Nama : Agus Riyanto

NIM : 222111855

Kelas : 2KS2

Pemrograman Besbasis Objek

Soal

❑ Sebuah **universitas** terdiri dari beberapa **fakultas** yang terdiri dari universitas berbagai **institut**. Setiap fakultas dan setiap institut memiliki nama. Alamat diketahui untuk setiap lembaga.

❑ Setiap fakultas dipimpin oleh seorang **dekan** yang merupakan **pegawai** universitas.

❑ Jumlah total karyawan diketahui. Karyawan memiliki nomor jaminan sosial, nama, dan alamat email. Semua karyawan memiliki kewajiban yang sama, yaitu melakukan absensi. Ada perbedaan antara **peneliti** dan **tenaga administrasi**.

❑ Perhitungan **persentase kehadiran** berbeda untuk staf penelitian dan administrasi. Tenaga administrasi: berdasarkan kehadiran rutin per hari. Tenaga peneliti: berdasarkan jumlah jam mengajar

❑ Rekan penelitian ditugaskan ke setidaknya satu lembaga. Bidang studi masing-masing rekan peneliti diketahui. Selain itu, rekan penelitian dapat terlibat dalam proyek selama beberapa jam, dan nama, tanggal mulai, dan tanggal akhir **proyek** diketahui. Beberapa rekan penelitian mengajar kursus. Mereka disebut **dosen**.

❑ Kursus memiliki nomor unik (ID), nama, dan durasi mingguan dalam jam.

Pembuatan Diagram UML

1. Langkah pertama : identifikasi class
   a. Dekan memiliki atribut yang sama dengan pegawai, tidak ebih special dari pegawai sehingga bisa diwakili oleh kelas pegawai
   b. Universitas bisa diwakili dengan fakultas
   c. Kelas yang terbentuk :
      i. Faculty
      ii. Institute
      iii. Employee
      iv. Research Associate
      v. Administratives Employee
      vi. Attendance
      vii. Project
      viii. Course
      ix. Lecturer
2. Identifikasi atribut

❑ A university consists of multiple **faculties** which are composed of university various **institutes**. Each **faculty and each institute** has a name. An address is known for each institute.

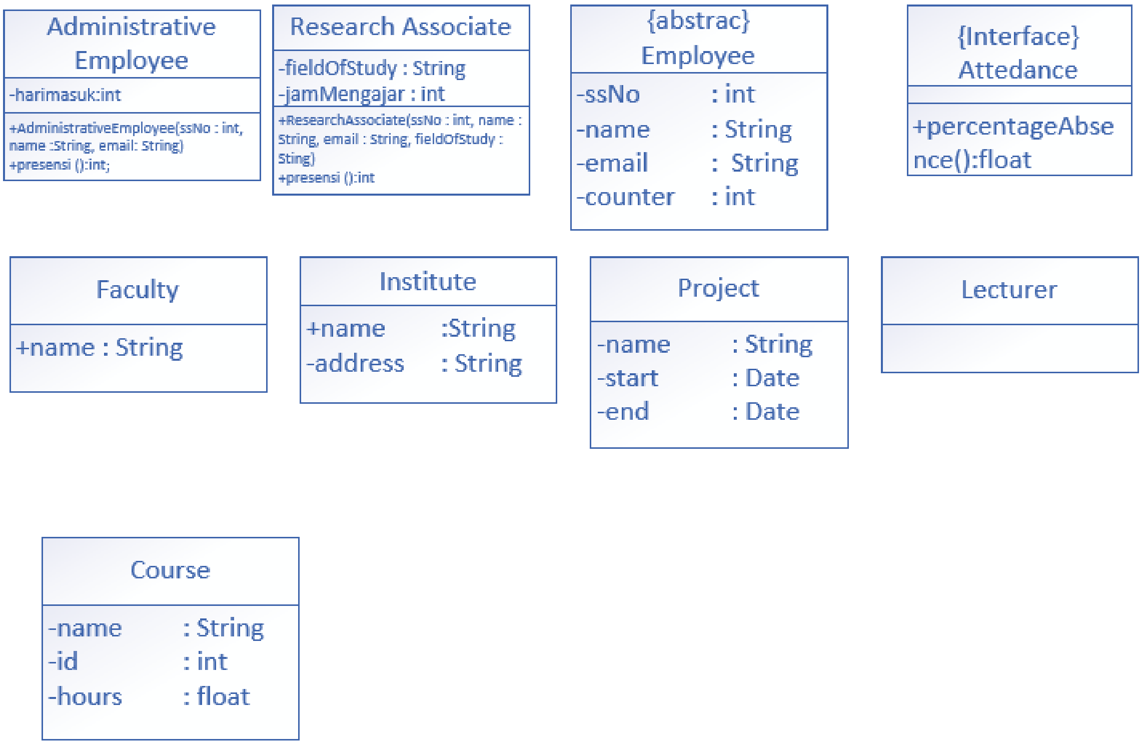
❑ Each faculty is led by a dean, who is an **employee** of the university.

❑ The total number of **employees** is known. **Employees** have a social security number, a name, and an e-mail address. **All employees** have same obligation, namely to take the attendance. There is a distinction between **research** and **administrative personnel.**

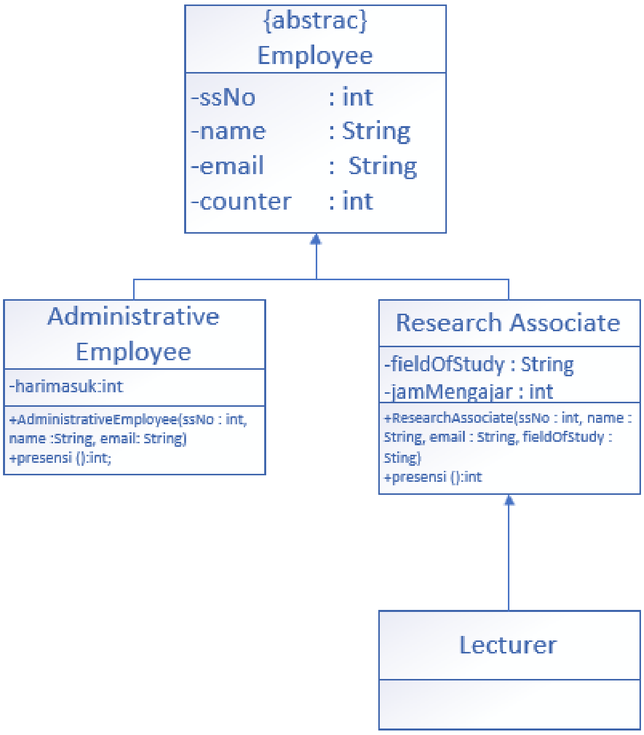
❑ Calculation of **attendance percentages** are different for both research and administrative personnel. **Administrative personnel**: based on routine attendance per day. **Research personnel**: based on the number of teaching hours

❑ **Research associates** are assigned to at least one institute. The field of study of each research associate is known. Furthermore, research associates can be involved in projects for a certain number of hours, and the name, starting date, and end date of the projects are known. Some research associates teach courses. They are called lecturers.

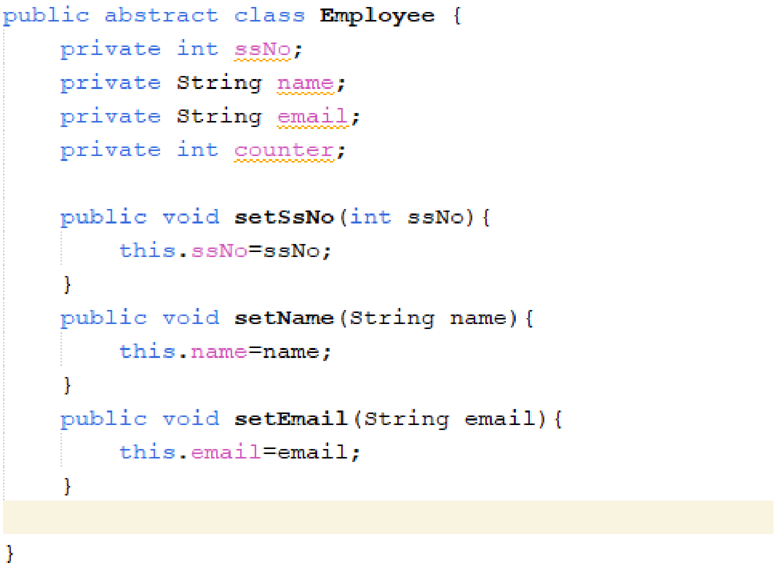
❑ Courses have a unique number (ID), a name, and a weekly duration in hours.



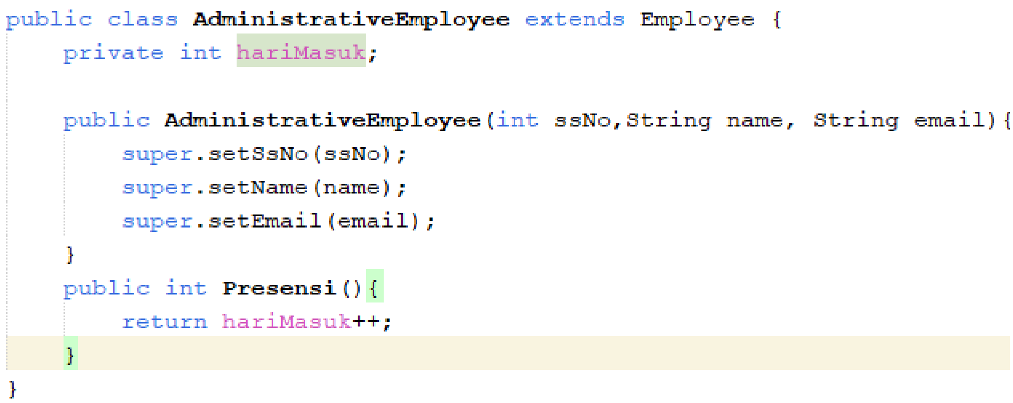
3. Identifiksi Hubungan
    a. There is a distinction between **research** and **administrative personnel.** Some research associates teach courses. They are called lecturers. (Inherentance)



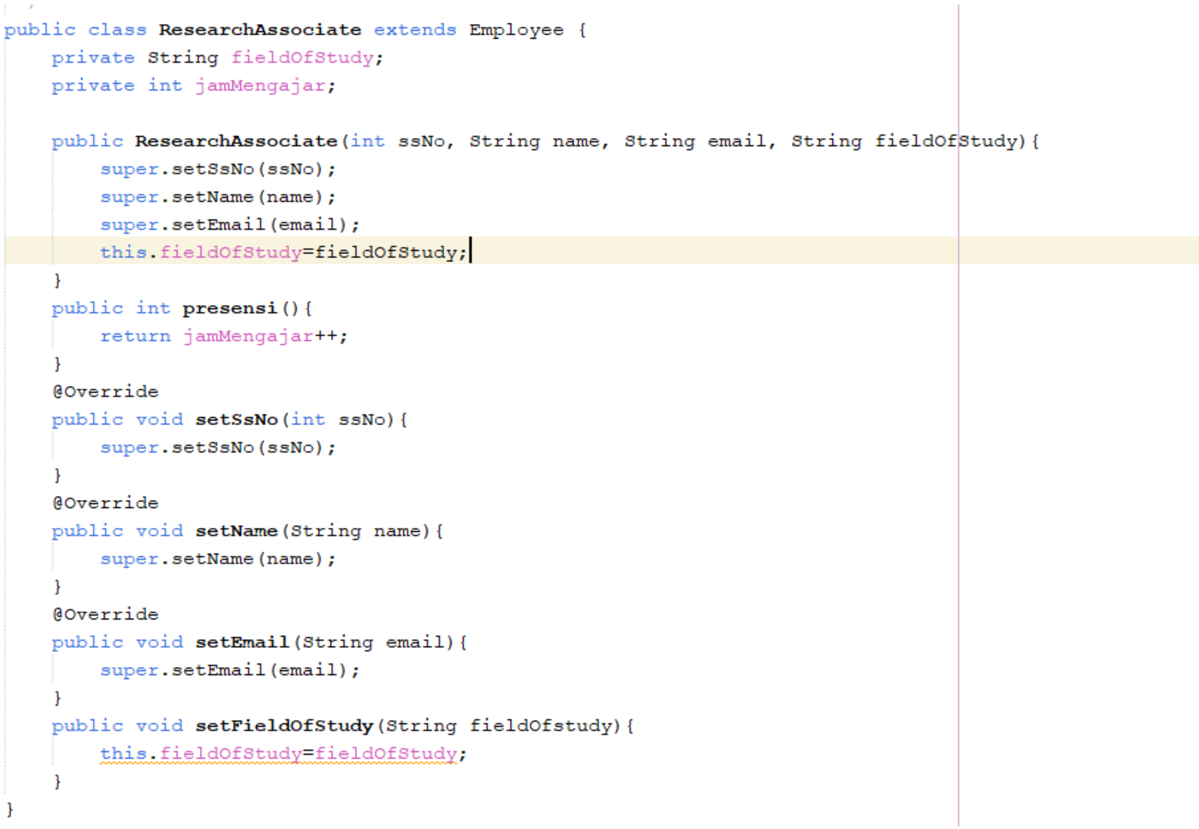
Code



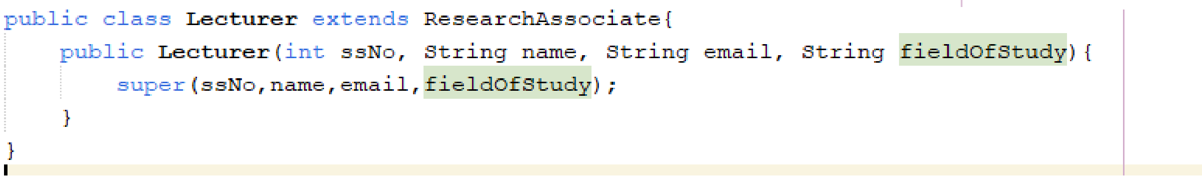
```
public abstract class Employee {
    private int ssNo;
    private String name;
    private String email;
    private int counter;

    public void setSsNo(int ssNo){
        this.ssNo=ssNo;
    }
    public void setName(String name){
        this.name=name;
    }
    public void setEmail(String email){
        this.email=email;
    }

}
```



```
public class AdministrativeEmployee extends Employee {
    private int hariMasuk;

    public AdministrativeEmployee(int ssNo,String name, String email){
        super.setSsNo(ssNo);
        super.setName(name);
        super.setEmail(email);
    }
    public int Presensi(){
        return hariMasuk++;
    }
}
```

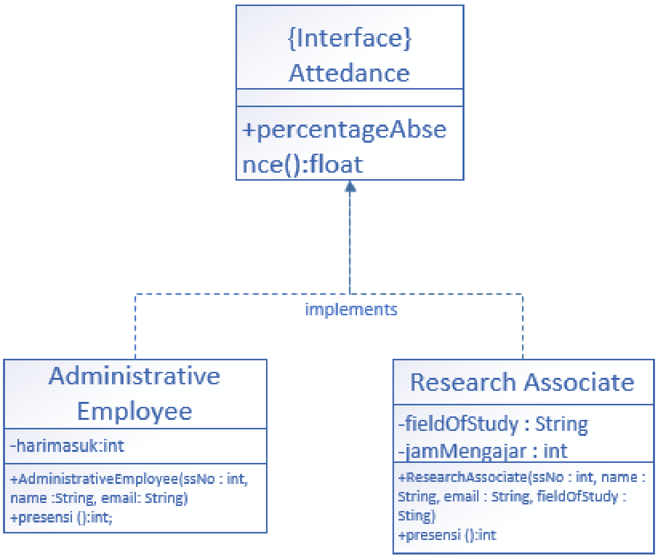


```
public class ResearchAssociate extends Employee {
    private String fieldOfStudy;
    private int jamMengajar;

    public ResearchAssociate(int ssNo, String name, String email, String fieldOfStudy){
        super.setSsNo(ssNo);
        super.setName(name);
        super.setEmail(email);
        this.fieldOfStudy=fieldOfStudy;
    }
    public int presensi(){
        return jamMengajar++;
    }
    @Override
    public void setSsNo(int ssNo){
        super.setSsNo(ssNo);
    }
    @Override
    public void setName(String name){
        super.setName(name);
    }
    @Override
    public void setEmail(String email){
        super.setEmail(email);
    }
    public void setFieldOfStudy(String fieldOfstudy){
        this.fieldOfStudy=fieldOfStudy;
    }
}
```

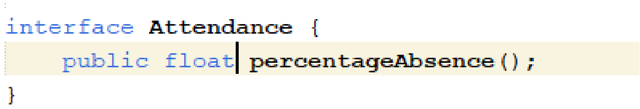


```
public class Lecturer extends ResearchAssociate{
    public Lecturer(int ssNo, String name, String email, String fieldOfStudy){
        super(ssNo,name,email,fieldOfStudy);
    }
}
```

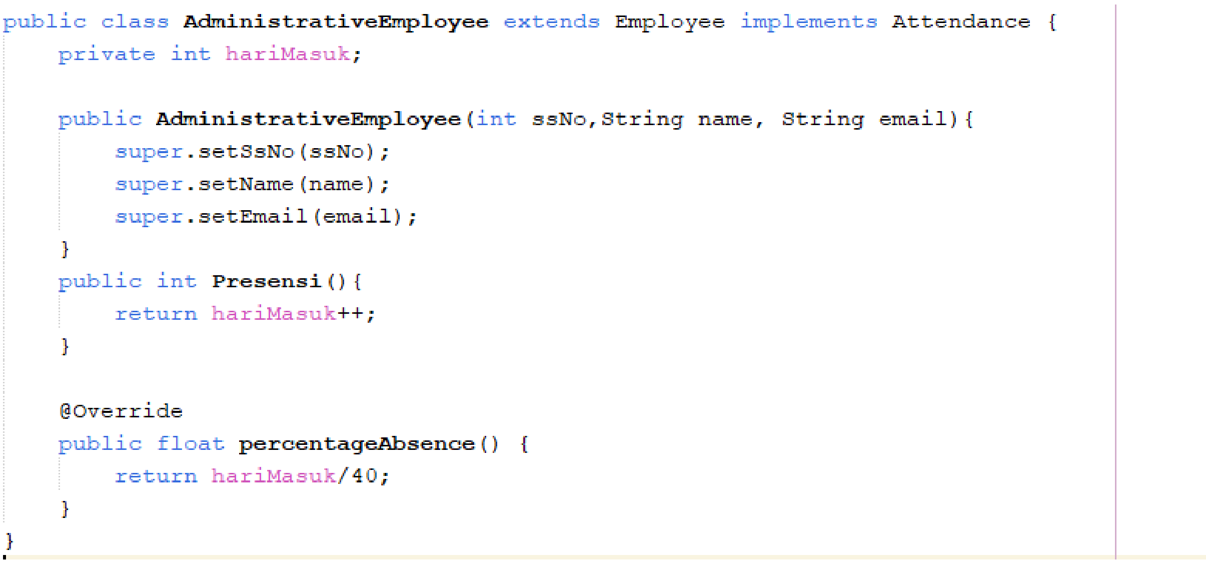
b. Calculation of **attendance percentages** are different for both research and administrative personnel. **Administrative personnel**: based on routine attendance per day. **Research personnel**: based on the number of teaching hours (interface)


{Interface}
Attedance
+percentageAbsence():float
implements
Administrative Employee
-harimasuk:int
+AdministrativeEmployee(ssNo : int, name :String, email: String)
+presensi ():int;
Research Associate
-fieldOfStudy : String
-jamMengajar : int
+ResearchAssociate(ssNo : int, name : String, email : String, fieldOfStudy : Sting)
+presensi ():int

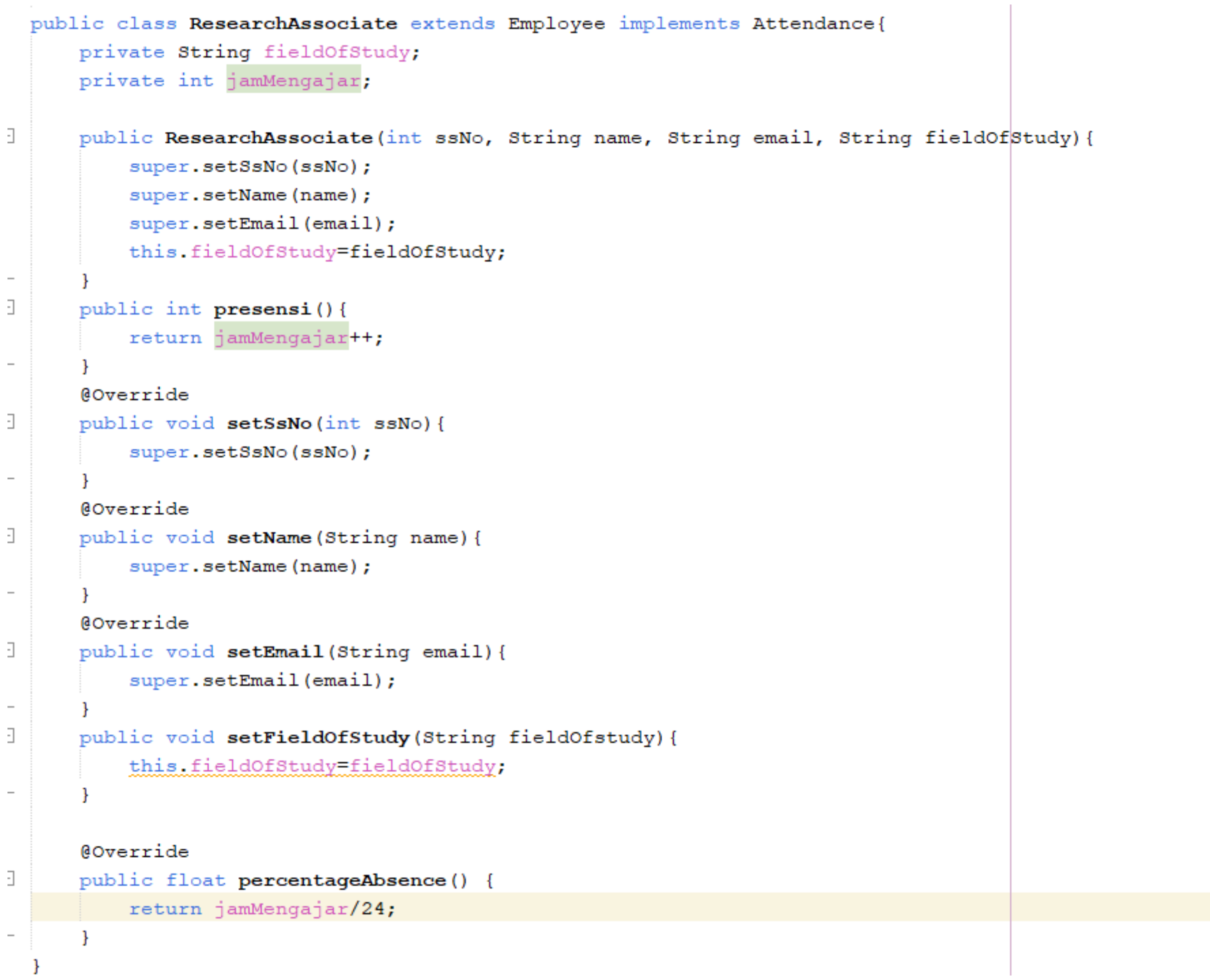
Code



```
interface Attendance {
    public float percentageAbsence();
}
```

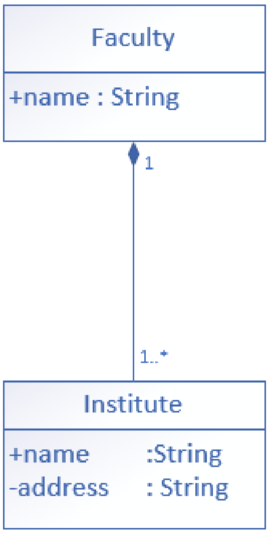


```
public class AdministrativeEmployee extends Employee implements Attendance {
    private int hariMasuk;

    public AdministrativeEmployee(int ssNo,String name, String email){
        super.setSsNo(ssNo);
        super.setName(name);
        super.setEmail(email);
    }
    public int Presensi(){
        return hariMasuk++;
    }

    @Override
    public float percentageAbsence() {
        return hariMasuk/40;
    }
}
```

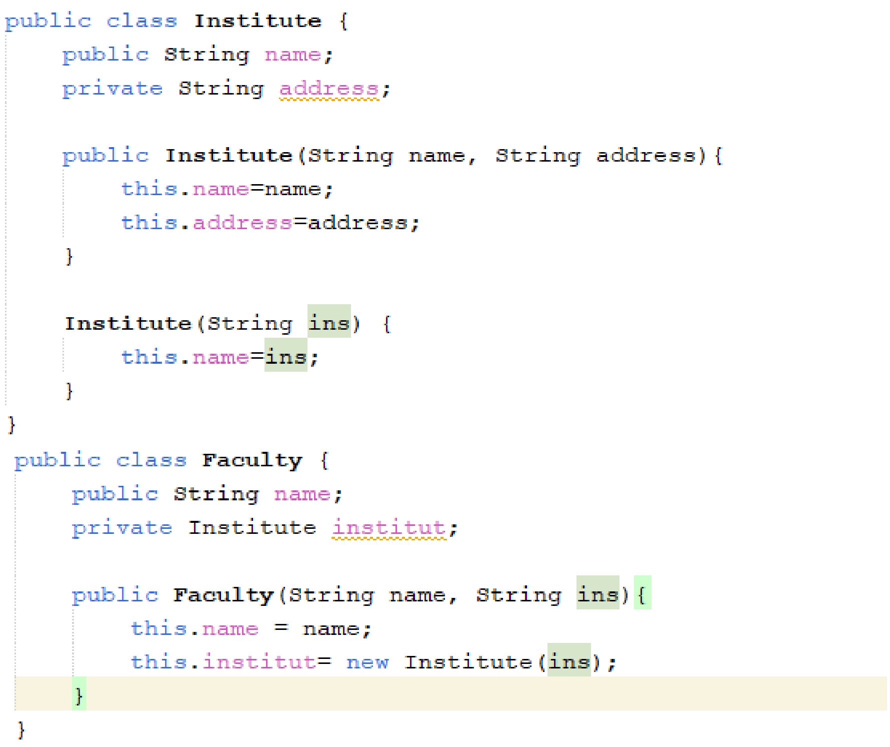


```
public class ResearchAssociate extends Employee implements Attendance{
    private String fieldOfStudy;
    private int jamMengajar;

    public ResearchAssociate(int ssNo, String name, String email, String fieldOfStudy){
        super.setSsNo(ssNo);
        super.setName(name);
        super.setEmail(email);
        this.fieldOfStudy=fieldOfStudy;
    }
    public int presensi(){
        return jamMengajar++;
    }
    @Override
    public void setSsNo(int ssNo){
        super.setSsNo(ssNo);
    }
    @Override
    public void setName(String name){
        super.setName(name);
    }
    @Override
    public void setEmail(String email){
        super.setEmail(email);
    }
    public void setFieldOfStudy(String fieldOfstudy){
        this.fieldOfStudy=fieldOfStudy;
    }

    @Override
    public float percentageAbsence()  {
        return jamMengajar/24;
    }
}
```

c. A university consists of multiple **faculties** which are composed of university various **institutes**. (Composition Many to One)

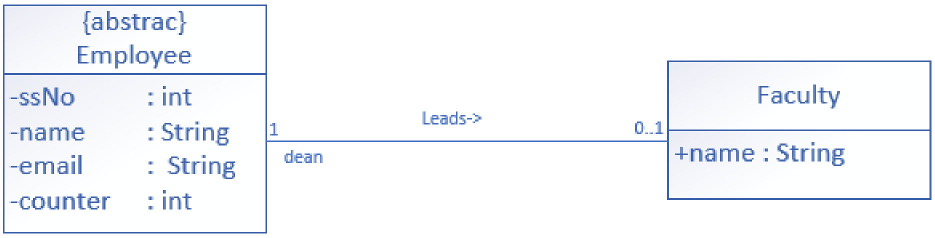


Code

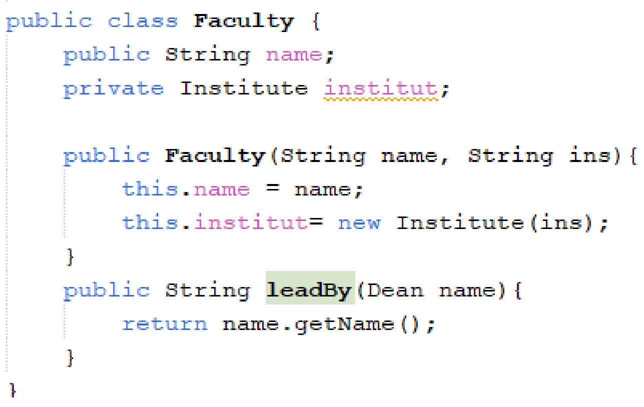


```
public class Institute {
    public String name;
    private String address;

    public Institute(String name, String address){
        this.name=name;
        this.address=address;
    }

    Institute(String ins) {
        this.name=ins;
    }
}
public class Faculty {
    public String name;
    private Institute institut;

    public Faculty(String name, String ins){
        this.name = name;
        this.institut= new Institute(ins);
    }
}
```

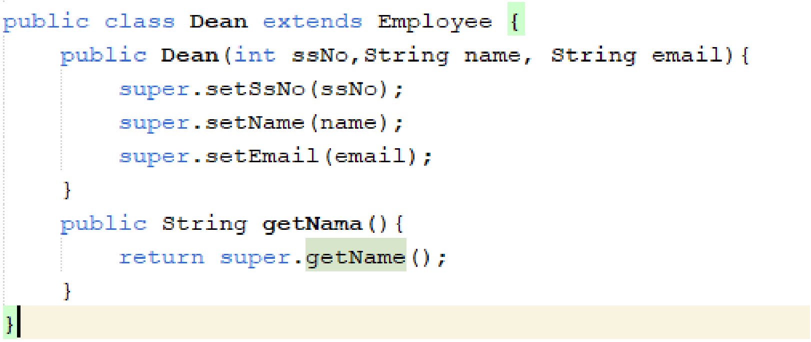
d. Each faculty is led by a dean, who is an **employee** of the university (association).



Code

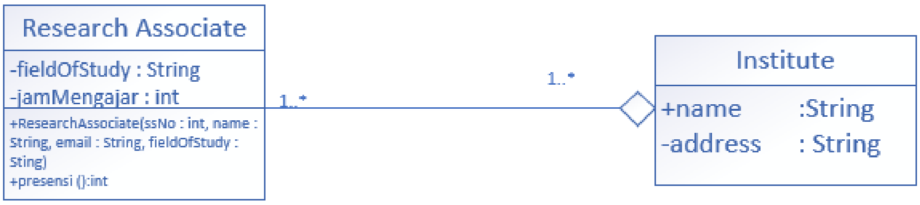


```
public class Faculty {
    public String name;
    private Institute institut;

    public Faculty(String name, String ins){
        this.name = name;
        this.institut= new Institute(ins);
    }
    public String leadBy(Dean name){
        return name.getName();
    }
}
```



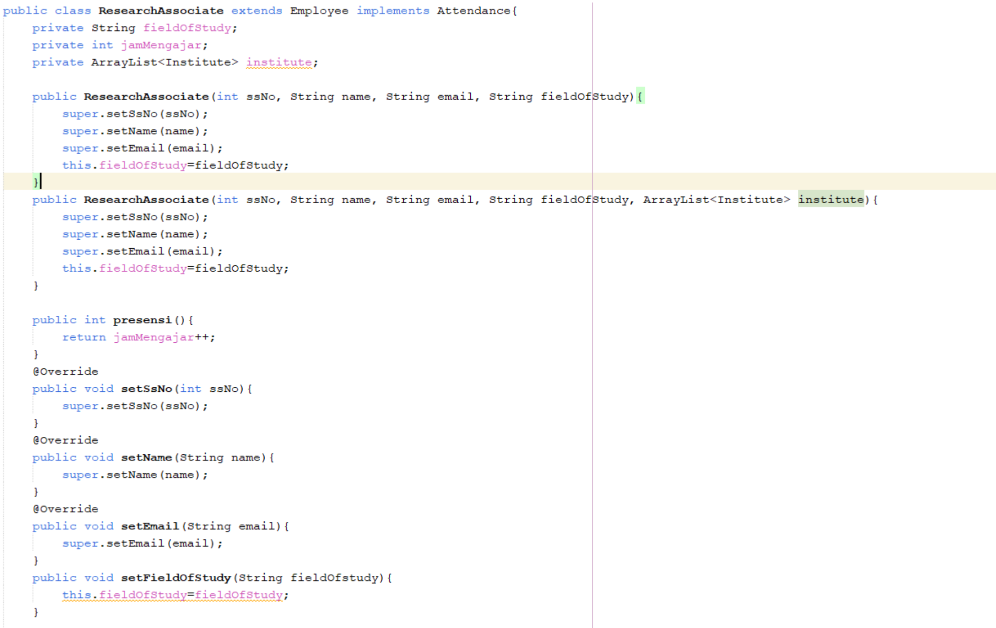
```java
public class Dean extends Employee {
    public Dean(int ssNo,String name, String email){
        super.setSsNo(ssNo);
        super.setName(name);
        super.setEmail(email);
    }
    public String getNama(){
        return super.getName();
    }
}
```

e. **Research associates** are assigned to at least one institute. (Agregasi)

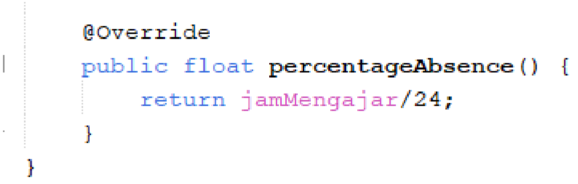


Research Association merupakan bagian dari institute, tetapi keberadaanya tidak bergantung

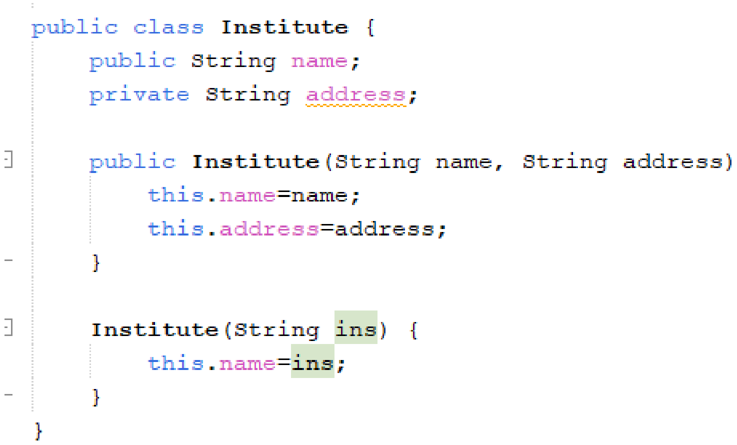
Code



```java
public class ResearchAssociate extends Employee implements Attendance{
    private String fieldOfStudy;
    private int jamMengajar;
    private ArrayList<Institute> institute;

    public ResearchAssociate(int ssNo, String name, String email, String fieldOfStudy){
        super.setSsNo(ssNo);
        super.setName(name);
        super.setEmail(email);
        this.fieldOfStudy=fieldOfStudy;
    }
    public ResearchAssociate(int ssNo, String name, String email, String fieldOfStudy, ArrayList<Institute> institute){
        super.setSsNo(ssNo);
        super.setName(name);
        super.setEmail(email);
        this.fieldOfStudy=fieldOfStudy;
    }

    public int presensi(){
        return jamMengajar++;
    }
    @Override
    public void setSsNo(int ssNo){
        super.setSsNo(ssNo);
    }
    @Override
    public void setName(String name){
        super.setName(name);
    }
    @Override
    public void setEmail(String email){
        super.setEmail(email);
    }
    public void setFieldOfStudy(String fieldOfstudy){
        this.fieldOfStudy=fieldOfStudy;
    }
```

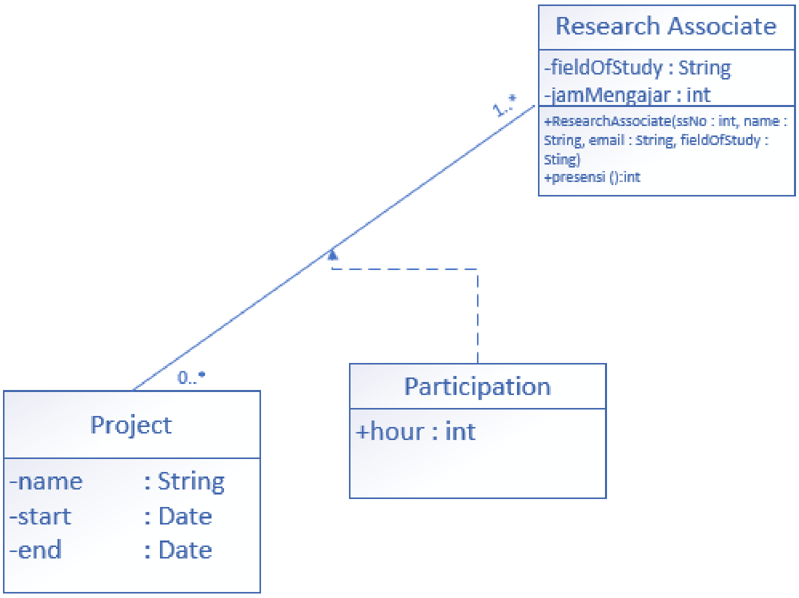


```java
    @Override
    public float percentageAbsence() {
        return jamMengajar/24;
    }
}
```



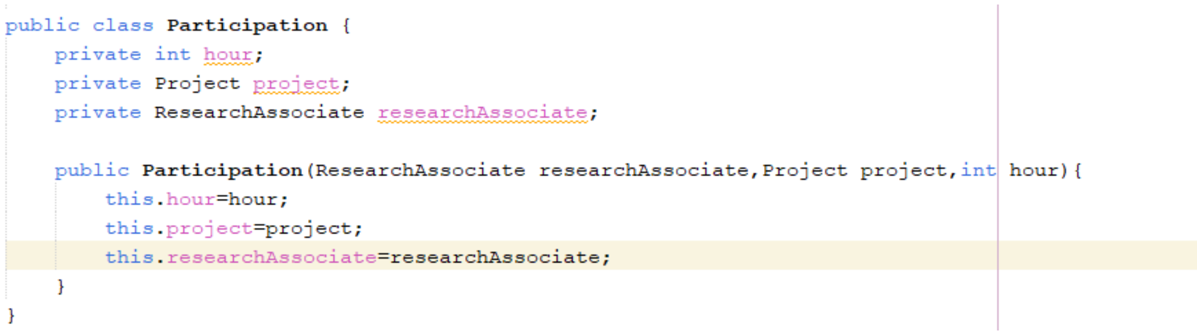
```
public class Institute {
    public String name;
    private String address;

    public Institute(String name, String address)
        this.name=name;
        this.address=address;
    }

    Institute(String ins) {
        this.name=ins;
    }
}
```

f. Furthermore, research associates can be involved in projects for a certain number of hours, and the name, starting date, and end date of the projects are known. (Asociation)

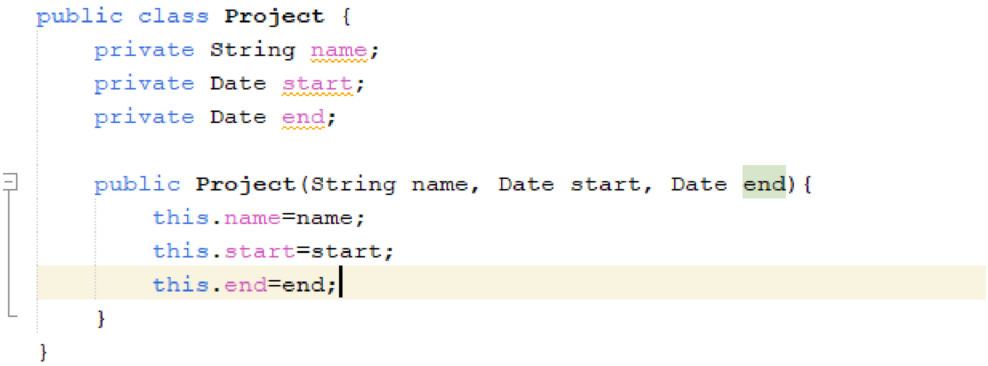


Research associate berpartisipasi Project

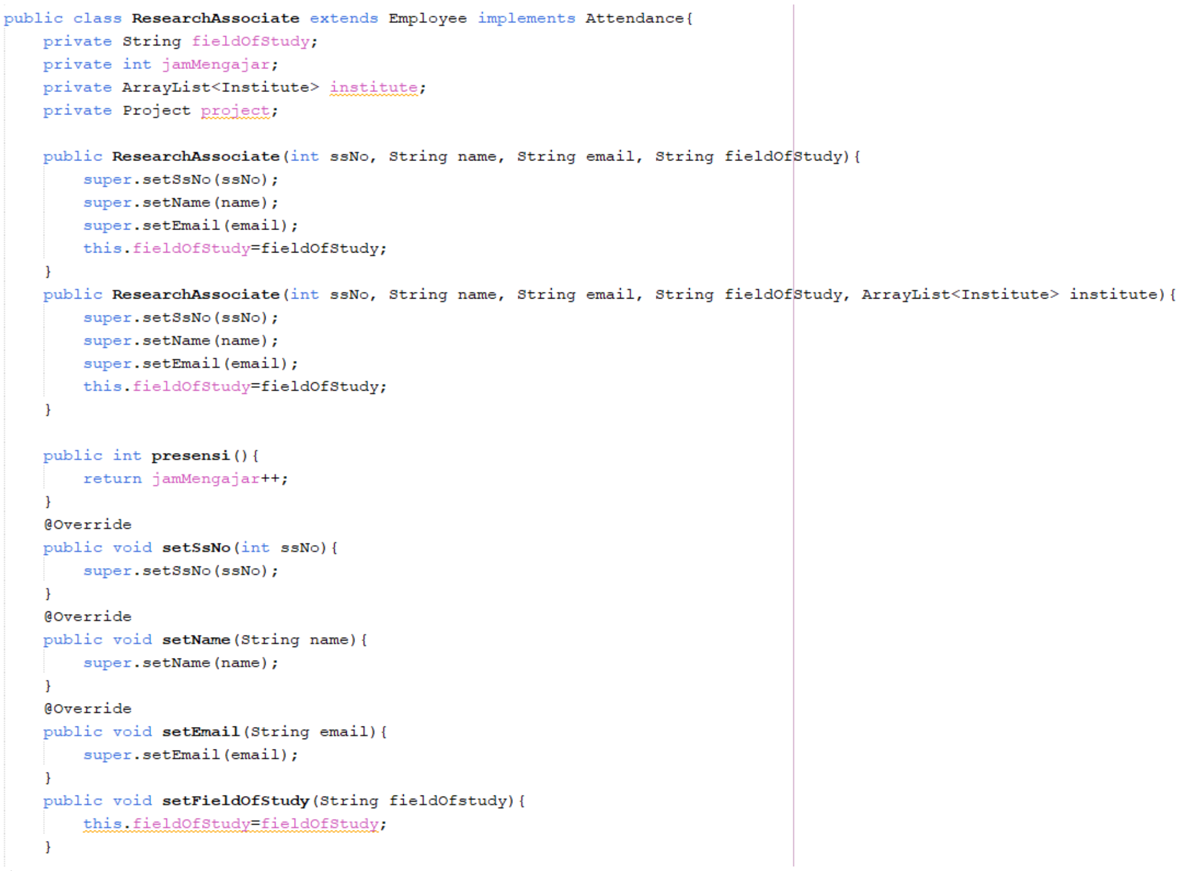
Code



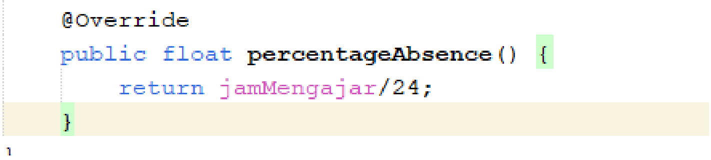
```
public class Participation {
    private int hour;
    private Project project;
    private ResearchAssociate researchAssociate;

    public Participation(ResearchAssociate researchAssociate,Project project,int hour){
        this.hour=hour;
        this.project=project;
        this.researchAssociate=researchAssociate;
    }
}
```



```
public class Project {
    private String name;
    private Date start;
    private Date end;

    public Project(String name, Date start, Date end){
        this.name=name;
        this.start=start;
        this.end=end;
    }
}
```

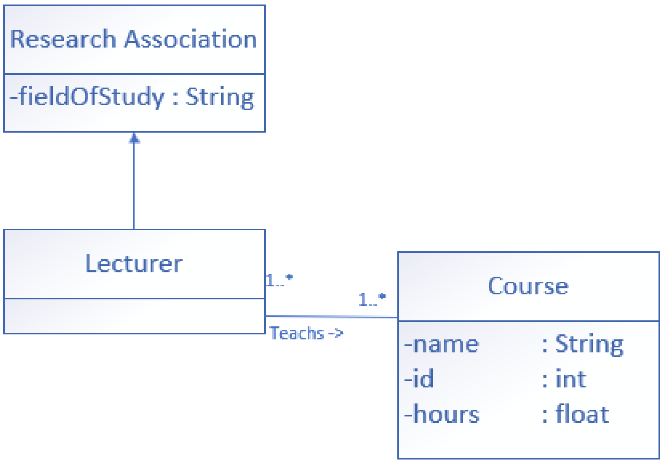


```
public class ResearchAssociate extends Employee implements Attendance{
    private String fieldOfStudy;
    private int jamMengajar;
    private ArrayList<Institute> institute;
    private Project project;

    public ResearchAssociate(int ssNo, String name, String email, String fieldOfStudy){
        super.setSsNo(ssNo);
        super.setName(name);
        super.setEmail(email);
        this.fieldOfStudy=fieldOfStudy;
    }
    public ResearchAssociate(int ssNo, String name, String email, String fieldOfStudy, ArrayList<Institute> institute){
        super.setSsNo(ssNo);
        super.setName(name);
        super.setEmail(email);
        this.fieldOfStudy=fieldOfStudy;
    }

    public int presensi(){
        return jamMengajar++;
    }
    @Override
    public void setSsNo(int ssNo){
        super.setSsNo(ssNo);
    }
    @Override
    public void setName(String name){
        super.setName(name);
    }
    @Override
    public void setEmail(String email){
        super.setEmail(email);
    }
    public void setFieldOfStudy(String fieldOfstudy){
        this.fieldOfStudy=fieldOfStudy;
    }
```



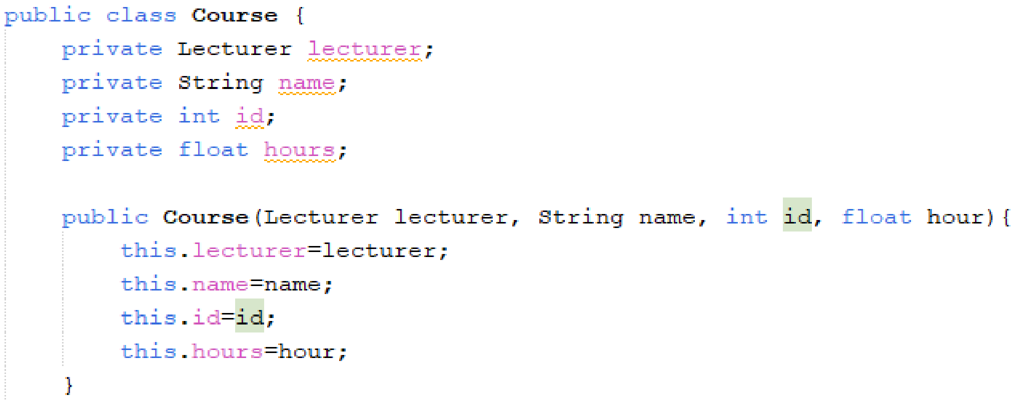
```
    @Override
    public float percentageAbsence() {
        return jamMengajar/24;
    }
```

g. Some research associates teach courses. They are called lecturers.

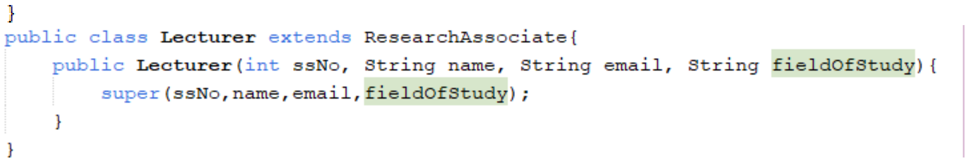


Lecturer inherits semua karakteristik, association, dan agregation dari Research Association.

Code

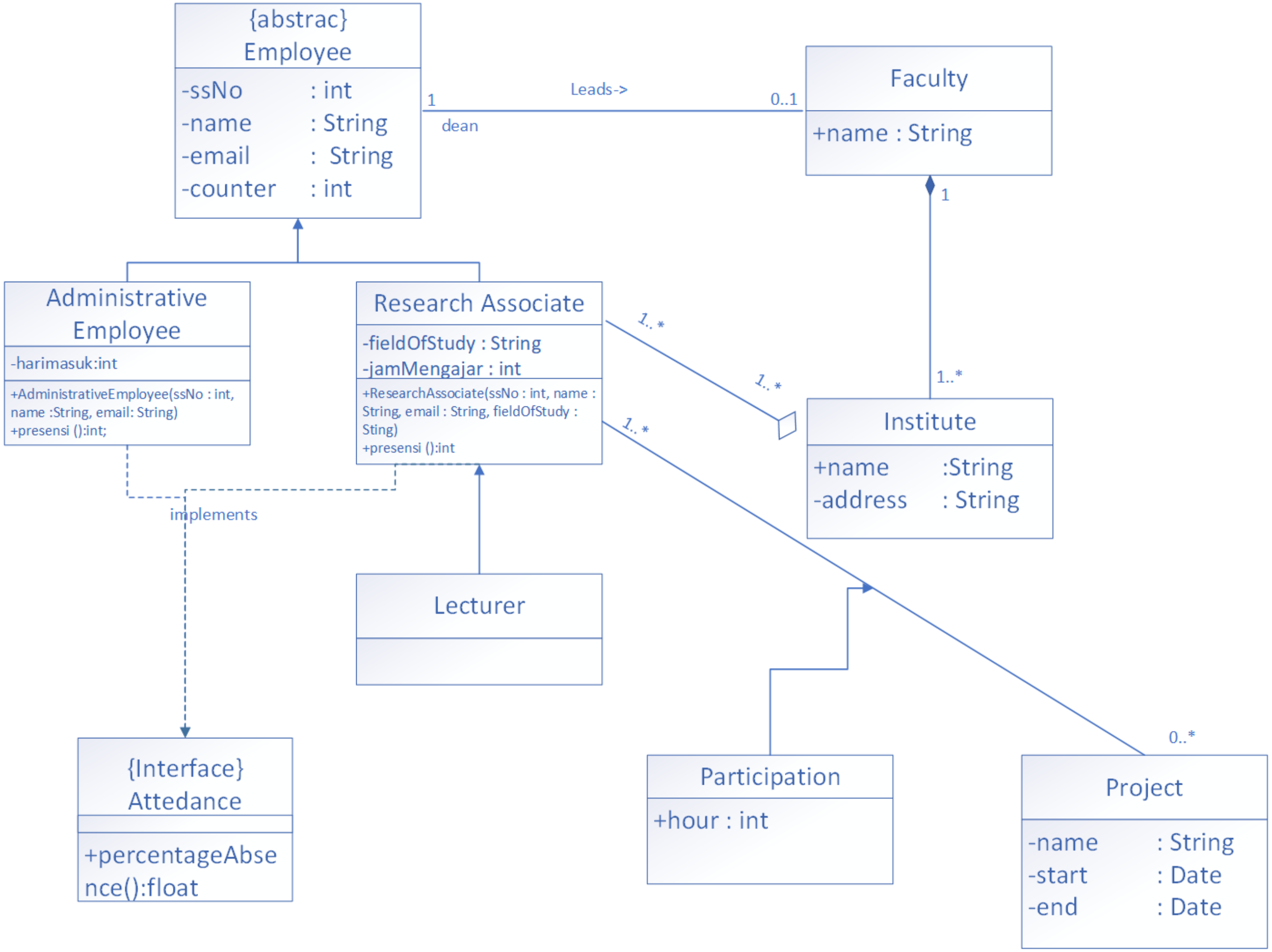


```
public class Course {
    private Lecturer lecturer;
    private String name;
    private int id;
    private float hours;

    public Course(Lecturer lecturer, String name, int id, float hour){
        this.lecturer=lecturer;
        this.name=name;
        this.id=id;
        this.hours=hour;
    }
}
```



```
public class Lecturer extends ResearchAssociate{
    public Lecturer(int ssNo, String name, String email, String fieldOfStudy){
        super(ssNo,name,email,fieldOfStudy);
    }
}
```

h. Class diagram Complete


{abstrac}
Employee
-ssNo : int
-name : String
-email : String
-counter : int
1
Leads->
0..1
dean
Faculty
+name : String
1
1..*
Administrative Employee
-harimasuk:int
+AdministrativeEmployee(ssNo : int, name :String, email: String)
+presensi ():int;
Research Associate
-fieldOfStudy : String
-jamMengajar : int
+ResearchAssociate(ssNo : int, name : String, email : String, fieldOfStudy : Sting)
+presensi ():int
1..*
1..*
1..*
Institute
+name :String
-address : String
implements
Lecturer
{Interface}
Attedance
+percentageAbsence():float
Participation
+hour : int
0..*
Project
-name : String
-start : Date
-end : Date