Laporan Pengerjaan Ujian Tengah Semester
Teknik Pemrograman Praktik

Disusun oleh :

Luthfi Satrio Wicaksono (231524049)

Kelas :

D4 – 1B Teknik Informatika

Tahun Ajaran 2023 – 2024

Bagian A

1. Bagian pada subclass Fulltime yang memiliki superclass Employee dan Interface Koperasi, ini merupakan bentuk solusi dari penerapan multiple inheritance pada bahasa java.

```
public class Fulltime extends Employee implements Koperasi {
    private int overtimeHours; // Total overtime hours
    private int numberOfChildren; // Number of children
    private double loanAmount; // Cooperative loan amount

    public Fulltime(String name, String address, String phoneNumber, String
position) {
        super(name, address, phoneNumber, position); // Position determines
the base salary
        this.overtimeHours = 0;
        this.numberOfChildren = 0;
        this.loanAmount = 0; // Default loan amount is zero
    }

    // Method to set overtime hours
    public void setOvertimeHours(int overtimeHours) {
        this.overtimeHours = overtimeHours;
    }

    // Method to set number of children
    public void setNumberOfChildren(int numberOfChildren) {
        this.numberOfChildren = numberOfChildren;
    }

    // Method to set loan amount
    public void setLoanAmount(double loanAmount) {
        this.loanAmount = loanAmount;
    }

    // Method to get loan amount
    public double getLoanAmount() {
        return loanAmount;
    }

    // Method to calculate total salary
    @Override
    public double getBaseSalary() {
        double salary = super.getBaseSalary() // Base salary
                + calculatePositionAllowance() // Position allowance
                + calculateCommunicationAllowance() // Communication allowance
                + calculateOvertimeAllowance() // Overtime allowance
                + calculateChildAllowance() // Child allowance
                - loanAmount; // Deduct cooperative loan
```

```
        return salary;
    }

    // Method to calculate position allowance
    private double calculatePositionAllowance() {
        switch (getPosition().toLowerCase()) {
            case "staf manager":
                return 5000000; // Position allowance for manager
            case "staf programmer":
                return 2000000; // Position allowance for programmer
            case "staf analis":
                return 3000000; // Position allowance for analyst
            default:
                return 0; // Invalid position, no position allowance
        }
    }

    // Method to calculate communication allowance
    private double calculateCommunicationAllowance() {
        return 500000; // Communication allowance for all positions
    }

    // Method to calculate overtime allowance
    private double calculateOvertimeAllowance() {
        // Overtime allowance for Saturday and Sunday
        double overtimeRate = 30000; // Rate per hour
        double overtimeAllowance = 0;
        if (overtimeHours > 0) {
            overtimeAllowance = overtimeHours * overtimeRate;
        }
        return overtimeAllowance;
    }

    // Method to calculate child allowance
    private double calculateChildAllowance() {
        // Maximum 2 children allowance with 500000 per child
        int maxChildren = 2;
        double childAllowancePerChild = 500000;
        return Math.min(numberOfChildren, maxChildren) *
childAllowancePerChild;
    }

    // Implement method from Koperasi interface
    @Override
    public double loanMonthly(double loanAmount) {
        // Set the loan amount
        setLoanAmount(loanAmount);
```

```
        // Return the loan amount given
        return loanAmount;
    }
}

public class Employee {
    private String name;
    private String address;
    private String phoneNumber;
    private double baseSalary; // Base salary for all employees
    private String position; // Position of the employee

    public Employee(String name, String address, String phoneNumbers,String
position) {
        this.name = name;
        this.address = address;
        this.phoneNumber = phoneNumber;
        this.position = position;
        this.baseSalary = determineBaseSalary(position); // Determine base
salary based on position
    }

    private double determineBaseSalary(String position) {
        switch (position.toLowerCase()) {
            case "staf manager":
                return 5000000;
            case "staf programmer":
            case "staf analis":
                return 3000000;
            default:
                return 0; // Invalid position, return 0
        }
    }

    public double getBaseSalary() {
        return baseSalary;
    }

    public void setBaseSalary(double baseSalary) {
        this.baseSalary = baseSalary;
    }

    // Method to get position
    public String getPosition() {
        return position;
    }

    // Other methods and attributes as needed
```

```
}

public interface Koperasi {
    public double loanMonthly(double loanAmount);
}
```

2. Agregasi has-a relationship terletak pada class department yang merupakan agregasi terhadap employee.

```
import java.util.ArrayList;
import java.util.List;

public class Department {
    private String departmentName;
    private List<Employee> employees; // List of employees in the department

    public Department(String departmentName) {
        this.departmentName = departmentName;
        this.employees = new ArrayList<>();
    }

    // Method to add an employee to the department
    public void addEmployee(Employee employee) {
        employees.add(employee);
    }

    // Method to remove an employee from the department
    public void removeEmployee(Employee employee) {
        employees.remove(employee);
    }

    // Method to get the department name
    public String departmentName() {
        return departmentName;
    }
}
```

B. buatlah program java berdasarkan diagram

1. Buatlah kelas kelas dari diagram kelas yang terdapat pada gambar 1, tambahkan instance filed yang diperlukan!, dan
2. Terapkan enkapsulasi pada kelas kelas yang anda buat!

Jawaban 1-2 :

```
public class Employee {
    private String name;
    private String address;
    private String phoneNumber;
    private double baseSalary; // Base salary for all employees
    private String position; // Position of the employee

    public Employee(String name, String address, String phoneNumbers,String
position) {
        this.name = name;
        this.address = address;
        this.phoneNumber = phoneNumber;
        this.position = position;
        this.baseSalary = determineBaseSalary(position); // Determine base
salary based on position
    }

    private double determineBaseSalary(String position) {
        switch (position.toLowerCase()) {
            case "staf manager":
                return 5000000;
            case "staf programmer":
            case "staf analis":
                return 3000000;
            default:
                return 0; // Invalid position, return 0
        }
    }

    public double getBaseSalary() {
        return baseSalary;
    }

    public void setBaseSalary(double baseSalary) {
        this.baseSalary = baseSalary;
    }

    // Method to get position
    public String getPosition() {
        return position;
```

```
    }


}
```

```
public class Fulltime extends Employee implements Koperasi {
    private int overtimeHours; // Total overtime hours
    private int numberOfChildren; // Number of children
    private double loanAmount; // Cooperative loan amount

    public Fulltime(String name, String address, String phoneNumber, String
position) {
        super(name, address, phoneNumber, position); // Position determines
the base salary
        this.overtimeHours = 0;
        this.numberOfChildren = 0;
        this.loanAmount = 0; // Default loan amount is zero
    }

    // Method to set overtime hours
    public void setOvertimeHours(int overtimeHours) {
        this.overtimeHours = overtimeHours;
    }

    // Method to set number of children
    public void setNumberOfChildren(int numberOfChildren) {
        this.numberOfChildren = numberOfChildren;
    }

    // Method to set loan amount
    public void setLoanAmount(double loanAmount) {
        this.loanAmount = loanAmount;
    }

    // Method to get loan amount
    public double getLoanAmount() {
        return loanAmount;
    }

    // Method to calculate total salary
    @Override
    public double getBaseSalary() {
        double salary = super.getBaseSalary() // Base salary
                + calculatePositionAllowance() // Position allowance
                + calculateCommunicationAllowance() // Communication allowance
                + calculateOvertimeAllowance() // Overtime allowance
                + calculateChildAllowance() // Child allowance
                - loanAmount; // Deduct cooperative loan
```

```
        return salary;
    }

    // Method to calculate position allowance
    private double calculatePositionAllowance() {
        switch (getPosition().toLowerCase()) {
            case "staf manager":
                return 5000000; // Position allowance for manager
            case "staf programmer":
                return 2000000; // Position allowance for programmer
            case "staf analis":
                return 3000000; // Position allowance for analyst
            default:
                return 0; // Invalid position, no position allowance
        }
    }

    // Method to calculate communication allowance
    private double calculateCommunicationAllowance() {
        return 500000; // Communication allowance for all positions
    }

    // Method to calculate overtime allowance
    private double calculateOvertimeAllowance() {
        // Overtime allowance for Saturday and Sunday
        double overtimeRate = 30000; // Rate per hour
        double overtimeAllowance = 0;
        if (overtimeHours > 0) {
            overtimeAllowance = overtimeHours * overtimeRate;
        }
        return overtimeAllowance;
    }

    // Method to calculate child allowance
    private double calculateChildAllowance() {
        // Maximum 2 children allowance with 500000 per child
        int maxChildren = 2;
        double childAllowancePerChild = 500000;
        return Math.min(numberOfChildren, maxChildren) *
childAllowancePerChild;
    }

    // Implement method from Koperasi interface
    @Override
    public double loanMonthly(double loanAmount) {
        // Set the loan amount
        setLoanAmount(loanAmount);
```

```
        // Return the loan amount given
        return loanAmount;
    }
}
```

```
import javax.swing.text.Position;

public class PartTime extends Employee {
    private double hourlyRate; // Tarif per jam untuk karyawan partime
    private int hoursWorked; // Total jam kerja untuk karyawan partime
    private int overtimeHours; // Total jam lembur untuk karyawan partime

    public PartTime(String name, String address, String phoneNumber,  String
Position,double hourlyRate, int hoursWorked, int overtimeHours) {
        super(name, address, phoneNumber, Position); // Gaji pokok untuk semua
karyawan
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
        this.overtimeHours = overtimeHours;
    }

    // Method untuk mengatur tarif per jam
    public void setHourlyRate(double hourlyRate) {
        this.hourlyRate = hourlyRate;
    }

    // Method untuk mengatur total jam kerja
    public void setHoursWorked(int hoursWorked) {
        this.hoursWorked = hoursWorked;
    }

    // Method untuk mengatur total jam lembur
    public void setOvertimeHours(int overtimeHours) {
        this.overtimeHours = overtimeHours;
    }

    // Method untuk menghitung gaji total untuk karyawan partime
    @Override
    public double getBaseSalary() {
        return super.getBaseSalary() + calculateOvertimeAllowance(); // Gaji
pokok + Tunjangan lembur
    }

    // Method untuk menghitung tunjangan lembur
    private double calculateOvertimeAllowance() {
        double overtimeRate = 30000; // Tarif lembur per jam
        return overtimeHours * overtimeRate; // Tunjangan lembur
```

```
    }
}
```

```
public interface Koperasi {
    public double loanMonthly(double loanAmount);
}

import java.util.ArrayList;
import java.util.List;

public class Department {
    private String departmentName;
    private List<Employee> employees; // List of employees in the department

    public Department(String departmentName) {
        this.departmentName = departmentName;
        this.employees = new ArrayList<>();
    }

    // Method to add an employee to the department
    public void addEmployee(Employee employee) {
        employees.add(employee);
    }

    // Method to remove an employee from the department
    public void removeEmployee(Employee employee) {
        employees.remove(employee);
    }

    // Method to get the department name
    public String departmentName() {
        return departmentName;
    }
}
```

3. Buatlah base salary terkait gaji pokok yang diterapkan pada kelas employee

```
      public class Employee {

    private String name;
    private String address;
    private String phoneNumber;
    private double baseSalary; // Base salary for all employees
    private String position; // Position of the employee
```

```
    public Employee(String name, String address, String phoneNumbers,String
position) {
        this.name = name;
        this.address = address;
        this.phoneNumber = phoneNumber;
        this.position = position;
        this.baseSalary = determineBaseSalary(position); // Determine base
salary based on position
    }

    private double determineBaseSalary(String position) {
        switch (position.toLowerCase()) {
            case "staf manager":
                return 5000000;
            case "staf programmer":
            case "staf analis":
                return 3000000;
            default:
                return 0; // Invalid position, return 0
        }
    }

    public double getBaseSalary() {
        return baseSalary;
    }

    public void setBaseSalary(double baseSalary) {
        this.baseSalary = baseSalary;
    }

    // Method to get position
    public String getPosition() {
        return position;
    }


}
```

4. Hasil Implementasi Soal No 4 Bagian B

```
public class Main {
    public static void main(String[] args) {
        // Membuat objek karyawan Asep (sebagai staf programmer yang sudah
bekerja 3 tahun)
        Fulltime asep = new Fulltime("Asep", "Jalan ABC No. 123",
"081234567890", "staf programmer");
        asep.setNumberOfChildren(2); // Asep memiliki 2 anak
```

```
        asep.setOvertimeHours(3); // Asep sudah bekerja selama 3 tahun
        double asepLoanAmount = 500000; // Pinjaman koperasi Asep

        // Membuat objek karyawan Ujang (sebagai karyawan paruh waktu)
        PartTime ujang = new PartTime("Ujang", "Jalan XYZ No. 456",
"089876543210", "staf programmer",0, 30000, 0);

        ujang.setOvertimeHours(5); // Ujang lembur selama 5 jam

        // Menghitung gaji Asep dan Ujang di bulan April
        double totalSalaryAsep = calculateTotalSalaryAprilAsep  (asep,
asepLoanAmount);
        double totalSalaryUjang = calculateTotalSalaryApril(ujang); // Ujang
tidak memiliki pinjaman koperasi

        // Menampilkan total gaji Asep dan Ujang di bulan April
        System.out.println("Total Gaji Asep di bulan April: Rp " +
totalSalaryAsep);
        System.out.println("Total Gaji Ujang di bulan April: Rp " +
totalSalaryUjang);
    }

    // Method untuk menghitung total gaji karyawan di bulan April
    private static double calculateTotalSalaryAprilAsep(Fulltime fulltime,
double loanAmount) {
        double totalSalary = fulltime.getBaseSalary(); // Gaji pokok
        totalSalary += fulltime.calculatePositionAllowance(); // Tunjangan
jabatan
        totalSalary += fulltime.calculateCommunicationAllowance(); //
Tunjangan komunikasi
        totalSalary += fulltime.calculateOvertimeAllowance(); // Tunjangan
lembur
        totalSalary += fulltime.calculateChildAllowance(); // Tunjangan anak
        totalSalary -= loanAmount; // Potongan pinjaman koperasi
        return totalSalary;
    }

    private static double calculateTotalSalaryApril(PartTime partTime) {
        return partTime.getBaseSalary(); // Gaji part-time tidak
memperhitungkan tunjangan lainnya
    }
}
```

Hasil Running :

```
PS D:\SMT 2\Teknik Pemograman\Code Praktikum\UTS>  d:; cd 'd:\SMT 2\Teknik Pemograman\Code Praktikum\UTS'; & 'C:\Program Files\Java\jdk-21\bin
\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Luffysw\AppData\Roaming\Code\User\workspaceStorage\cbb
21e639f014fa0319177c3fb5e59a6\redhat.java\jdt_ws\UTS_c2b966fa\bin' 'Main'
Total Gaji Asep di bulan April: Rp 9680000.0
Total Gaji Ujang di bulan April: Rp 3150000.0
```