Question 1 :

import java.util.Scanner;

public class Employee {

int employeeId;

String employeeName;

String designation;

public Employee(Scanner sc) {

// Data Validation for employeeId

do {

System.out.println("Enter the employeeID (positive integer):");

while (!sc.hasNextInt()) {

System.out.println("Invalid input! Please enter a positive integer for employeeID:");

sc.next(); // consume the invalid input

}

employeeId = sc.nextInt();

} while (employeeId <= 0);

sc.nextLine(); // Consume the newline character left by nextInt()

// Data Validation for employeeName

do {

System.out.println("Enter the name of the employee:");

employeeName = sc.nextLine().trim();

} while (employeeName.isEmpty());

// Data Validation for designation

do {

System.out.println("Enter the designation of the employee:");

designation = sc.nextLine().trim();

} while (designation.isEmpty());

}

void displayDetails() {

System.out.println("Employee ID: " + employeeId);

System.out.println("Employee Name: " + employeeName);

System.out.println("Designation: " + designation);

}

public double calculateBonus() {

return 0;

}

public static class HourlyEmployee extends Employee {

double hourlyRate;

int hoursWorked;

public HourlyEmployee(Scanner sc) {

super(sc);

// Data Validation for hourlyRate

do {

System.out.println("Enter the hourlyRate (positive value):");

while (!sc.hasNextDouble()) {

System.out.println("Invalid input! Please enter a positive value for hourlyRate:");

sc.next(); // consume the invalid input

}

hourlyRate = sc.nextDouble();

} while (hourlyRate <= 0);

// Data Validation for hoursWorked

do {

System.out.println("Enter the number of hours worked (0-168):");

while (!sc.hasNextInt()) {

System.out.println("Invalid input! Please enter a valid integer for hoursWorked:");

sc.next(); // consume the invalid input

}

hoursWorked = sc.nextInt();

} while (hoursWorked < 0 || hoursWorked > 168);

}

@Override

void displayDetails() {

super.displayDetails();

System.out.println("Hourly Rate: " + hourlyRate);

System.out.println("Number of Hours Worked: " + hoursWorked);

}

public double calculateBonus() {

return hourlyRate \* hoursWorked \* 0.10;

}

double calculateWeekSalary() {

return hourlyRate \* hoursWorked;

}

}

public static class SalariedEmployee extends Employee {

double monthlySalary;

public SalariedEmployee(Scanner sc) {

super(sc);

// Data Validation for monthlySalary

do {

System.out.println("Enter the Monthly Salary (positive value):");

while (!sc.hasNextDouble()) {

System.out.println("Invalid input! Please enter a positive value for Monthly Salary:");

sc.next(); // consume the invalid input

}

monthlySalary = sc.nextDouble();

} while (monthlySalary <= 0);

}

@Override

void displayDetails() {

super.displayDetails();

System.out.println("Monthly Salary:" + monthlySalary);

}

public double calculateBonus() {

return monthlySalary / 4;

}

public static class ExecutiveEmployee extends SalariedEmployee {

double bonusPercentage;

public ExecutiveEmployee(Scanner sc) {

super(sc);

// Data Validation for bonusPercentage

do {

System.out.println("Enter Bonus Percentage (0-100):");

while (!sc.hasNextDouble()) {

System.out.println("Invalid input! Please enter a valid percentage for Bonus Percentage:");

sc.next(); // consume the invalid input

}

bonusPercentage = sc.nextDouble();

} while (bonusPercentage < 0 || bonusPercentage > 100);

}

@Override

public double calculateBonus() {

return super.calculateBonus() + (monthlySalary \* (bonusPercentage / 100));

}

public double calculateMonthlySalary() {

return super.calculateBonus(); // Use the overridden method from SalariedEmployee

}

}

public double calculateMonthlySalary() {

return super.calculateBonus(); // Use the overridden method from Employee

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

HourlyEmployee h = new HourlyEmployee(sc);

h.displayDetails();

System.out.println("Weekly Salary: $" + h.calculateWeekSalary());

System.out.println("Bonus: $" + h.calculateBonus());

SalariedEmployee sE = new SalariedEmployee(sc);

sE.displayDetails();

System.out.println("Weekly Salary: $" + sE.calculateMonthlySalary());

System.out.println("Bonus: $" + sE.calculateBonus());

SalariedEmployee.ExecutiveEmployee executiveEmployee = new SalariedEmployee.ExecutiveEmployee(sc);

executiveEmployee.displayDetails();

System.out.println("Weekly Salary: $" + executiveEmployee.calculateMonthlySalary());

System.out.println("Bonus: $" + executiveEmployee.calculateBonus());

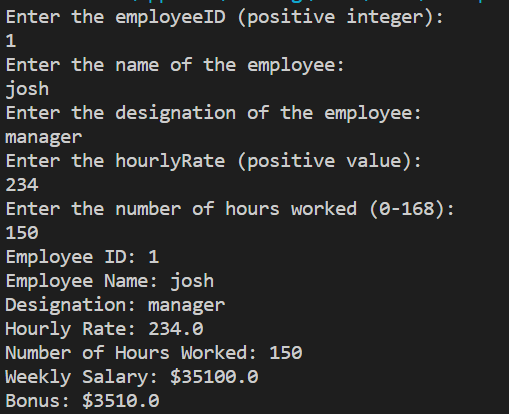
// Close the Scanner after usage

sc.close();

}

}

output



@override is used is to return the subtype of the return by the overridden. The annotation (overide)indicates that the method in a sub class is to override a method with its superclassor parent class which help in preventing the common mistakes.

Quetion 2