**1. Write a Java program to create a class known as "BankAccount" with methods called deposit() and withdraw(). Create a subclass called SavingsAccount that overrides the withdraw() method to prevent withdrawals if the account balance falls below one hundred.**

class BankAccount {

private double balance;

public BankAccount(double initialBalance) {

balance = initialBalance;

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

System.out.println("Deposited: " + amount);

} else {

System.out.println("Deposit amount must be positive.");

}

}

public void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

System.out.println("Withdrew: " + amount);

} else {

System.out.println("Insufficient balance or invalid amount.");

}

}

public double getBalance() {

return balance;

}

}

class SavingsAccount extends BankAccount {

public SavingsAccount(double initialBalance) {

super(initialBalance);

}

public void withdraw(double amount) {

if (getBalance() - amount < 100) {

System.out.println("Withdrawal denied: Balance cannot fall below 100.");

} else {

super.withdraw(amount);

}

}

}

public class Main {

public static void main(String[] args) {

BankAccount myAccount = new BankAccount(500);

myAccount.deposit(200);

myAccount.withdraw(150);

System.out.println("Current Balance: " + myAccount.getBalance());

SavingsAccount mySavingsAccount = new SavingsAccount(500);

mySavingsAccount.deposit(200);

mySavingsAccount.withdraw(150);

mySavingsAccount.withdraw(450);

System.out.println("Current Balance: " + mySavingsAccount.getBalance());

}

}

**Output**: Deposited: 200.0

Withdrew: 150.0

Current Balance: 550.0

Deposited: 200.0

Withdrew: 150.0

Withdrawal denied: Balance cannot fall below 100.

Current Balance: 550.0

**2. Write a Java program that creates a class hierarchy for employees of a company. The base class should be Employee, with subclasses Manager, Developer, and Programmer. Each subclass should have properties such as name, address, salary, and job title. Implement methods for calculating bonuses, generating performance reports, and managing projects.**

class Employee {

String name;

String address;

double salary;

String jobTitle;

Employee(String employeeName, String employeeAddress, double employeeSalary, String employeeJobTitle) {

name = employeeName;

address = employeeAddress;

salary = employeeSalary;

jobTitle = employeeJobTitle;

}

void calculateBonus() {

System.out.println(jobTitle + " " + name + " receives a bonus of: " + (salary \* 0.10));

}

void generatePerformanceReport() {

System.out.println("Generating performance report for " + jobTitle + " " + name);

}

void displayDetails() {

System.out.println("Name: " + name);

System.out.println("Address: " + address);

System.out.println("Salary: " + salary);

System.out.println("Job Title: " + jobTitle);

}

}

class Manager extends Employee {

int numberOfReports;

Manager(String employeeName, String employeeAddress, double employeeSalary, String employeeJobTitle, int numberOfReports) {

super(employeeName, employeeAddress, employeeSalary, employeeJobTitle);

this.numberOfReports = numberOfReports;

}

void calculateBonus() {

double bonus = salary \* 0.15 + (numberOfReports \* 50);

System.out.println(jobTitle + " " + name + " receives a bonus of: " + bonus);

}

void manageProjects() {

System.out.println(jobTitle + " " + name + " is managing projects.");

}

}

class Developer extends Employee {

String programmingLanguage;

Developer(String employeeName, String employeeAddress, double employeeSalary, String employeeJobTitle, String employeeProgrammingLanguage) {

super(employeeName, employeeAddress, employeeSalary, employeeJobTitle);

programmingLanguage = employeeProgrammingLanguage;

}

void calculateBonus() {

double bonus = salary \* 0.12;

System.out.println(jobTitle + " " + name + " receives a bonus of: " + bonus);

}

void developSoftware() {

System.out.println(jobTitle + " " + name + " is developing software using " + programmingLanguage + ".");

}

}

class Programmer extends Developer {

String project;

Programmer(String employeeName, String employeeAddress, double employeeSalary, String employeeJobTitle, String employeeProgrammingLanguage, String employeeProject) {

super(employeeName, employeeAddress, employeeSalary, employeeJobTitle, employeeProgrammingLanguage);

project = employeeProject;

}

void calculateBonus() {

double bonus = salary \* 0.10 + 200;

System.out.println(jobTitle + " " + name + " receives a bonus of: " + bonus);

}

void writeCode() {

System.out.println(jobTitle + " " + name + " is writing code for the " + project + " project.");

}

}

class Company {

public static void main(String[] args) {

Manager manager = new Manager("ABC", "123 Main St", 95000, "Manager", 5);

Developer developer = new Developer("XYZ", "456 Oak St", 80000, "Developer", "Java");

Programmer programmer = new Programmer("PQR", "789 Pine St", 70000, "Programmer", "Python", "Website Redesign");

manager.displayDetails();

manager.calculateBonus();

manager.generatePerformanceReport();

manager.manageProjects();

System.out.println();

developer.displayDetails();

developer.calculateBonus();

developer.generatePerformanceReport();

developer.developSoftware();

System.out.println();

programmer.displayDetails();

programmer.calculateBonus();

programmer.generatePerformanceReport();

programmer.writeCode();

}

}

**Output:** Name: ABC

Address: 123 Main St

Salary: 95000.0

Job Title: Manager

Manager ABC receives a bonus of: 850.0

Generating performance report for Manager ABC

Manager ABC is managing projects.

Name: XYZ

Address: 456 Oak St

Salary: 80000.0

Job Title: Developer

Developer XYZ receives a bonus of: 9600.0

Generating performance report for Developer XYZ

Developer XYZ is developing software using Java.

Name: PQR

Address: 789 Pine St

Salary: 70000.0

Job Title: Programmer

Programmer PQR receives a bonus of: 2700.0

Generating performance report for Programmer PQR

Programmer PQR is writing code for the Website Redesign project.

**3.Implement Following: a. Create abstract class shapes with dim1, dim2 variables and abstract area() method. Class b. rectangle and triangle inherits shape class. Calculate area of rectangle and triangle.**

abstract class Shape {

double dim1;

double dim2;

Shape(double d1, double d2) {

dim1 = d1;

dim2 = d2;

}

abstract double area();

}

class Rectangle extends Shape {

Rectangle(double length, double width) {

super(length, width);

}

double area() {

return dim1 \* dim2;

}

}

class Triangle extends Shape {

Triangle(double base, double height) {

super(base, height);

}

double area() {

return 0.5 \* dim1 \* dim2;

}

}

public class Main {

public static void main(String[] args) {

Shape rectangle = new Rectangle(10, 5);

Shape triangle = new Triangle(6, 8);

System.out.println("Area of Rectangle: " + rectangle.area());

System.out.println("Area of Triangle: " + triangle.area());

}

}

**Output:** Area of Rectangle: 50.0

Area of Triangle: 24.0

**4. Write a program to perform Multilevel Inheritance**

class Student {

int rollNumber;

void setRollNumber(int r) {

rollNumber = r;

}

void displayRollNumber() {

System.out.println("Roll Number: " + rollNumber);

}

}

class Test extends Student {

int sub1, sub2;

void setMarks(int s1, int s2) {

sub1 = s1;

sub2 = s2;

}

void displayMarks() {

System.out.println("Subject 1 Marks: " + sub1);

System.out.println("Subject 2 Marks: " + sub2);

}

}

class Result extends Test {

void displayResult() {

int total = sub1 + sub2;

System.out.println("Total Marks: " + total);

}

}

public class MultilevelInheritance {

public static void main(String[] args) {

Result student = new Result();

student.setRollNumber(101);

student.setMarks(85, 90);

student.displayRollNumber();

student.displayMarks();

student.displayResult();

}

}

**Output:** Roll Number: 101

Subject 1 Marks: 85

Subject 2 Marks: 90

Total Marks: 175