Assignment 3

Q1.

class Bank {

String name;

String accno;

double p;

Bank(String name, String accno, double p) {

this.name = name;

this.accno = accno;

this.p = p;

}

void display() {

System.out.println("Customer Details:");

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

System.out.println("Account Number: " + accno);

System.out.println("Principal Amount: " + p);

}

}

class Account extends Bank {

double amt;

Account(String name, String accno, double p, double amt) {

super(name, accno, p);

this.amt = amt;

}

void deposit() {

p += amt;

System.out.println("Deposit Successful. Updated Principal Amount: " + p);

}

void withdraw() {

if (amt > p) {

System.out.println("INSUFFICIENT BALANCE");

} else {

p -= amt;

if (p < 500) {

double penalty = (500 - p) / 10;

p -= penalty;

System.out.println("Withdrawal Successful with Penalty. Updated Principal Amount: " + p);

} else {

System.out.println("Withdrawal Successful. Updated Principal Amount: " + p);

}

}

}

void display() {

super.display();

System.out.println("Transaction Amount: " + amt);

}

}

public class Main {

public static void main(String[] args) {

Account customerAccount = new Account("John Doe", "123456", 1000.0, 200.0);

customerAccount.display();

customerAccount.deposit();

customerAccount.display();

customerAccount.withdraw();

customerAccount.display();

}

}

Q2.

class Number {

int n;

Number(int nn) {

n = nn;

}

int factorial(int a) {

if (a == 0 || a == 1) {

return 1;

} else {

return a \* factorial(a - 1);

}

}

void display() {

System.out.println("Value of n: " + n);

}

}

class Series extends Number {

int sum;

Series(int nn) {

super(nn);

sum = 0;

}

// Method to calculate the sum of the series S = 1! + 2! + 3! + ... + n!

void calsum() {

for (int i = 1; i <= n; i++) {

sum += factorial(i);

}

}

void display() {

super.display();

System.out.println("Sum of the series: " + sum);

}

}

public class Main {

public static void main(String[] args) {

// Creating a Series object

Series seriesObj = new Series(5);

seriesObj.calsum();

seriesObj.display();

}

}

Q3.

class Product {

String name;

int code;

double amount;

Product(String n, int c, double p) {

name = n;

code = c;

amount = p;

}

void show() {

System.out.println("Product Details:");

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

System.out.println("Product Code: " + code);

System.out.println("Sale Amount: " + amount);

}

}

class Sales extends Product {

int day;

double tax;

double totamt;

Sales(String n, int c, double p, int day) {

super(n, c, p);

this.day = day;

this.tax = 0.0;

this.totamt = 0.0;

}

void compute() {

tax = 0.124 \* amount;

double fine = (day > 30) ? 0.025 \* amount : 0;

totamt = amount + tax + fine;

}

void show() {

super.show();

System.out.println("Days to Pay: " + day);

System.out.println("Service Tax: " + tax);

System.out.println("Total Amount Paid: " + totamt);

}

}

public class Main {

public static void main(String[] args) {

Sales salesObj = new Sales("Product1", 101, 500.0, 35);

salesObj.compute();

salesObj.show();

}

}

Q4.

class Worker {

String name;

double basic;

Worker(String name, double basic) {

this.name = name;

this.basic = basic;

}

void display() {

System.out.println("Worker Details:");

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

System.out.println("Basic Pay: " + basic);

}

}

class Wages extends Worker {

double hrs;

double rate;

double wage;

Wages(String name, double basic, double hrs, double rate) {

super(name, basic);

this.hrs = hrs;

this.rate = rate;

this.wage = 0.0;

}

double overtime() {

if (hrs > 40) {

return (hrs - 40) \* rate;

} else {

return 0.0;

}

}

void display() {

super.display();

double overtimeAmount = overtime();

wage = basic + overtimeAmount;

System.out.println("Hours Worked: " + hrs);

System.out.println("Overtime Amount: " + overtimeAmount);

System.out.println("Total Wage: " + wage);

}

}

public class Main {

public static void main(String[] args) {

Wages wagesObj = new Wages("John Doe", 1000.0, 45.0, 20.0);

wagesObj.display();

}

}

Q5.

class Stock {

String item;

int qt;

double rate;

double amt;

// Parameterized constructor to assign values to the data members

Stock(String item, int qt, double rate) {

this.item = item;

this.qt = qt;

this.rate = rate;

this.amt = qt \* rate; // Calculate the net value of the item in stock

}

// Method to display the stock details

void display() {

System.out.println("Stock Details:");

System.out.println("Item: " + item);

System.out.println("Quantity: " + qt);

System.out.println("Unit Price: " + rate);

System.out.println("Net Value: " + amt);

}

}

class Purchase extends Stock {

int pqty;

double prate;

// Parameterized constructor to assign values to the data members of both classes

Purchase(String item, int qt, double rate, int pqty, double prate) {

super(item, qt, rate);

this.pqty = pqty;

this.prate = prate;

}

// Method to update stock and display stock details before and after updating

void update() {

System.out.println("Stock Details Before Update:");

super.display(); // Display stock details before update

// Update stock by adding the previous quantity with the purchased quantity

// Replace the rate if there is a difference in the purchase rate

if (prate != rate) {

rate = prate;

}

qt += pqty;

amt = qt \* rate; // Update the current stock value

System.out.println("\nStock Details After Update:");

super.display(); // Display stock details after update

}

// Method to display stock details before and after updating

void display() {

System.out.println("\nStock Details After Purchase:");

super.display(); // Display stock details before update

}

}

// Example usage

public class Main {

public static void main(String[] args) {

// Creating a Purchase object

Purchase purchaseObj = new Purchase("Item1", 50, 10.0, 20, 12.0);

// Updating stock and displaying details before and after updating

purchaseObj.update();

purchaseObj.display();

}

}