Problem definition on Inheritance

You are required to create a Java program for an employee management system using inheritance. The system should allow you to create and manage different types of employees such as **Manager**, **Engineer**, and **Salesperson**.

Requirements

Here are the requirements for the employee management system:

Employee class

The **Employee** class should contain the following fields:

- **name** - the name of the employee
- **employeeID** - the employee ID of the employee
- **salary** - the salary of the employee

The **Employee** class should also contain the following methods:

- **calculatePay()** - a method that calculates the pay for an employee based on their salary and any additional bonuses.
- **displayEmployeeDetails()** - a method that displays the details of an employee including their name, employee ID, salary, and any additional information depending on their type.
- **updateEmployeeDetails()** - a method that allows the user to update the details of an employee such as their name, employee ID, and salary.

Manager class

The **Manager** class should inherit from the **Employee** class and contain the following additional fields:

- **numEmployeesManaged** - the number of employees managed by the manager
- **bonusAmount** - the bonus amount for the manager

The **Manager** class should also contain the following methods:

- **calculatePay()** - a method that calculates the pay for a manager based on their salary and bonus amount.
- **displayEmployeeDetails()** - a method that displays the details of a manager including their name, employee ID, salary, number of employees managed, and bonus amount.
- **updateEmployeeDetails()** - a method that allows the user to update the details of a manager such as their name, employee ID, salary, number of employees managed, and bonus amount.

Engineer class

The **Engineer** class should inherit from the **Employee** class and contain the following additional fields:

- **fieldOfEngineering** - the field of engineering that the engineer specializes in
- **numProjectsCompleted** - the number of projects completed by the engineer

The **Engineer** class should also contain the following methods:

- **displayEmployeeDetails()** - a method that displays the details of an engineer including their name, employee ID, salary, field of engineering, and number of projects completed.
- **updateEmployeeDetails()** - a method that allows the user to update the details of an engineer such as their name, employee ID, salary, field of engineering, and number of projects completed.

Salesperson class

The **Salesperson** class should inherit from the **Employee** class and contain the following additional fields:

- **salesQuota** - the sales quota for the salesperson
- **numSales** - the number of sales made by the salesperson

The **Salesperson** class should also contain the following methods:

- **calculatePay()** - a method that calculates the pay for a salesperson based on their salary and any commission earned.
- **displayEmployeeDetails()** - a method that displays the details of a salesperson including their name, employee ID, salary, sales quota, and number of sales.
- **updateEmployeeDetails()** - a method that allows the user to update the details of a salesperson such as their name, employee ID, salary, sales quota, and number of sales.

Bonus

Here are some bonus requirements that you can implement if you want to:

- Add a method to calculate the total number of employees in the company.
- Add a method to calculate the total salary of all employees in the company.

Conclusion

In conclusion, this assignment required the creation of a Java program for an employee management system using inheritance. The program includes four classes - **Employee**, **Manager**, **Engineer**, and **Salesperson**.

The **Employee** class contains the basic information about an employee, such as their name, employee ID, and salary, as well as methods to display and update this information.

The **Manager** class is a subclass of **Employee** and adds the fields of the number of employees managed and the bonus amount, along with appropriate methods.

The **Engineer** class is also a subclass of **Employee** and includes the engineer's field of engineering and number of projects completed, along with appropriate methods.

The **Salesperson** class is also a subclass of **Employee** and includes the salesperson's sales quota and number of sales, along with appropriate methods.

The implementation of these classes provides a comprehensive solution for managing employee data, including creating, displaying, and updating employee details.

Overall, the use of inheritance simplifies the process of creating and managing employees, as the shared functionality is implemented in the base **Employee** class, and the specific functionality is implemented in the derived classes.

Assignment Description:

You are required to create a Java program that implements an employee management system using inheritance. You are given an **Employee** class which contains basic information about an employee such as their name, employee ID, and salary.

You will create three types of employees:

1. **Manager** class which extends the **Employee** class and contains additional information about the manager such as the number of employees they manage and their bonus amount.

2. **Engineer** class which extends the **Employee** class and contains additional information about the engineer such as their field of engineering and the number of projects they have completed.
3. **Salesperson** class which extends the **Employee** class and contains additional information about the salesperson such as their sales quota and the number of sales they have made.

You should create the following methods:

1. **calculatePay()** - This method should calculate the pay for an employee based on their salary and any additional bonuses.
2. **displayEmployeeDetails()** - This method should display the details of an employee including their name, employee ID, salary, and any additional information depending on their type.
3. **updateEmployeeDetails()** - This method should allow the user to update the details of an employee such as their name, employee ID, and salary.

Requirements:

1. You should create a separate class for each type of employee.
2. You should use inheritance to ensure that the **Manager**, **Engineer**, and **Salesperson** classes inherit from the **Employee** class.
3. You should use appropriate access modifiers to ensure that the employee details can only be accessed and modified within the relevant classes.
4. You should test your program by creating several employees of each type and displaying their details.

Code template:

```
// Employee class
public class Employee {
   // Fields
   private String name;
   private int employeeID;
   private double salary;

   // Constructors
   public Employee(String name, int employeeID, double salary) {
      // Implement the constructor
   }

   // Getters and setters
   public String getName() {
      // Implement the getter
   }

   public void setName(String name) {
      // Implement the setter
   }

   public int getEmployeeID() {
      // Implement the getter
```

```
    }

    public void setEmployeeID(int employeeID) {
        // Implement the setter
    }

    public double getSalary() {
        // Implement the getter
    }

    public void setSalary(double salary) {
        // Implement the setter
    }

    // Methods
    public double calculatePay() {
        // Implement the method
    }

    public void displayEmployeeDetails() {
        // Implement the method
    }

    public void updateEmployeeDetails() {
        // Implement the method
    }
}

// Manager class
public class Manager extends Employee {
    // Fields
    private int numEmployeesManaged;
    private double bonusAmount;

    // Constructors
    public Manager(String name, int employeeID, double salary, int numEmployeesManaged,
double bonusAmount) {
        // Implement the constructor
    }

    // Getters and setters
    public int getNumEmployeesManaged() {
        // Implement the getter
    }

    public void setNumEmployeesManaged(int numEmployeesManaged) {
        // Implement the setter
    }

    public double getBonusAmount() {
        // Implement the getter
    }

    public void setBonusAmount(double bonusAmount) {
        // Implement the setter
    }

    // Methods
   public double calculatePay() {
        // Implement the method
    }

    public void displayEmployeeDetails() {
        // Implement the method
    }

    public void updateEmployeeDetails(String name, int employeeID, double salary, int
numEmployeesManaged, double bonusAmount) {
        // Implement the method
    }

// Engineer class
public class Engineer extends Employee {
    // Fields
    private String fieldOfEngineering;
    private int numProjectsCompleted;

    // Constructors
    public Engineer(String name, int employeeID, double salary, String fieldOfEngineering, int
numProjectsCompleted) {
        // Implement the constructor
    }

    // Getters and setters
    public String getFieldOfEngineering() {
        // Implement the getter
    }

    public void setFieldOfEngineering(String fieldOfEngineering) {
                // Implement the setter
    }

    public int getNumProjectsCompleted() {
        // Implement the getter
    }

    public void setNumProjectsCompleted(int numProjectsCompleted) {
        // Implement the setter
    }

    // Methods
    public void displayEmployeeDetails() {
        // Implement the method
    }

    public void updateEmployeeDetails(String name, int employeeID, double salary, String
fieldOfEngineering, int numProjectsCompleted) {
        // Implement the method
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating an instance of Employee class
        Employee employee = new Employee("John Smith", 123, 50000.00);

        // Displaying the details of the Employee
        employee.displayEmployeeDetails();

        // Updating the details of the Employee
        employee.updateEmployeeDetails("Jane Smith", 124, 60000.00);
        employee.displayEmployeeDetails();

        // Creating an instance of Manager class
        Manager manager = new Manager("Bob Johnson", 456, 80000.00, 5, 10000.00);

        // Displaying the details of the Manager
        manager.displayEmployeeDetails();

        // Updating the details of the Manager
        manager.updateEmployeeDetails("Alice Johnson", 457, 90000.00, 10, 15000.00);
        manager.displayEmployeeDetails();

        // Creating an instance of Engineer class
        Engineer engineer = new Engineer("Dave Brown", 789, 70000.00, "Software", 3);

        // Displaying the details of the Engineer
        engineer.displayEmployeeDetails();

        // Updating the details of the Engineer
        engineer.updateEmployeeDetails("Susan Brown", 790, 75000.00, "Hardware", 5);
        engineer.displayEmployeeDetails();

        // Creating an instance of Salesperson class
        Salesperson salesperson = new Salesperson("Mary Davis", 101112, 60000.00,
200000.00, 50);

        // Displaying the details of the Salesperson
        salesperson.displayEmployeeDetails();

        // Updating the details of the Salesperson
        salesperson.updateEmployeeDetails("Mike Davis", 101113, 70000.00, 250000.00, 75);
        salesperson.displayEmployeeDetails();
    }
}
```

Sample output

```
Name: John Smith
Employee ID: 123
Salary: $50000.0

Name: Jane Smith
Employee ID: 124
Salary: $60000.0

Name: Bob Johnson
Employee ID: 456
Salary: $80000.0
Number of Employees Managed: 5
Bonus Amount: $10000.0

Name: Alice Johnson
Employee ID: 457
Salary: $90000.0
Number of Employees Managed: 10
Bonus Amount: $15000.0

Name: Dave Brown
Employee ID: 789
Salary: $70000.0
Field of Engineering: Software
Number of Projects Completed: 3

Name: Susan Brown
Employee ID: 790
Salary: $75000.0
Field of Engineering: Hardware
Number of Projects Completed: 5
```

```
Name: Mary Davis
Employee ID: 101112
Salary: $60000.0
Sales Quota: $200000.0
Number of Sales: 50

Name: Mike Davis
Employee ID: 101113
Salary: $70000.0
Sales Quota: $250000.0
Number of Sales: 75
```

This output shows the details of the employees after they have been created and their details have been updated. You can modify the code to create and update employees with different values to test the classes further.