class Loan{

protected int tenure;

protected float interestRate;

Loan(int tenure, float interestRate){

this.tenure = tenure;

this.interestRate = interestRate;

}

}

class HomeLoan extends Loan{

HomeLoan(){

super(5,8.5f); //invoking super class constructor

}

public double calculateEMI(double principal){

double simpleInterest = (principal \* interestRate \* tenure) / 100;

double emi = (simpleInterest + principal) / tenure;

int additionalTax = 200;

return emi + additionalTax;

}

}

class ExecuteLoan{

public static void main (String[] args) {

HomeLoan loan = new HomeLoan(); //Runtime polymorphism

double hloan = loan.calculateEMI(2000000);

System.out.println("Home loan emi per year..." + hloan);

}

}

////////////////////////////////////////////////////////////////////////////

class Loan {

public double calculateEMI(double principal) {

double simpleInterest = (principal\*8.5\*5) / 100;

double emi = (simpleInterest+principal)/5;

return emi;

}

}

class HomeLoan extends Loan {

public double calculateEMI(double principal) {

int additionaltax = 200;

double emi = super.calculateEMI(principal); //calling super class method

return emi + additionaltax;

}

}

class ExecuteLoan {

public static void main(String[] args) {

Loan loan = null;

loan = new HomeLoan(); // Runtime polymorphism

double hloan = loan.calculateEMI(2000000);

System.out.println("Home loan emi per year..."+ hloan);

}

}

///////////////////////////////////////////////

public class Loan{

final int tenure = 0; //Value is made as final

double principal;

float interestRate;

String accountNumber;

}

class HomeLoan extends Loan{

HomeLoan(){

super.tenure = 10; //Value cannot be modified as the variable is final

}

}

///////////////////////////////////////////

public class Loan {

int tenure = 0;

double principal;

float interestRate;

String accountNumber;

final double calculateEMI(){ //A final method cannot be overridden by subclasses.

return 2000;

}

}

class HomeLoan extends Loan {

double calculateEMI(){ //cannot be overridden

return 2000;

}

}

////////////////////////////////////////////////

public final class Loan { // class is made as final. No other class can extend this class.

int tenure = 0;

double principal;

float interestRate;

String accountNumber;

double calculateEMI(){

return 2000;

}

}

// The below code is wrong because Loan is not extensible

class HomeLoan extends Loan

{

}

////////////////////////////////////////////////////////////////

class Demo {

final int tenure = 0;

double principal;

float interestRate;

String accountNumber;

final double calculateEMI(){

return 2000;

}

}

class Demo2 extends Demo{

// Error as final method is overriding

double calculateEMI(){

return 8000;

}

}

class FinalDemo{

public static void main(String[] args) {

Demo d = new Demo();

d.tenure = 1; //Error as tenure is final

System.out.println(d.tenure);

System.out.println(d.calculateEMI());

}

}

////////////////////////////////////////////////

final class Demo {

int tenure = 0;

double principal;

float interestRate;

String accountNumber;

double calculateEMI(){

return 2000;

}

}

class Dummy extends Demo{

// Error as class is final

double calculateEMI(){

return 8000;

}

}

class FinalDemo{

public static void main(String[] args) {

Demo d = new Demo();

System.out.println(d.tenure);

System.out.println(d.calculateEMI());

}

}

//////////////////////////////////////////////////

class Employee{

private String employeeId;

Employee(String employeeId){

this.employeeId=employeeId;

}

public int reward(double...fixedDeposit){ //Variable argument

double sum=0;

int rewardPoint=0;

for(double deposit:fixedDeposit){

sum=sum+deposit;

}

if(sum>1000000){

rewardPoint=20000;

}

else if(sum<1000000 && sum>=500000){

rewardPoint=10000;

}

else{

rewardPoint = 20000;

}

return rewardPoint;

}

public String getEmployeeId(){

return employeeId;

}

}

class Execute{

public static void main(String[] args){

Employee employee1=new Employee("E1001");

int rewardPoint=employee1.reward(100000,200000,300000);

Employee employee2=new Employee("E1002");

int rewardPoint1=employee2.reward(100000,100000);

System.out.println(employee1.getEmployeeId() +" has got a reward of "+rewardPoint);

System.out.println(employee2.getEmployeeId() +" has got a reward of "+rewardPoint1);

}

}

////////////////////////////////////////////////////

class Employee{

private String employeeId;

Employee(String employeeId){

this.employeeId=employeeId;

}

public int reward(double...fixedDeposit){ //Variable argument

double sum=0;

int rewardPoint=0;

for(double deposit:fixedDeposit){

sum=sum+deposit;

}

if(sum>1000000){

rewardPoint=20000;

}

else if(sum<1000000 && sum>=500000){

rewardPoint=10000;

}

else {

rewardPoint;

}

return rewardPoint;

}

public String getEmployeeId(){

return employeeId;

}

}

public class Execute{

public static void main(String[] args){

Employee employee1=new Employee("E1001");

int rewardPoint=employee1.reward(100000,200000,300000);

Employee employee2=new Employee("E1002");

int rewardPoint1=employee2.reward(100000,100000);

System.out.println(employee1.getEmployeeId() +"has got a reward of "+rewardPoint);

System.out.println(employee2.getEmployeeId() +"has got a reward of "+rewardPoint1);

}

}

//////////////////////////////////////////////////////////

enum Designation{

CEO(2),GeneralManager(4),RegionalManager(20),BranchManager(30);

private int noofEmployees;

Designation(int noofEmployees)

{

this.noofEmployees=noofEmployees;

}

public int getNoofEmployees(){

return noofEmployees;

}

}

class Bank{

public void reportees(Designation designation){

System.out.println(designation.getNoofEmployees());

}

public static void main(String[] args){

Bank bank=new Bank();

bank.reportees(Designation.CEO);

}

}

///////////////////////////////////////////////////

class EventRegistration {

private String name;

private String nameOfEvent;

private double registrationFee;

public EventRegistration(String name, String nameOfEvent) {

this.name = name;

this.nameOfEvent = nameOfEvent;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getNameOfEvent() {

return nameOfEvent;

}

public void setNameOfEvent(String nameOfEvent) {

this.nameOfEvent = nameOfEvent;

}

public double getRegistrationFee() {

return registrationFee;

}

public void setRegistrationFee(double registrationFee) {

this.registrationFee = registrationFee;

}

public void registerEvent() {

System.out.println("Please choose a valid event");

}

}

class SingleEventRegistration extends EventRegistration {

private int participantNo;

public SingleEventRegistration(String name, String nameOfEvent, int participantNo) {

super(name, nameOfEvent);

this.participantNo = participantNo;

}

public int getParticipantNo() {

return participantNo;

}

public void setParticipantNo(int participantNo) {

this.participantNo = participantNo;

}

@Override

public void registerEvent() {

double baseFee = 0;

switch (getNameOfEvent()) {

case "ShakeALeg":

baseFee = 100;

break;

case "Sing&Win":

baseFee = 150;

break;

case "PlayAway":

baseFee = 130;

break;

}

setRegistrationFee(baseFee);

System.out.println("Thank You " + getName() + " for your participation. Your registration fee is: " + getRegistrationFee());

System.out.println("You are participant no: " + getParticipantNo());

}

}

class TeamEventRegistration extends EventRegistration {

private int noOfParticipants;

private int teamNo;

public TeamEventRegistration(String name, String nameOfEvent, int noOfParticipants, int teamNo) {

super(name, nameOfEvent);

this.noOfParticipants = noOfParticipants;

this.teamNo = teamNo;

}

public int getNoOfParticipants() {

return noOfParticipants;

}

public void setNoOfParticipants(int noOfParticipants) {

this.noOfParticipants = noOfParticipants;

}

public int getTeamNo() {

return teamNo;

}

public void setTeamNo(int teamNo) {

this.teamNo = teamNo;

}

@Override

public void registerEvent() {

double baseFee = 0;

switch (getNameOfEvent()) {

case "ShakeALeg":

baseFee = 50;

break;

case "Sing&Win":

baseFee = 60;

break;

case "Actathon":

baseFee = 80;

break;

case "PlayAway":

baseFee = 100;

break;

}

setRegistrationFee(baseFee \* noOfParticipants);

System.out.println("Thank You " + getName() + " for your participation. Your registration fee is: " + getRegistrationFee());

System.out.println("You are participant no: " + getTeamNo());

}

}

public class ShowYourTalentRegistration {

public static void main(String[] args) {

SingleEventRegistration participant1 = new SingleEventRegistration("Jenny", "Sing&Win", 1);

participant1.registerEvent();

TeamEventRegistration team1 = new TeamEventRegistration("Aura", "ShakeALeg", 5, 1);

team1.registerEvent();

SingleEventRegistration participant2 = new SingleEventRegistration("Hudson", "PlayAway", 2);

participant2.registerEvent();

}

}

//////////////////////////////////////////////////////////////////