ", model=" + model + "]";

}

protected void personalUse() {

System.out.println("About my car");

Car car = new Car();

car.setColor("Blue");

car.setModel("Maruti Suzuki");

car.setNoOfWheels(4);

System.out.println(car);

}

}

public class Solution36 {

public static void main(String[] args) {

Truck truck = new Truck();

Bus bus = new Bus();

Car car = new Car();

Vehicle vehicle = new Vehicle(truck, bus, car );

}

}

--------------------------------------------------------------------------------------------------------------------------------------

**Lab Exercise No:**37

**Exercise Objective(s):***Overriding*

**Exercise:** *In the Lab Exercise 30, create a super class called Animal and make all the existing classes as*

*the sub class for Animal class. Move the method isVegetarian and canClimb to the super class*

*and implement generically. Whenever necessary change the implementation of these methods*

*in the respective subclasses. Display the characteristic of each animal.*

**Recommended duration:** *20 Mins*

**Solution Guidance (if applicable):** *NA*

**Lab Exercise No:**38

**Exercise Objective(s):***final Keyword*

**Exercise:***In the Lab Exercise 37, make the Lion, Tiger, Deer, Monkey, Elephant and Giraffe classes such*

*that these classes cannot be inherited.*

**Recommended duration:** *20 Mins*

**Solution Guidance (if applicable):** *NA*

--------------------------------------------------------------------------------------------------------------------------------------

**Lab Exercise No:**39

**Exercise Objective(s):***Polymorphism*

**Exercise:***Create a class called Worker. Write classes DailyWorker and SalariedWorker that inherit from*

*Worker.Every worker has a name and a salaryrate. Write method Pay (int hours) to compute*

*the week pay of every worker. A Daily worker is paid on the basis of the number of days*

*she/he works.The salaried worker gets paid the wage for 40 hours a week no matter what the*

*actual hours are. Test this program to calculate the pay of workers.*

**SOLUTION:**

**/\*** Understanding the concept of polymorphism by changing the weekly pay method behavior depending upon employment type \*/

package com.hsbc.demo;

public class ThirtyNineSol {

public static void main(String[] args) {

// TODO Auto-generated method stub

//Taking a salaried and daily worker

DailyWorker work1 = new DailyWorker("Rohitkaran", 100, 52);

int salary = work1.Pay(work1.getNoOfHrsWorked(),work1.getSalaryRate());

System.out.println("The payment of given daily worker is : " + salary);

SalariedWorker work2 = new SalariedWorker("Swathy", 200, 35);

salary = work2.Pay(work2.getNoOfHrsWorked(),work2.getSalaryRate());

System.out.println("The payment of given salaried worker is : " + salary);

}

}

//Parent class

class Worker{

private String name;

private int salaryRate;

private int noOfHrsWorked;

public Worker() {

super();

}

public Worker(String name, int salaryRate, int noOfHrsWorked) { //parametrize constructor

super();

this.name = name;

this.salaryRate = salaryRate;

this.noOfHrsWorked = noOfHrsWorked;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getSalaryRate() {

return salaryRate;

}

public void setSalaryRate(int salaryRate) {

this.salaryRate = salaryRate;

}

public int getNoOfHrsWorked() {

return noOfHrsWorked;

}

public void setNoOfHrsWorked(int noOfHrsWorked) {

this.noOfHrsWorked = noOfHrsWorked;

}

}

//Subclass of Worker class

class DailyWorker extends Worker{

public DailyWorker(String name, int salaryRate, int noOfHrsWorked) {

super(name,salaryRate,noOfHrsWorked);

// TODO Auto-generated constructor stub

}

//Pay method of DailyWorker class

public int Pay(int noOfHrsWorked,int salaryRate) {

int salary= noOfHrsWorked \* salaryRate;

return salary;

}

}

//Subclass of Worker class

class SalariedWorker extends Worker{

public SalariedWorker(String name, int salaryRate, int noOfHrsWorked) {

super(name,salaryRate,noOfHrsWorked);

// TODO Auto-generated constructor stub

}

//Pay method of SalariedWorker class

public int Pay(int noOfHrsWorked,int salaryRate) {

int salary= 40 \* salaryRate;

return salary;

}

}

*--------------------------------------------------------------------------------------------------------------------------------------*

**Lab Exercise No:**40

**Exercise Objective(s):***Polymorphism*

**Exercise:***ModifyLab Exercise 38 by creating intermediate Super classes called WildAnimals and*

*DomesticAnimals and create corresponding subclasses for the same. Create generic*

*methods in the super class and display the details of wild animals, domestic animals in*

*general separately and also display the details of each animal separately.*

**Recommended duration:** *20 Mins*

**Solution Guidance (if applicable):** *NA*

--------------------------------------------------------------------------------------------------------------------------------------

**Lab Exercise No:**41

**Exercise Objective(s):***abstract classes*

**Exercise:** *Create a class called Shape3D with the following method signatures alone, volume () and*

*surfaceArea (). Then create subclasses like Cylinder, Sphere, and Cubeetc and implement*

*these methods.*

**SOLUTION:**

/\* implementing the concept of abstract class through Shape class and overriding its two methods surface area and volume in the child classes \*/

package com.hsbc.pack;

import java.io.\*;

abstract class Shape{ //abstract class

abstract public void surfaceArea();

abstract public void vol();

final float pi=3.14f;

}

class Sphere extends Shape{

double r;

private double area;

private double volume;

public void accept() { //takes the value of radius for area and volume calculations

System.out.println("Enter the radius of the Sphere: ");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

r=Double.parseDouble(br.readLine());

}

public void surfaceArea(){ //overridden function from abstract Shape class

area=pi\*r\*r;

}

public void vol(){ //overridden function from abstract Shape class

volume=1.3334\*pi\*r\*r\*r; // 4/3=1.3334

}

public void display(){

area=surfaceArea();

System.out.println("The area of sphere is: "+area);

Volume=Vol();

System.out.println("The volume of sphere is: "+volume);

}

}

class Cone extends Shape{

double h,r,area,volume;

public void accept(){ //accepts the value of height and radius of cone

System.out.println("Enter radius and height of the Cone: ");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

r=Double.parseDouble(br.readLine());

h=Double.parseDouble(br.readLine());

}

public void surfaceArea(){ //overridden function from abstract Shape class

double sq=h\*h+r\*r;

area=pi\*r\*(r+java.lang.Math.sqrt(sq));

}

public void vol(){ //overridden function from abstract Shape class

double d=h/3;

volume=pi\*r\*r\*d;

}

public void display(){

area=surfaceArea();

System.out.println("The area of Cone is: "+area);

volume=vol();

System.out.println("The volume of Cone is: "+volume);

}

}

class Cylinder extends Shape{

double r,h,area,volume;

public void accept(){ //accepting the radius and height of Cylinder

System.out.println("Enter radius and height of the Cylinder: ");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

r=Double.parseDouble(br.readLine());

h=Double.parseDouble(br.readLine());

}

public void surfaceArea(){ //overridden function from abstract Shape class

area=(2\*pi\*r\*h)+(2\*pi\*r\*r);

}

public void Volume(){ //overridden function from abstract Shape class

volume=pi\*r\*r\*h;

}

public void display(){

area=surfaceArea();

System.out.println("The area of Cylinder is: "+area);

Volume=vol();

System.out.println("The volume of Cylinder is: "+volume);

}

}

class Box extends Shape{

double l,b,h,area,volume;

public void accept(){

System.out.println("Enter length, breadth and height of the Box: ");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

l=Double.parseDouble(br.readLine());

b=Double.parseDouble(br.readLine());

h=Double.parseDouble(br.readLine());

}

public void surfaceArea(){ //overridden function from abstract Shape class

area=(2\*l\*b)+(2\*b\*h)+(2\*l\*h);

}

public void Volume(){ //overridden function from abstract Shape class

volume=l\*b\*h;

}

public void display(){

surfaceArea();

Volume();

System.out.println("The area of Box is: "+area);

System.out.println("The volume of Box is: "+volume);

}

}

public class AbstractDemo {

public static void main(String [] args) throws IOException{

Sphere s=new Sphere();

s.accept();

s.display();

Cone co=new Cone();

co.accept();

co.display();

Cylinder cy=new Cylinder();

cy.accept();

cy.display();

Box b=new Box();

b.accept();

b.display();

}

}

**Lab Exercise No:**42

**Exercise Objective(s):***abstract classes*

**Exercise:** *Create the classes required to store data regarding different types of courses that employees*

*Ina company can enroll for. All courses have name and course fee. Courses are also either*

*classroom delivered or delivered online. Courses could also be full time or part time. The*

*program must be menu based input which enables the course coordinator to register*

*employees for courses, list out employees registered for specific courses, deregister employees*

*from a course.*

**Recommended duration:***30Mins*

**Solution Guidance (if applicable):***The output can be like this,*

*Enter the number of the choice you want to choose:*

1. *To register for a course*
2. *To deregister for a course*
3. *To list the courses offered*
4. *To list the employees registered for a specific course.*

-------------------------------------------------------------------------------------------------------------------------------------

**JAVA**

**Interface**

**Lab Exercise No:**43

**Exercise Objective(s):** *Implementation of an interface*

**Exercise:** *Implement Lab Exercise 41using Interfaces.*

**SOLUTION:**

package com.hsbc.pack2;

/\*\*\*\*\*\*\*\*\*\*\*\*\* Implementing inheritance with Interface \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

import java.util.Scanner;

interface Shape0{

void volume(double radius,double height);

void area(double radius,double height);

}

class Cylinder0 implements Shape0{

@Override

public void volume(double radius, double height) {

// TODO Auto-generated method stub

double vol = 3.14\*radius\*radius\*height;

System.out.println("Volume of Cylinder: "+vol);

}

@Override

public void area(double radius, double height) {

// TODO Auto-generated method stub

double sarea = 2\*3.14\*radius\*(radius+height);

System.out.println("SurfaceArea of Cylinder: "+sarea);

}

}

class Sphere0 implements Shape0{

@Override

public void volume(double radius, double height) {

// TODO Auto-generated method stub

double vol = (4/3)\*3.14\*radius\*radius\*radius;

System.out.println("Volume of Sphere: "+vol);

}

@Override

public void area(double radius, double height) {

// TODO Auto-generated method stub

double sarea = 4\*3.14\*radius\*radius;

System.out.println("Volume of Sphere: "+sarea);

}

}

class Cube0 implements Shape0{

@Override

public void volume(double radius, double height) {

// TODO Auto-generated method stub

double vol = height\*height\*height;

System.out.println("Volume of Cube: "+vol);

}

@Override

public void area(double radius, double height) {

// TODO Auto-generated method stub

double sarea = 6\*height\*height;

System.out.println("Volume of cube is: " + sarea);

}

}

public class Solution43 {

public static void main(String[] args) {

// TODO Auto-generated method stub

double h, rad;

Cylinder0 c = new Cylinder0();

Scanner s = new Scanner(System.in);

System.out.print("Enter Base Radius & Vertical Height of Cylinder: ");

rad = s.nextDouble();

h = s.nextDouble();

c.volume(rad, h);

c.area(rad, h);

Sphere0 sp = new Sphere0();

System.out.print("Enter Radius of Sphere: ");

rad = s.nextDouble();

sp.volume(rad, 0);

sp.area(rad, 0);

Cube0 cu = new Cube0();

System.out.print("Enter Side of a Cube: ");

h = s.nextDouble();

cu.volume(0, h);

cu.area(0, h);

}

}

**--------------------------------------------------------------------------------------------------------------------------------------**

**Lab Exercise No:**44

**Exercise Objective(s):** *Implementing more than one interface.*

**Exercise:** *Create two interfaces namely Drawable and Fillable. Create class called Line, Circle, Square and*

*implement following methods through interface.*

*<I>Drawable ------ drawingColor(), thickness()*

*<I>Fillable ---------- fillingColor(), size()*

/\* Implementing the concept of multiple inheritance through interfaces and implementing them in different classes \*/

**SOLUTION:**

public class Main{ //Caller class which will call the interface methods

public static void main(String arr[]){

Line line = new Line(10);

line.drawingColor();

line.fillingColor();

Circle circle = new Circle(5);

circle.thickness();

circle.size();

Square square = new Square(10);

square.drawingColor();

square.size();

}

}

interface Drawable {

void drawingColor();

void thickness();

};

interface Fillable {

void fillingColor();

float size();

};

class Line implements Drawable,Fillable{ //multiple inheritance through interface implementation

int length;

public Line() {

}

public Line(int length) {

this.length = length;

}

@Override

public void drawingColor() {

System.out.println("Drawing Color on Line");

}

@Override

public void thickness() {

System.out.println("thikness of ,Line");

}

@Override

public void fillingColor() {

System.out.println("paint filled on Line");

}

@Override

public float size() {

return length;

}

}

class Circle implements Drawable,Fillable{

int radius;

public Circle() {

}

public Circle(int radius) {

this.radius = radius;

}

@Override

public void drawingColor() {

System.out.println("Drawing Color on Cricle");

}

@Override

public void thickness() {

System.out.println("thikness of Circle");

}

@Override

public void fillingColor() {

System.out.println("paint filled on Circle");

}

@Override

public float size() {

return (float) (2\*3.14\*radius);

}

}

class Square implements Drawable,Fillable{

int side;

@Override

public void drawingColor() {

System.out.println("Drawing Color on Squre");

}

public Square() {

}

public Square(int side) {

this.side = side;

}

@Override

public void thickness() {

System.out.println("thikness of Square");

}

@Override

public void fillingColor() {

System.out.println("paint filled on Square");

}

@Override

public float size() {

return side\*side;

}

}

*--------------------------------------------------------------------------------------------------------------------------------------*

**Lab Exercise No:**45

**Exercise Objective(s):***Implementation of an interface*

**Exercise:***In Lab Exercise 39, create a package called finance and within it create an interface called*

*Payable.It should define the getPayInfo() methodthat all the worker classeswillimplement.*

*Now display the details of the monthly pay of the workers.*

**Recommended duration:** *20 Mins*

**Solution Guidance (if applicable):** *NA*

**Lab Exercise No:**46

**Exercise Objective(s):***Implementation of an interface*

**Exercise:***Createa package called bank with the following Interfaces.*

*<Interface> Account*

*<Interface>DepositAcc <Interface>LoanAcc*

*<Interface> Interest*

*<Interface>CreditInterest <Interface>DebitInterest*

1. *<Interface> Account*

*Data members: Four String variables to hold the account type “Savings, Fixed,PersonalLoan,*

*HousingLoan”*

*Methods: createAcc()*

1. *<Interface>DepositAcc*

*Methods: withdraw (), deposit(),getBalance()*

1. *<Interface>LoanAcc*

*Methods: repayPrincipal (),payInterest (),payPartialPrincipal ()*

1. *<Interface>Interest*

*Data members: Four double variables to hold the interest percentage of Savings account, Fixed*

*deposit account,PersonalLoan account and HousingLoan account.*

*Methods: calcInt()*

1. *<Interface>CreditInterest*

*Methods: addMonthlyInt(),addHalfYrlyInt(),addAnnualInt()*

1. *<Interface>DebitInterest*

*Methods: deductMonthlyInt(),deductHalfYrlyInt(),deductAnnualInt()*

*Create a package called BankImpl and create the following classes in it.*

1. *SavingsAcc which implements DepositAcc and CreditInterest*
2. *FDAcc which implements DepositAcc and CreditInterest*
3. *PersonalLoanAcc which implements LoanAcc and DebitInterest*
4. *HousingLoanAcc which implements LoanAcc and DebitInterest*

*Now create a class called MyAccount and create instances of all the accounts and generate appropriate output.*

**Recommended duration:***40Mins*

**Solution Guidance (if applicable):** *NA*

**JAVA**

**Inner class**

**Lab Exercise No:**47

**Exercise Objective(s):***Inner classes*

**Exercise:***Create a class called BankAccount with deposit (), withdraw () and getBalance () methods.*

*Create an inner class called InterestAdder and implement the interest calculations and add*

*the interest to the current balance.*

**SOLUTION**

/\* Understanding the concept of inner class with BankAccount class as outer class and InterestAdder as inner class \*/

public class BankAccount {

float currentBalance;

public BankAccount(float currentBalance) {

this.currentBalance = currentBalance;

}

void deposit(){

}

void withdraw(int value){

}

float getBalance(){

return currentBalance;

}

public static void main(String array[]){

BankAccount acc = new BankAccount(1000000); //interest if calculated and added as well to the current balance

new BankAccount.InterestAdder().intCal(acc);

System.out.println("Account balance after the interest calculation: "+acc.currentBalance);

}

static class InterestAdder{ //(Current balance \* 12%)/12

void intCal(BankAccount acc){

acc.currentBalance = acc.currentBalance\*(float)(0.12)/(float)12 + acc.currentBalance;

}

}

}

*--------------------------------------------------------------------------------------------------------------------------------------*

**Lab Exercise No:**48

**Exercise Objective(s):***Local inner classes*

**Exercise:***Create a class called BankAccount with deposit (), withdraw () and getBalance () methods.*

*Create a local inner class inside the getBalance() method called InterestAdder and implement the interest calculations and add the interest to the current balance.*

**SOLUTION**

/\* Implementing the local inner class inside the method getBalance of BankAccount class \*/

package com.hsbc.pack2;

public class BankAccount {

float balance;

public void setBalance(float balance){ //this method set the balance to the balance field

this.balance = balance;

}

public float getBalance(){ //returns the updated balance after adding interest

class Inner{ //local inner class for adding interest to balance

public float interestAdder()

{

return (balance+(balance\*12/100));

}

}

Inner inn = new Inner();

this.balance = inn.interestAdder();

return balance;

}

public void deposit(float amount){ //updates the balance with the deposit amount

this.balance += amount;

}

public void withdraw(float amount){ //updates the balance with the withdrawl amount

this.balance -= amount;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

BankAccount48 obj = new BankAccount48();

obj.deposit(500);

System.out.println(obj.getBalance());

}

}

*--------------------------------------------------------------------------------------------------------------------------------------*

**Lab Exercise No:**49

**Exercise Objective(s):***Anonymous inner classes*

**Exercise:***Create a class called BankAccount with deposit (), withdraw () and getBalance () methods.*

*Create an anonymous inner class in the getBalance() method to do the interest calculations*

*and add the interest to the current balance.*

**SOLUTION**

/\* Implementing the concept of anonymous inner class inside the method getBalance of BankAccount class \*/

abstract class Calculate{

abstract int interest();

}

class BankAccount{

public int balance, interest;

void deposit(int x) { //method updates the balance with the deposit amount x

balance = balance+x;

}

void withdraw(int x) { method updates the balance with the withdrawl amount x

balance = balance-x;

}

int getBalance() {

Calculate p=new Calculate(){ //anonymous inner class inside the getBalance Method

int interest(){

return(((balance)\*12)/(100\*12)); //returns the interst amount

}

};

balance=balance+p.interest(); //updating balance with interest

return balance;

}

public static void main(String[] args)

{

BankAccount b=new BankAccount();

b.deposit(5);

System.out.println(b.getBalance());

}

}

----------------------------------------------------------------------END--------------------------------------------------------------