Name: Fitri Cahyaniati

Class : SIB2G

ID : 2341760198

Exercise :

- **For A studies case with a minimum of 3 classes that are interconnected associate .**
- **Create at least 1 attribute type ArrayList of objects**
- **Determine attributes and methods in each class**
- **Draw the class diagram .**

Studies Case : System Management Garden Plant

I. Description :

System This used For manage garden plant with monitor and maintain plant in a way efficient . There is three main classes : Garden , Plant , and Worker . Garden Class containing information about garden and plant list as well as workers , class Plants containing information about plants , and Worker class containing information about worker a caring garden plant .

Class 1: Garden

- **Attributes :**
  - GardenName : String – Garden name
  - location : String – Garden location
  - PlantList : ArrayList < Plants > – List of plants in the garden
  - listWorkers : ArrayList < Workers > – List of workers garden
- **Method:**
  - addPlants (Plants) plants) : Add plant new to garden
  - addWorker (Worker) workers) : Add worker new to garden
  - assignWorkersToPlants (Plants) Plants , Workers worker) : Assign worker For nurse plant certain

Class 2: Plants

- **Attributes :**
  - PlantName : String – Plant name
  - Plant type : String – Type plants (example : " Tomato ", " Kangkung ")

- HealthStatus : String – Health status plants (example : " Healthy ", " Need Maintenance ")
- workers : Workers – Workers who are responsible answer For nurse plant

- **Method:**
  - updateHealthStatus (String status) : Updates the health status plant
  - setWorker (Worker workers) : Determine care worker plant

Class 3: Workers

- **Attributes :**
  - WorkerName : String – Worker name
  - idWorker : String – Worker ID
  - PlantList : ArrayList < Plants > – List of plants cared for by the worker
- **Method:**
  - addPlants (Plants) plants) : Add plant to the list of plants cared for
  - deletePlants (Plants plants) : Delete plant from the list of plants cared for

II. Class Diagram :

- Garden class :

Garden
- GardenName : String - location : String - PlantList : ArrayList < Plants > - list of workers : ArrayList < Workers >
+ addPlant ( ) + addWorker ( ) + assignWorkersToPlants ( )

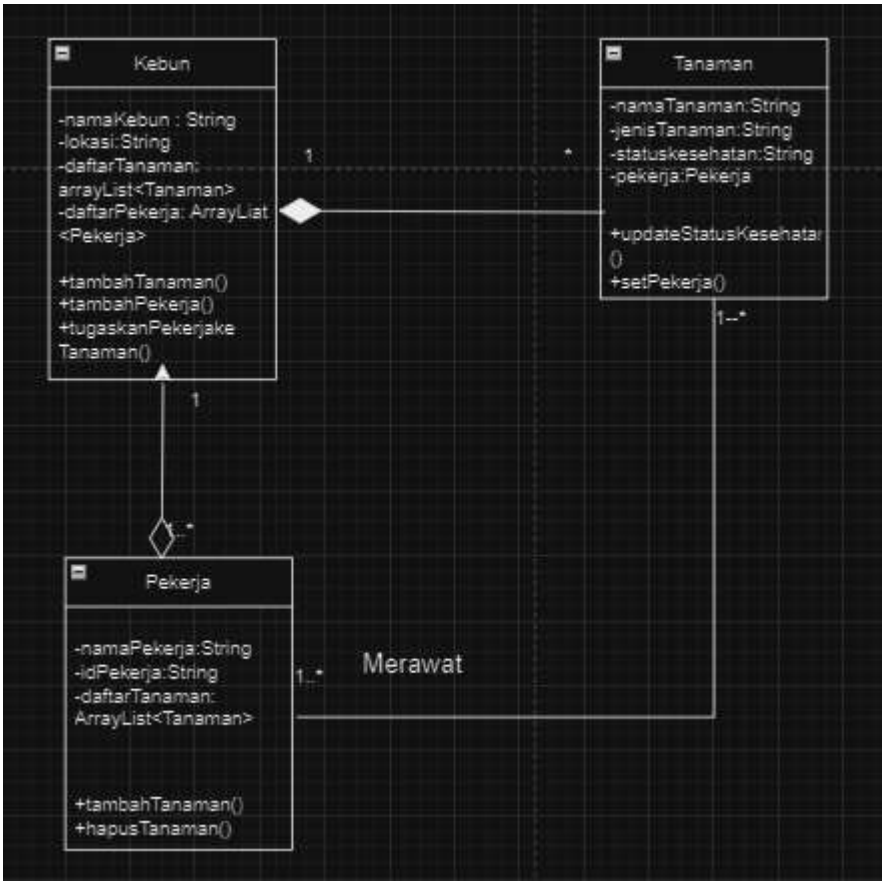
- Plant Class :

Plant
- PlantName : String - Plant type : String - HealthStatus : String - worker : Worker
+ updateHealthStatus ( ) + setWorker ( )

- Worker Class :

Worker
- WorkerName : String - idWorker : String - PlantList : ArrayList < Plants >
+ addPlant ( ) + deletePlant ( )

- **+ (Public)** : Method / attribute Can accessed from outside class .
- **– (Private)** : Method / attribute only Can accessed from in class .



III. Relationship :

- Garden -> Plant :
  Each Garden Can have Many Plants Bahasa Indonesia: But each Plant is only owned by one Garden .
  Garden : 1 Plant : : 0.. *

- Garden -> Workers **:**
  Each Garden Can have Many Workers Bahasa Indonesia: But each Worker only works in one Garden .
  Garden : 1 Worker : 1..*

- Workers -> Plants :
  Each Worker Can care for Many Plants Bahasa Indonesia: but each Plant is cared for by one Worker .
  Worker : 1..* Plant : 1.. *

IV. There is four type relation base between classes **:** Inheritance **,** Association **,** Aggregation **,** and Composition . Here is explanation and application:

a. Association
   - Definition: A "uses-a" relationship where one class uses another class, but does not own or control it. The relationship can be one-to-one, one-to-many, or many-to-many.
   - Example:
     A Garden is associated with a Plant and a Worker. Each garden can have many plants and workers.

b. Aggregation:
   - Definition: A "has-a" relationship where one class contains another class, but the contained objects can exist independently.
   - Example:
     A Worker has a list of Plants to care for, but a Plant can exist without being assigned to a Worker.

c. Composition
   - Definition: A stronger form of Aggregation where the contained objects cannot exist without their container. If the container is destroyed, the contained objects are also destroyed.
   - Example:
     A Garden contains a List of Plants. If the Garden is destroyed, all the Plants in it are also destroyed.

V. Code Program :

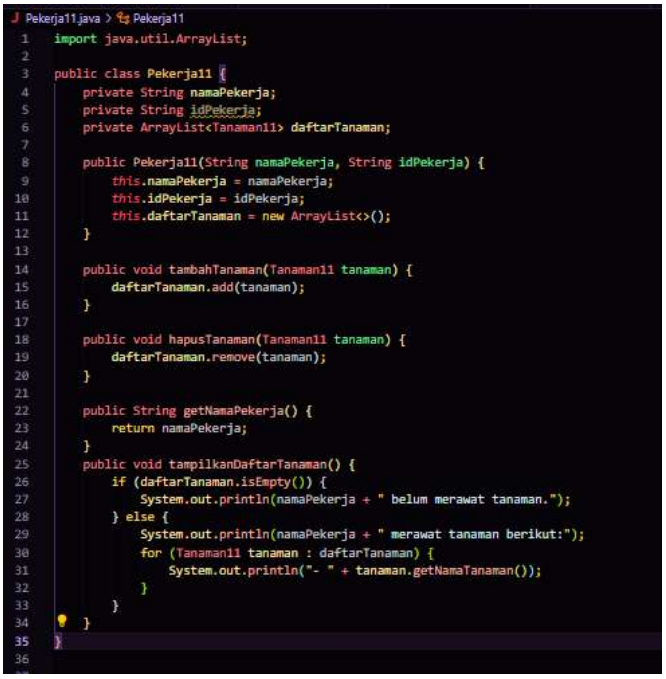
a. Kebun

```java
import java.util.ArrayList;

public class Kebun11 {
    private String namaKebun;
    private String lokasi;
    private ArrayList<Tanaman11> daftarTanaman;
    private ArrayList<Pekerja11> daftarPekerja;

    public Kebun11(String namaKebun, String lokasi) {
        this.namaKebun = namaKebun;
        this.lokasi = lokasi;
        this.daftarTanaman = new ArrayList<>();
        this.daftarPekerja = new ArrayList<>();
    }

    public void tambahTanaman(Tanaman11 tanaman) {
        daftarTanaman.add(tanaman);
    }

    public void tambahPekerja(Pekerja11 pekerja) {
        daftarPekerja.add(pekerja);
    }

    public void tugaskanPekerjaKeTanaman(Tanaman11 tanaman, Pekerja11 pekerja) {
        tanaman.setPekerja(pekerja);
        pekerja.tambahTanaman(tanaman);
    }

    // Getter untuk namaKebun
    public String getNamaKebun() {
        return namaKebun;
    }

    // Getter untuk daftarTanaman
    public ArrayList<Tanaman11> getDaftarTanaman() {
        return daftarTanaman;
    }
}
```

b. Tanaman

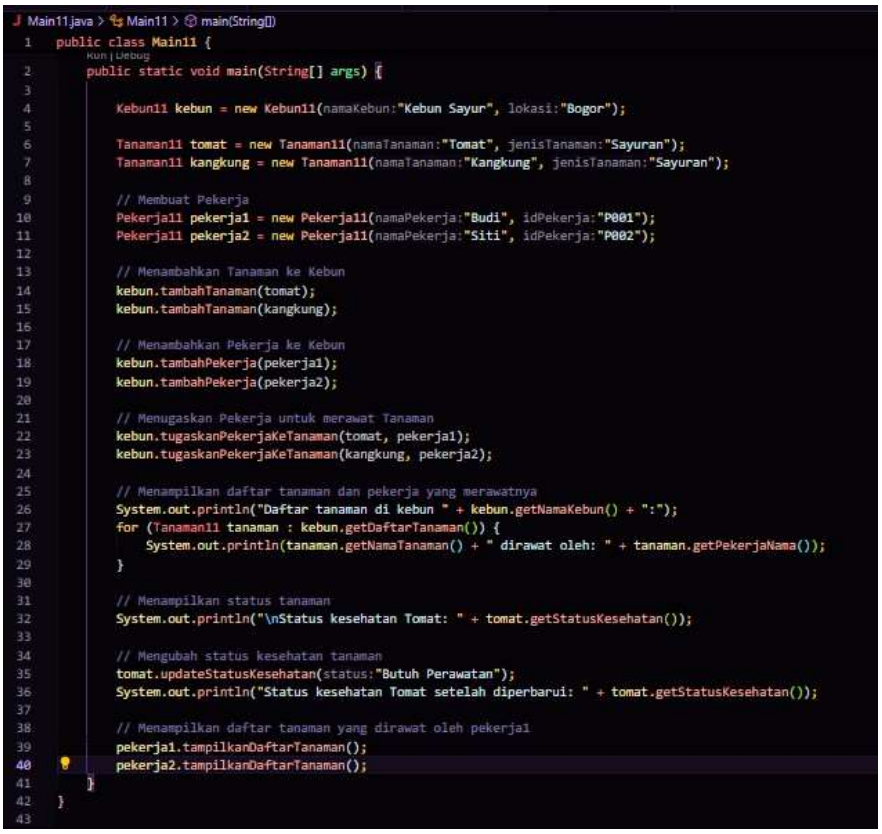
```java
public class Tanaman11 {
    private String namaTanaman;
    private String jenisTanaman;
    private String statusKesehatan;
    private Pekerja11 pekerja;

    public Tanaman11(String namaTanaman, String jenisTanaman) {
        this.namaTanaman = namaTanaman;
        this.jenisTanaman = jenisTanaman;
        this.statusKesehatan = "Sehat"; // Default status
    }

    public void updateStatusKesehatan(String status) {
        this.statusKesehatan = status;
    }

    public void setPekerja(Pekerja11 pekerja) {
        this.pekerja = pekerja;
    }

    public String getStatusKesehatan() {
        return statusKesehatan;
    }

    public String getNamaTanaman() {
        return namaTanaman;
    }
    public String getPekerjaNama() {
        return pekerja != null ? pekerja.getNamaPekerja() : "Belum ditugaskan";
    }
}
```

c. Pekerja



```java
import java.util.ArrayList;

public class Pekerja11 {
    private String namaPekerja;
    private String idPekerja;
    private ArrayList<Tanaman11> daftarTanaman;

    public Pekerja11(String namaPekerja, String idPekerja) {
        this.namaPekerja = namaPekerja;
        this.idPekerja = idPekerja;
        this.daftarTanaman = new ArrayList<>();
    }

    public void tambahTanaman(Tanaman11 tanaman) {
        daftarTanaman.add(tanaman);
    }

    public void hapusTanaman(Tanaman11 tanaman) {
        daftarTanaman.remove(tanaman);
    }

    public String getNamaPekerja() {
        return namaPekerja;
    }
    public void tampilkanDaftarTanaman() {
        if (daftarTanaman.isEmpty()) {
            System.out.println(namaPekerja + " belum merawat tanaman.");
        } else {
            System.out.println(namaPekerja + " merawat tanaman berikut:");
            for (Tanaman11 tanaman : daftarTanaman) {
                System.out.println("- " + tanaman.getNamaTanaman());
            }
        }
    }
}
```

d. Main



```java
public class Main11 {
    public static void main(String[] args) {

        Kebun11 kebun = new Kebun11(namaKebun:"Kebun Sayur", lokasi:"Bogor");

        Tanaman11 tomat = new Tanaman11(namaTanaman:"Tomat", jenisTanaman:"Sayuran");
        Tanaman11 kangkung = new Tanaman11(namaTanaman:"Kangkung", jenisTanaman:"Sayuran");

        // Membuat Pekerja
        Pekerja11 pekerja1 = new Pekerja11(namaPekerja:"Budi", idPekerja:"P001");
        Pekerja11 pekerja2 = new Pekerja11(namaPekerja:"Siti", idPekerja:"P002");

        // Menambahkan Tanaman ke Kebun
        kebun.tambahTanaman(tomat);
        kebun.tambahTanaman(kangkung);

        // Menambahkan Pekerja ke Kebun
        kebun.tambahPekerja(pekerja1);
        kebun.tambahPekerja(pekerja2);

        // Menugaskan Pekerja untuk merawat Tanaman
        kebun.tugaskanPekerjaKeTanaman(tomat, pekerja1);
        kebun.tugaskanPekerjaKeTanaman(kangkung, pekerja2);

        // Menampilkan daftar tanaman dan pekerja yang merawatnya
        System.out.println("Daftar tanaman di kebun " + kebun.getNamaKebun() + ":");
        for (Tanaman11 tanaman : kebun.getDaftarTanaman()) {
            System.out.println(tanaman.getNamaTanaman() + " dirawat oleh: " + tanaman.getPekerjaNama());
        }

        // Menampilkan status tanaman
        System.out.println("\nStatus kesehatan Tomat: " + tomat.getStatusKesehatan());

        // Mengubah status kesehatan tanaman
        tomat.updateStatusKesehatan(status:"Butuh Perawatan");
        System.out.println("Status kesehatan Tomat setelah diperbarui: " + tomat.getStatusKesehatan());

        // Menampilkan daftar tanaman yang dirawat oleh pekerja1
        pekerja1.tampilkanDaftarTanaman();
        pekerja2.tampilkanDaftarTanaman();
    }
}
```

e. Output

```
Messages' '-cp' 'C:\Users\HP\AppData\Roaming\Code\User\workspac
Daftar tanaman di kebun Kebun Sayur:
Tomat dirawat oleh: Budi
Kangkung dirawat oleh: Siti

Status kesehatan Tomat: Sehat
Status kesehatan Tomat setelah diperbarui: Butuh Perawatan
Budi merawat tanaman berikut:
- Tomat
Siti merawat tanaman berikut:
- Kangkung
PS D:\SEMESTER 3\PBO\Java\Jobsheet04\Perkebunan>
```

List of plants in the garden: The program displays each plant in the garden and the workers who are caring for it.
Plant health status: Health status of the Tomato plants before and after being updated (default)
List of plants cared for by workers: Displays the plants being cared for by Budi and Siti