Nama : Rizky Satria Gunawan

Kelas : 2C-D3

NIM : 241511089

UTS PBO PRAKTEK

Jawaban bagian A

1. Solusi Multiple Inheritance di Java

Bagian pada diagram yang menunjukkan solusi untuk multiple inheritance di Java adalah hubungan antara kelas Fulltime, kelas abstrak Employee, dan interface LoanMonthly. Java tidak mendukung multiple inheritance (sebuah kelas tidak bisa mewarisi lebih dari satu kelas). Namun, masalah ini bisa diatasi dengan menggunakan kombinasi satu kelas warisan (extends) dan implementasi satu atau lebih interface (implements).

Contoh Syntax code nya :

```
public class Employee extends Person implements LoanMonthly {
   @Override
   public double getLoanMonthly() {
      // hitung cicilan pinjaman koperasi
   }
}
```

2. Relasi Agregasi (Has-A Relationship)

Bagian pada diagram yang menerapkan relasi agregasi (has-a relationship) adalah garis dengan simbol wajik kosong (hollow diamond) yang menghubungkan kelas Department dengan kelas Employee.

Agregasi adalah bentuk asosiasi khusus yang merepresentasikan hubungan "memiliki" atau "terdiri dari" (whole-part), di mana "bagian" (part) dapat ada secara independen dari "keseluruhan" (whole).

Whole (Department): Kelas Department bertindak sebagai "keseluruhan".

Part (Employee): Kelas Employee bertindak sebagai "bagian".

Diagram ini dibaca sebagai: Sebuah Department memiliki (has-a) seorang Employee (atau beberapa Employee). Hubungan ini bersifat agregasi karena jika sebuah objek Department dihancurkan,objek Employee yang terkait dengannya tidak ikut hancur dan bisa saja dipindahkan ke departemen lain. Kehidupan objek Employee tidak bergantung sepenuhnya pada Department.

Contoh Syntax code nya :

```
public class Employee extends Person implements LoanMonthly {
    private Department department;
    public void setDepartment(Department department) {
        this.department = department;
    }
    public Department getDepartment() {
        return department;
    }
}
```

Jawaban Bagian B

1. Lengkapi Kelas kelas

Employee.java

```java
public interface Employee {
    double getSalary();
}
```

Fulltime.java

```java
import java.util.HashMap;
import java.util.Map;

public class Fulltime implements Employee {
    private static final double OVERTIME_RATE = 30_000;
    private static final double COMMUNICATION_ALLOWANCE = 500_000;
    private static final double MANAGER_POSITION_ALLOWANCE = 5_000_000;
    private static final double CHILD_ALLOWANCE = 500_000;
    private static final int MAX_CHILDREN_ALLOWANCE = 2;

    private static final Map<String, Double> DEFAULT_BASE_SALARY;

    static {
        DEFAULT_BASE_SALARY = new HashMap<>();
        DEFAULT_BASE_SALARY.put("Staff Manager", 5_000_000d);
        DEFAULT_BASE_SALARY.put("Staff Programmer", 3_000_000d);
        DEFAULT_BASE_SALARY.put("Staff Analis", 4_000_000d);
    }

    private String name;
    private String position;
    private double baseSalary;
    private int numberOfChildren;
    private int yearsOfService;
    private double cooperativeLoanInstallment;
    private double overtimeHours;

    public Fulltime(String name,
                    String position,
                    int yearsOfService,
                    int numberOfChildren,
```

```
                    double cooperativeLoanInstallment,
                    double overtimeHours) {
        this.name = name;
        this.position = position;
        this.baseSalary = resolveBaseSalary(position);
        this.yearsOfService = yearsOfService;
        this.numberOfChildren = numberOfChildren;
        this.cooperativeLoanInstallment = cooperativeLoanInstallment;
        this.overtimeHours = overtimeHours;

    }

    private double resolveBaseSalary(String position) {
        return DEFAULT_BASE_SALARY.getOrDefault(position, 0d);
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPosition() {
        return position;
    }

    public void setPosition(String position) {
        this.position = position;
        this.baseSalary = resolveBaseSalary(position);
    }

    public double getBaseSalary() {
        return baseSalary;
    }

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

    public int getNumberOfChildren() {
        return numberOfChildren;
    }
```

```
    public void setNumberOfChildren(int numberOfChildren) {
        this.numberOfChildren = numberOfChildren;
    }

    public int getYearsOfService() {
        return yearsOfService;
    }

    public void setYearsOfService(int yearsOfService) {
        this.yearsOfService = yearsOfService;
    }

    public double getCooperativeLoanInstallment() {
        return cooperativeLoanInstallment;
    }

    public void setCooperativeLoanInstallment(double cooperativeLoanInstallment)
{
        this.cooperativeLoanInstallment = cooperativeLoanInstallment;
    }

    public double getOvertimeHours() {
        return overtimeHours;
    }

    public void setOvertimeHours(double overtimeHours) {
        this.overtimeHours = overtimeHours;
    }

    private double calculateOvertimeAllowance() {
        return overtimeHours * OVERTIME_RATE;
    }

    private double calculatePositionAllowance() {
        return "Staff Manager".equalsIgnoreCase(position) ?
MANAGER_POSITION_ALLOWANCE : 0;
    }

    private double calculateChildAllowance() {
        int eligibleChildren = Math.min(numberOfChildren,
MAX_CHILDREN_ALLOWANCE);
        return eligibleChildren * CHILD_ALLOWANCE;
    }

    private double calculateCommunicationAllowance() {
```

```
        return COMMUNICATION_ALLOWANCE;
    }

    @Override
    public double getSalary() {
        double allowances = calculateOvertimeAllowance()
                + calculatePositionAllowance()
                + calculateChildAllowance()
                + calculateCommunicationAllowance();
        return baseSalary + allowances - cooperativeLoanInstallment;
    }
}
```

Parttime.java

```
public class Parttime implements Employee {
    private static final double OVERTIME_RATE = 30_000;

    private String name;
    private double baseSalary;
    private double overtimeHours;

    public Parttime(String name, double baseSalary, double overtimeHours) {
        this.name = name;
        this.baseSalary = baseSalary;
        this.overtimeHours = overtimeHours;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getBaseSalary() {
        return baseSalary;
    }

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

```
    public double getOvertimeHours() {
        return overtimeHours;
    }

    public void setOvertimeHours(double overtimeHours) {
        this.overtimeHours = overtimeHours;
    }

    private double calculateOvertimeAllowance() {
        return overtimeHours * OVERTIME_RATE;
    }

    @Override
    public double getSalary() {
        return baseSalary + calculateOvertimeAllowance();
    }
}
```

Main.java

```
import java.time.LocalTime;

public class Main {
    public static void main(String[] args) {
    double asepOvertimeHours = calculateHours(LocalTime.of(9, 0),
LocalTime.of(12, 0));
        double ujangOvertimeHours = calculateHours(LocalTime.of(13, 0),
LocalTime.of(18, 0));

        Fulltime asep = new Fulltime(
                "Asep",
        "Staff Programmer",
                3,
        2,
                500_000,
                asepOvertimeHours
        );

        Parttime ujang = new Parttime(
                "Ujang",
                3_000_000,
```

```
                ujangOvertimeHours
        );

        double asepSalary = asep.getSalary();
        double ujangSalary = ujang.getSalary();

        System.out.println("Nota Gaji Bulan April 2025");
        System.out.println("==========================");
        printSlip(asep.getName(), "Fulltime", asep.getPosition(),
asepOvertimeHours, asepSalary);
        printSlip(ujang.getName(), "Parttime", "-", ujangOvertimeHours,
ujangSalary);
        System.out.printf("Total Gaji Dibayarkan : Rp %,.0f%n", asepSalary +
ujangSalary);
    }

    private static double calculateHours(LocalTime start, LocalTime end) {
        return (end.toSecondOfDay() - start.toSecondOfDay()) / 3600.0;
    }

    private static void printSlip(String name, String status, String position,
double overtimeHours, double salary) {
        System.out.println("Nama            : " + name);
        System.out.println("Status          : " + status);
        System.out.println("Jabatan         : " + position);
        System.out.printf("Jam Lembur       : %.1f jam%n", overtimeHours);
        System.out.printf("Total Gaji       : Rp %,.0f%n", salary);
    }
}
```

Output dari program ini

```
Total Gaji Dibayarkan : Rp 7,390,000
PS C:\Users\lenovo\OneDrive - Politeknik Negeri Bandung\Documents\rizky satria\Kampus Polban\Mata-Kuliah-PBO\2C_241511089_Ri
zky Satria Gunawan_Uuts_pbo_2025>  c:; cd 'c:\Users\lenovo\OneDrive - Politeknik Negeri Bandung\Documents\rizky satria\Kampu
s Polban\Mata-Kuliah-PBO\2C_241511089_Rizky Satria Gunawan_Uuts_pbo_2025'; & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-X
X:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\lenovo\AppData\Roaming\Code\User\workspaceStorage\1bac37c973bc37e93f5
09684a9dc0496\redhat.java\jdt_ws\2C_241511089_Rizky Satria Gunawan_Uuts_pbo_2025_e55c1b5c\bin' 'Main'
Nota Gaji Bulan April 2025
=========================
Nama            : Asep
Status          : Fulltime
Jabatan         : Staff Programmer
Jam Lembur       : 3.0 jam
Total Gaji       : Rp 4,090,000

Nama            : Ujang
Status          : Parttime
Jabatan         : -
Jam Lembur       : 5.0 jam
Total Gaji       : Rp 3,150,000

Total Gaji Dibayarkan : Rp 7,240,000
PS C:\Users\lenovo\OneDrive - Politeknik Negeri Bandung\Documents\rizky satria\Kampus Polban\Mata-Kuliah-PBO\2C_241511089_Ri
zky Satria Gunawan_Uuts_pbo_2025>
```