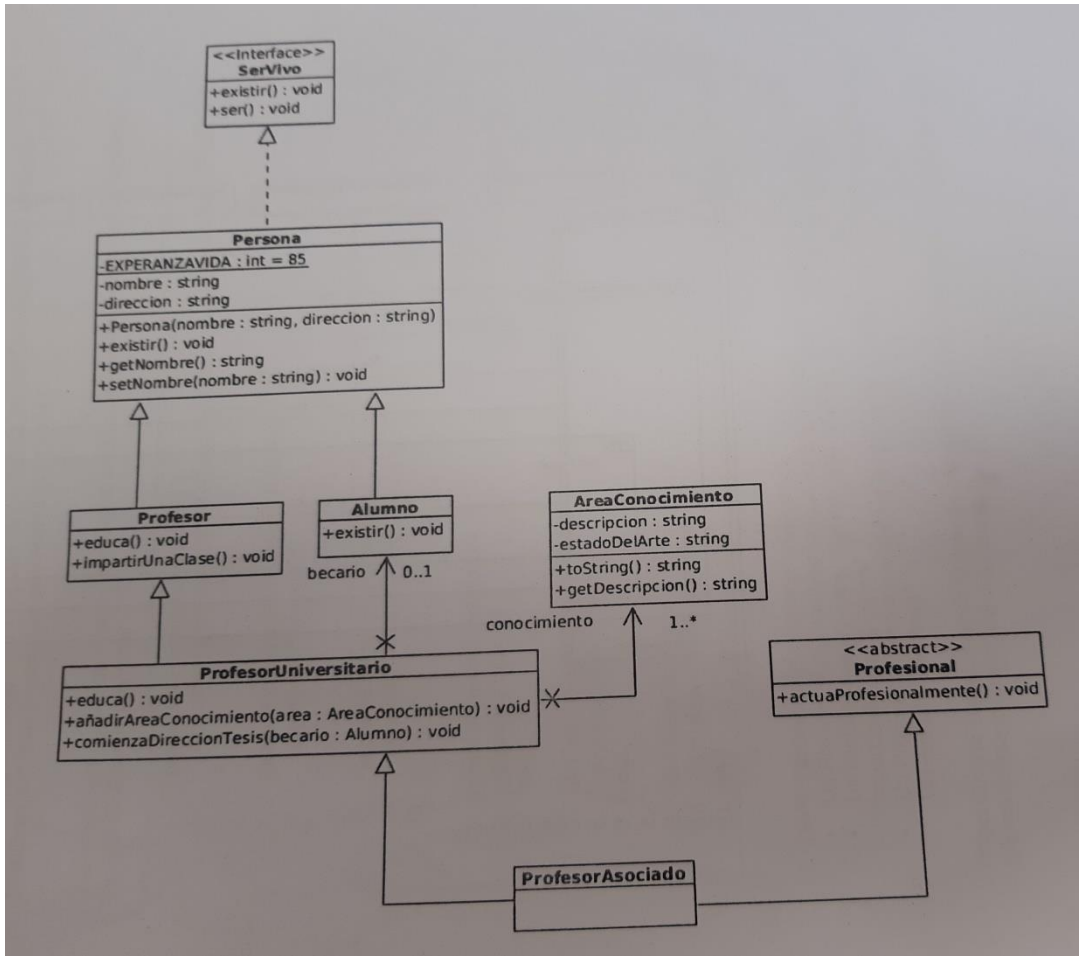
Clase de Repaso PDOO

Ejercicios preparados por

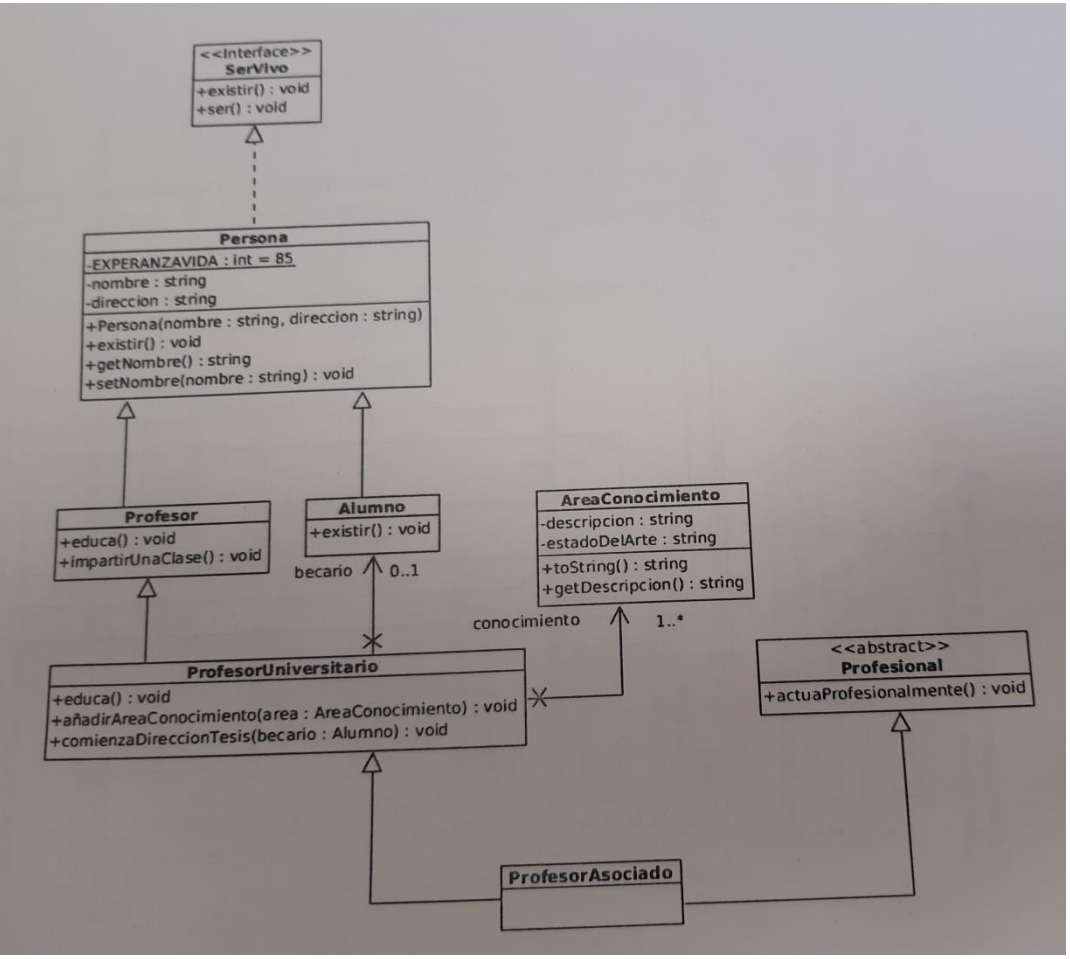
- Nuria Medina-

Ejercicio 1



Implementa el constructor de Profesor y ProfesorUniversitario en Java y Ruby.

Ejercicio 1



Implementa el constructor de Profesor y ProfesorUniversitario en Java y Ruby.

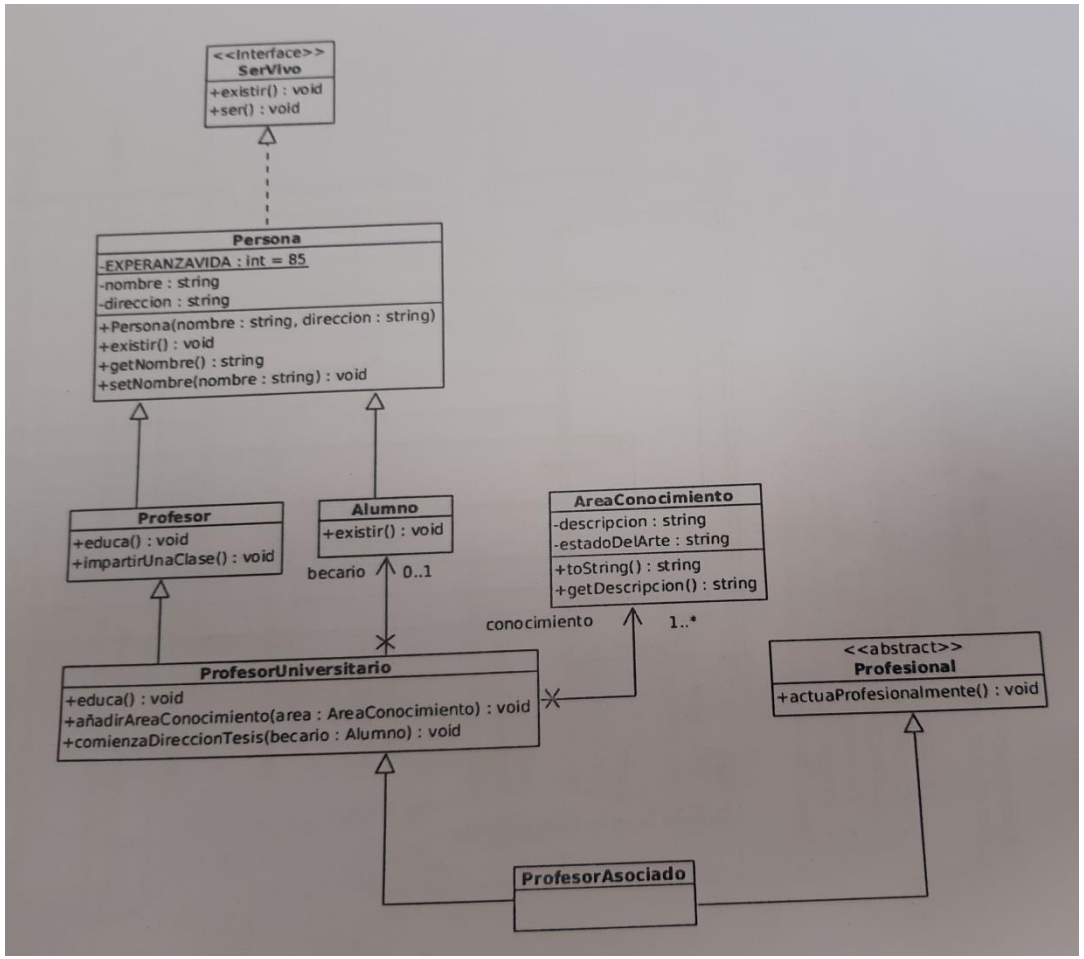
JAVA:

```
public Profesor (String n, String d){
  super(n, d);
}

public ProfesorUniversitario (String n, String
d, Alumno al){
  super(n, d);
  becario = al;
  conocimiento = new ArrayList();
}
```

RUBY:

```
#En Profesor
def initialize(n, d)
   super(n,d)
end

#En ProfesorUniversitario

def initialize(n, d, al)
   super(n,d)
   @becario = al
   @conocimiento = Array.new
end
```

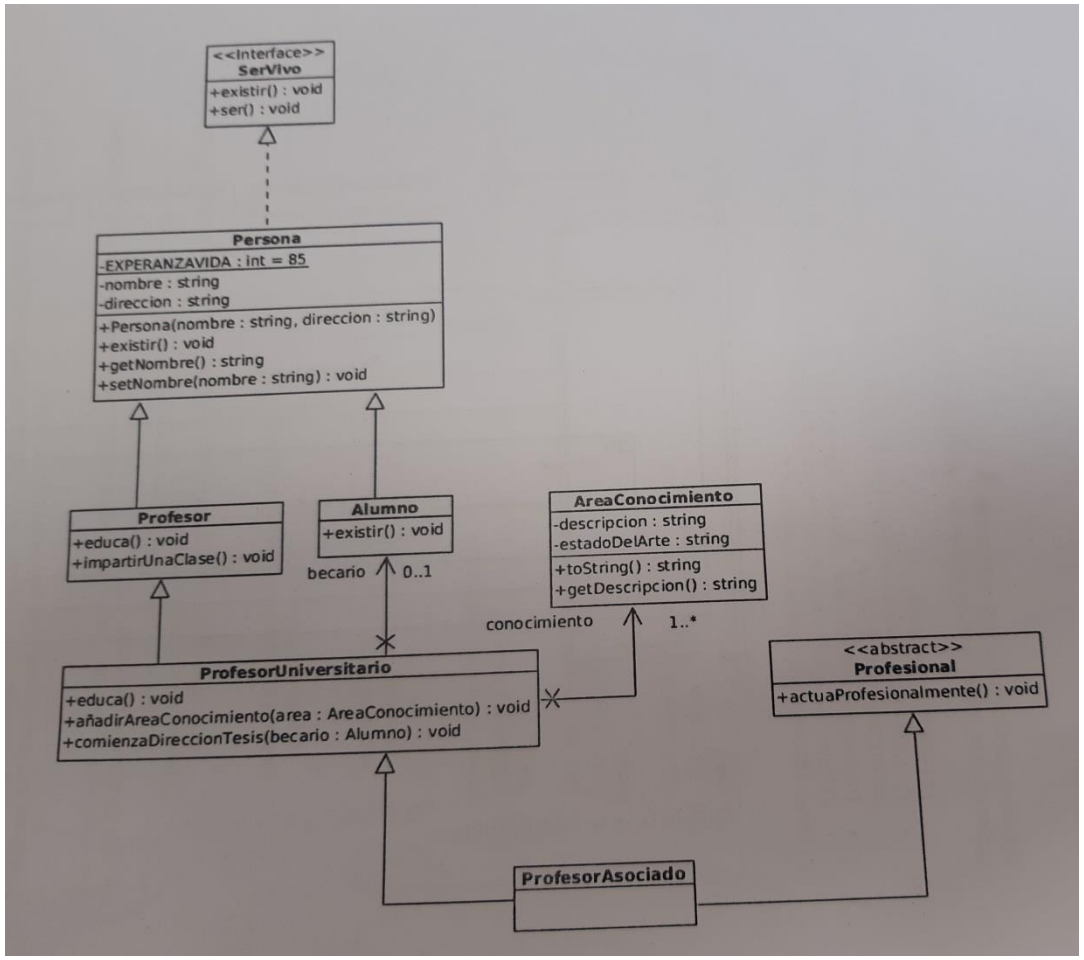
Ejercicio 2



Indica los errores de estos códigos en ProfesorUniversitario:

```
private void perdidaDeMemoria(){
   conocimiento = new ArrayList<>();
}

public void boicotea(ProfesorUniversitario p)

{
   p.perdidaDeMemoria();
}

public void cambiaNombre(){
   nombre="Dr. " + nombre;
}
------

private def perdidaDeMemoria
   @conocimiento = []
end

def boicoteda(prof)
  prof.perdidaDeMemoeria
End

def cambiaNombre
 @nombre = "Dr. " + @nombre
end
```

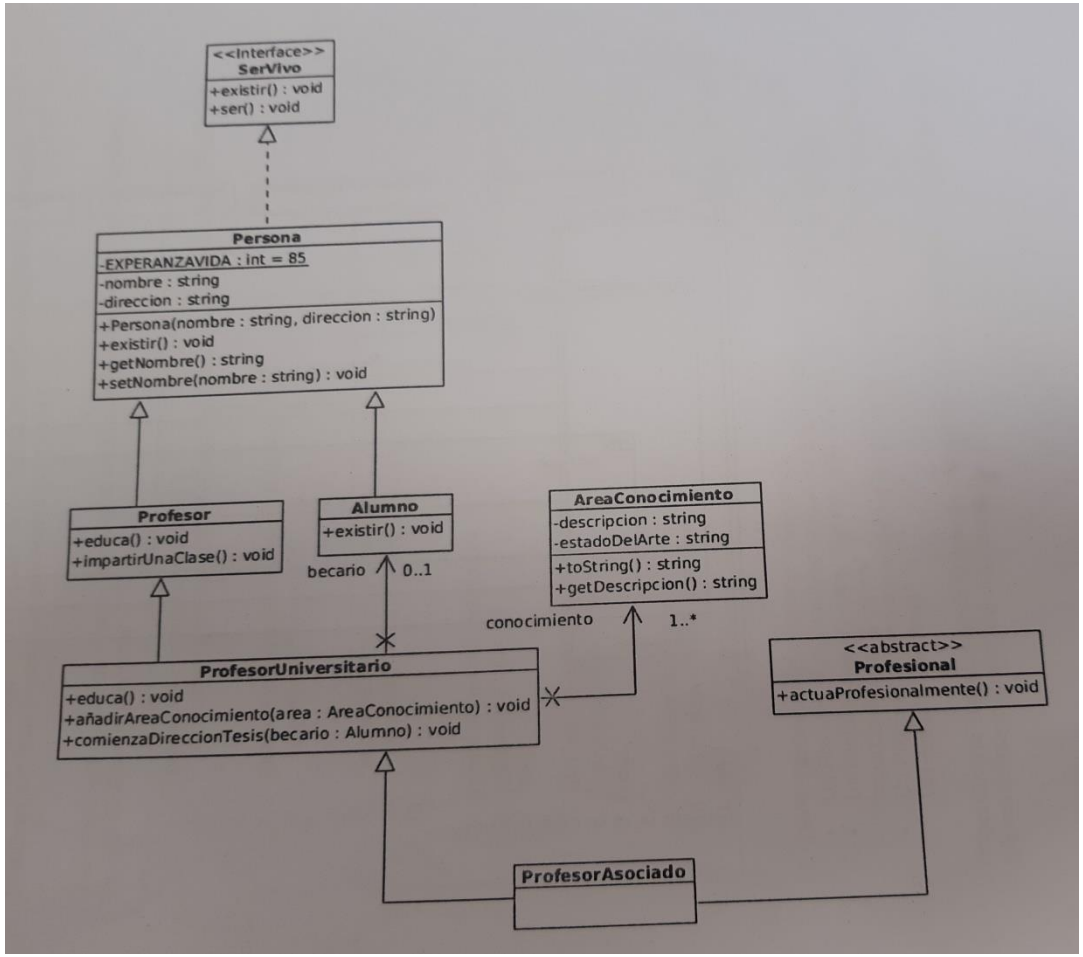
Ejercicio 2



Indica los errores de estos códigos en ProfesorUniversitario:

```
private void perdidaDeMemoria(){
   conocimiento = new ArrayList<>();
}

public void boicotea(ProfesorUniversitario p)

{
   p.perdidaDeMemoria();
}

public void cambiaNombre(){
   nombre="Dr. " + nombre;
}
------

private def perdidaDeMemoria
   @conocimiento = []
end

def boicoteda(prof)
  prof.perdidaDeMemoria
End

def cambiaNombre
 @nombre = "Dr. " + @nombre
end
```

Ejercicio 3

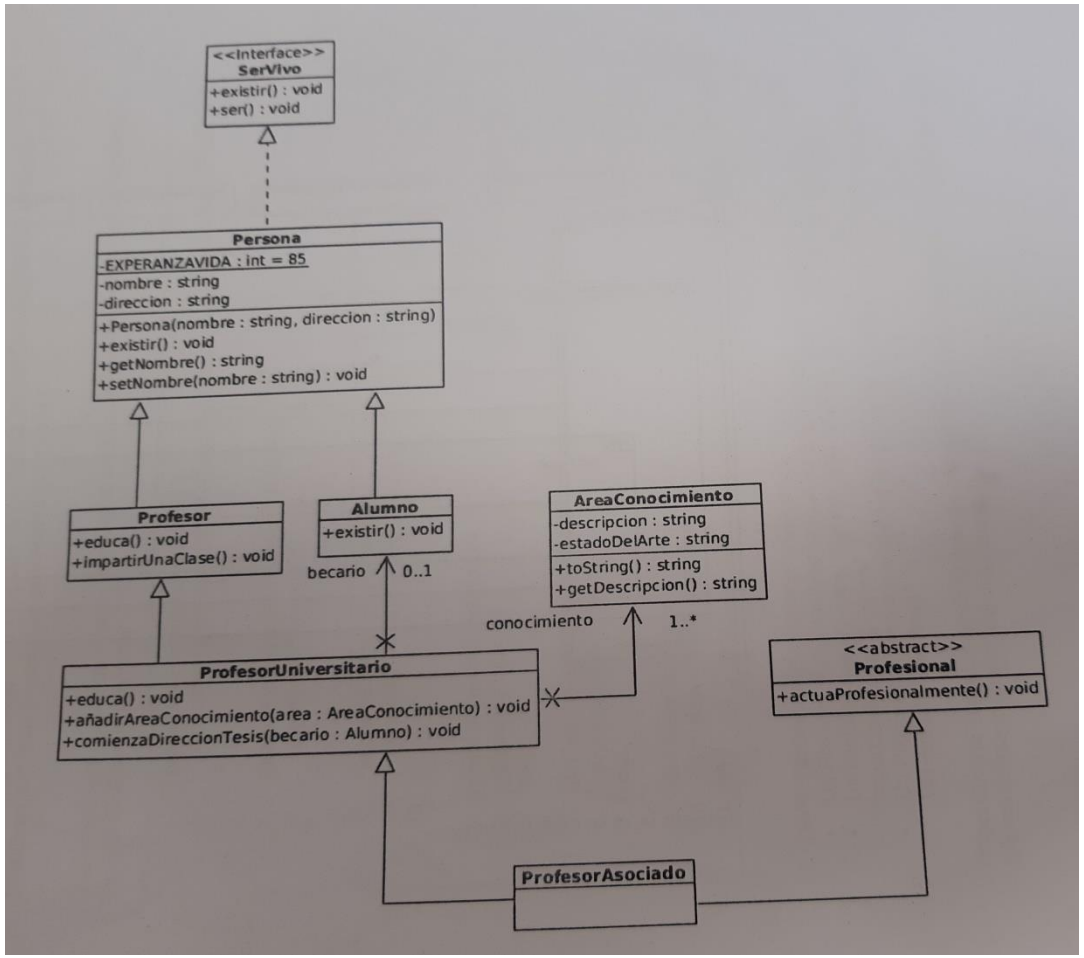


Implementa en Java el método educa e impartirUnaClase en todas las clases que sea necesario. Teniendo en cuenta que:

- Un profesor educa escribiendo por consola “Profesor educando”
- Un profesor imparte clase, educando primero y escribiendo por consola “Profesor impartiendo clase” después
- Un ProfesorUniversitario educa como un Profesor y además escribe en consola “Profesor Universitario educando”

```
//
private void perdidaDeMemoria(){
   conocimiento = new ArrayList<>();
}

public void boicotea(ProfesorUniversitario p)

{
   p.perdidaDeMemoria();
}

public void cambiaNombre(){
   nombre="Dr. " + nombre;
}
------

private def perdidaDeMemoria
   @conocimiento = []
end
```

Ejercicio 3

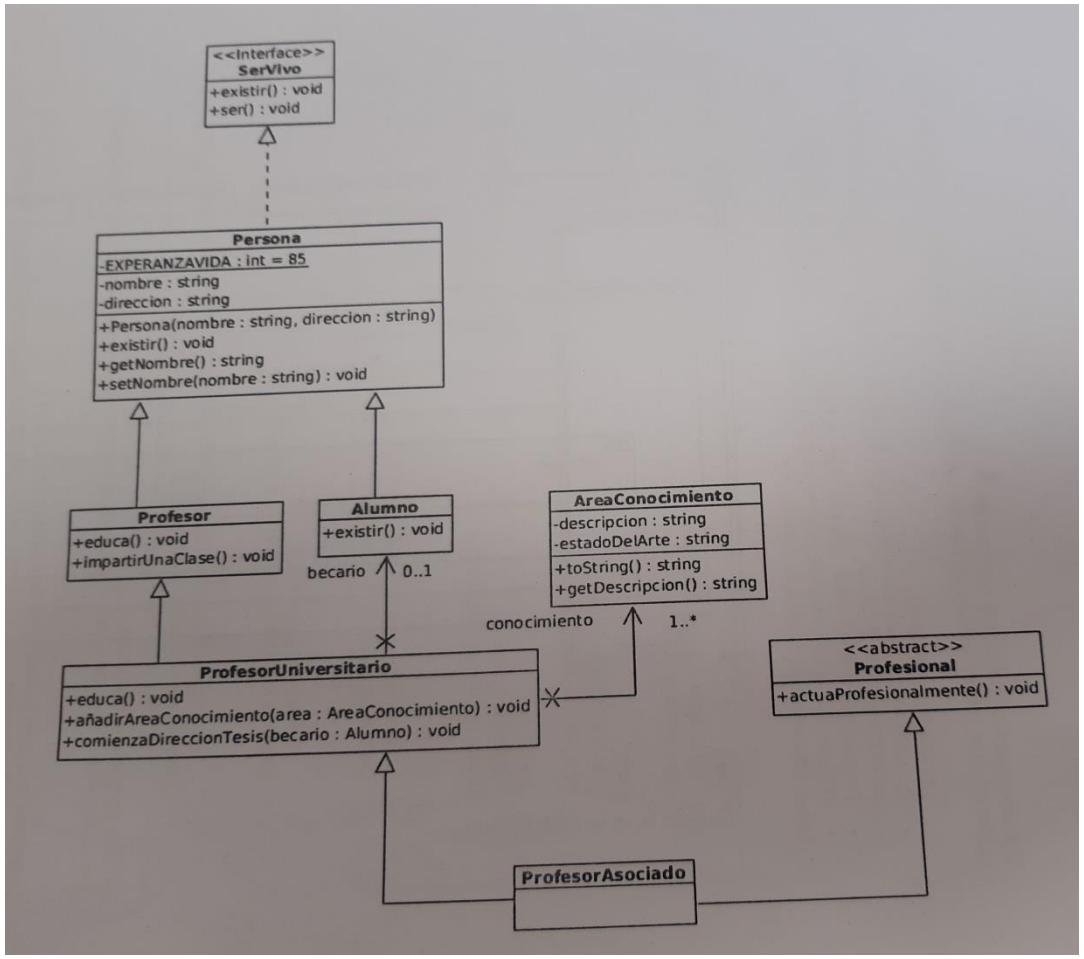


Implementa en Java el método educa e impartirUnaClase en todas las clases que sea necesario. Teniendo en cuenta que:

- Un profesor educa escribiendo por consola “Profesor educando”
- Un profesor imparte clase, educando primero y escribiendo por consola “Profesor impartiendo clase” después
- Un ProfesorUniversitario educa como un Profesor y además escribe en consola “Profesor Universitario educando”

```
//En Profesor

public void educa(){
  System.out.println(“Profesor educando”);
}

public void impartirUnaClase(){
  this.educa();
  System.out.println(“Profesor impartiendo
clase”);
}

//En ProfesorUniversitario
@Override
public void educa(){
   super.educa();
   System.out.println(“Profesor
Universitario educando”);
}
```

Ejercicio 4

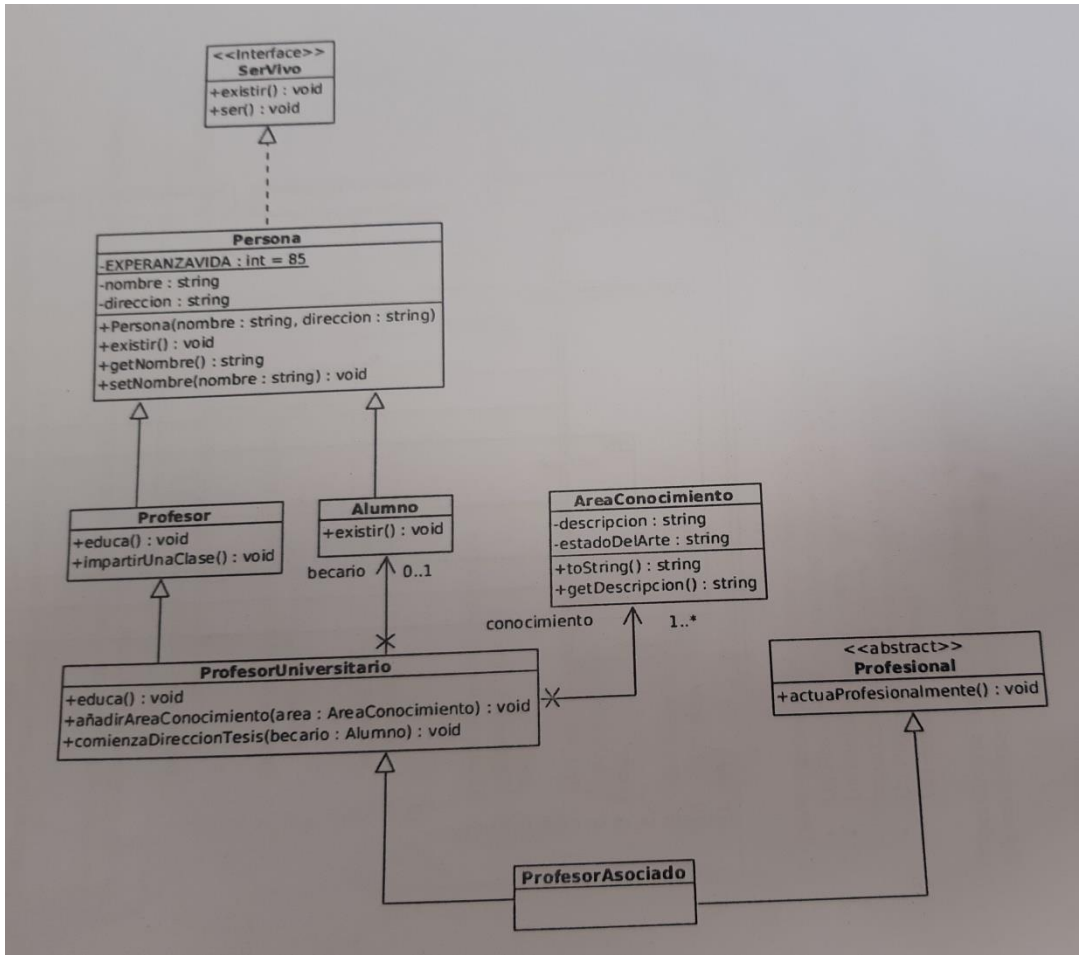



```
//En Profesor

public void educa(){
  System.out.println("Profesor educando");
}

public void impartirUnaClase(){
  this.educa();
  System.out.println("Profesor impartiendo
clase");
}

//En ProfesorUniversitario
@Override
public void educa(){
   super.educa();
   System.out.println("Profesor
Universitario educando");
}
```

¿Cuál es la salida del siguiente código?

```
Profesor prof1 = new ProfesorUniversitario(…);

pro1.impartirUnaClase();

((Profesor) prof1).impartirUnaClase();

((ProfesorUniversitario) prof1).impartirUnaClase();
```

Ejercicio 4



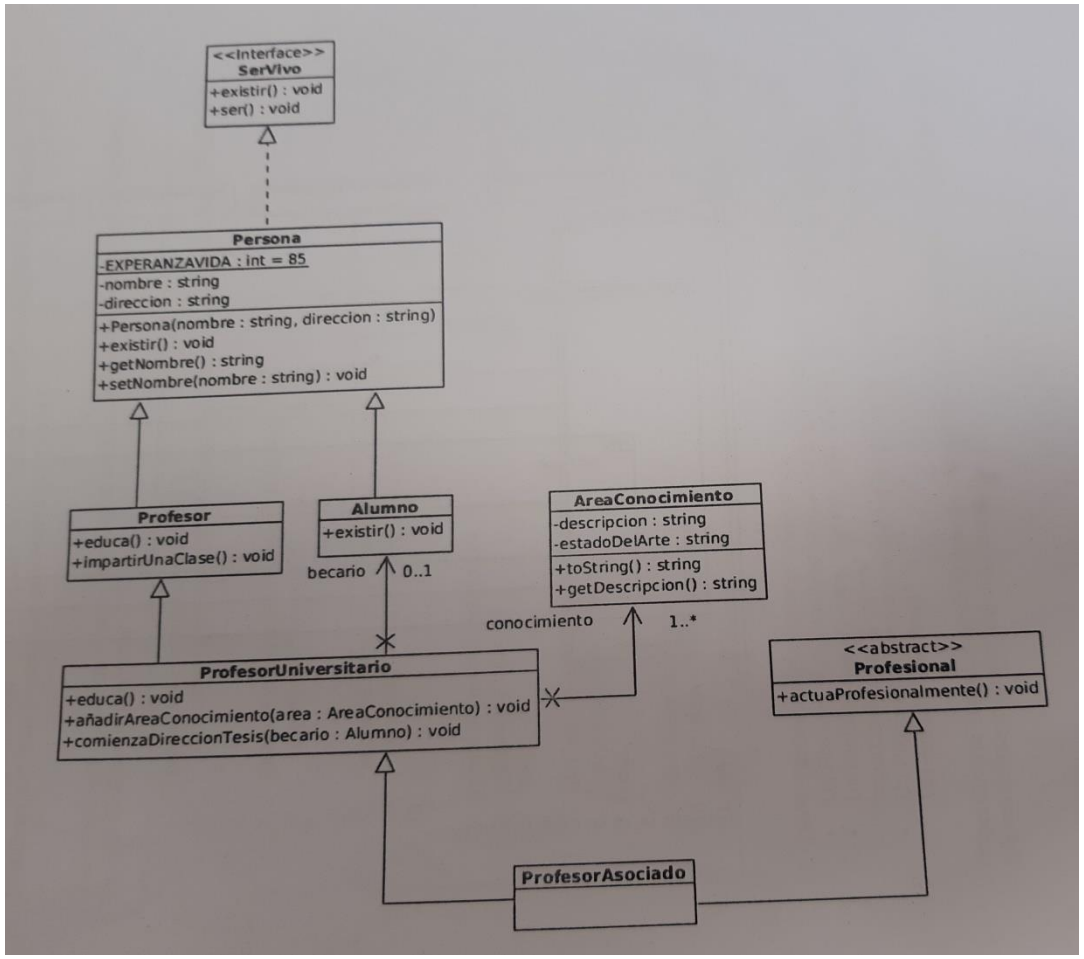

```
//En Profesor

public void educa(){
  System.out.println("Profesor educando");
}

public void impartirUnaClase(){
  this.educa();
  System.out.println("Profesor impartiendo
clase");
}

//En ProfesorUniversitario
@Override
public void educa(){
   super.educa();
   System.out.println("Profesor
Universitario educando");
}
```

¿Cuál es la salida del siguiente código?

```
Profesor prof1 = new ProfesorUniversitario(…);

prof1.impartirUnaClase();
```

Profesor educando
Profesor universitario educando
Profesor impartiendo clase

Ejercicio 4




```
//En Profesor

public void educa(){
  System.out.println("Profesor educando");
}

public void impartirUnaClase(){
  this.educa();
  System.out.println("Profesor impartiendo
clase");
}

//En ProfesorUniversitario
@Override
public void educa(){
   super.educa();
   System.out.println("Profesor
Universitario educando");
}
```

¿Cuál es la salida del siguiente código?

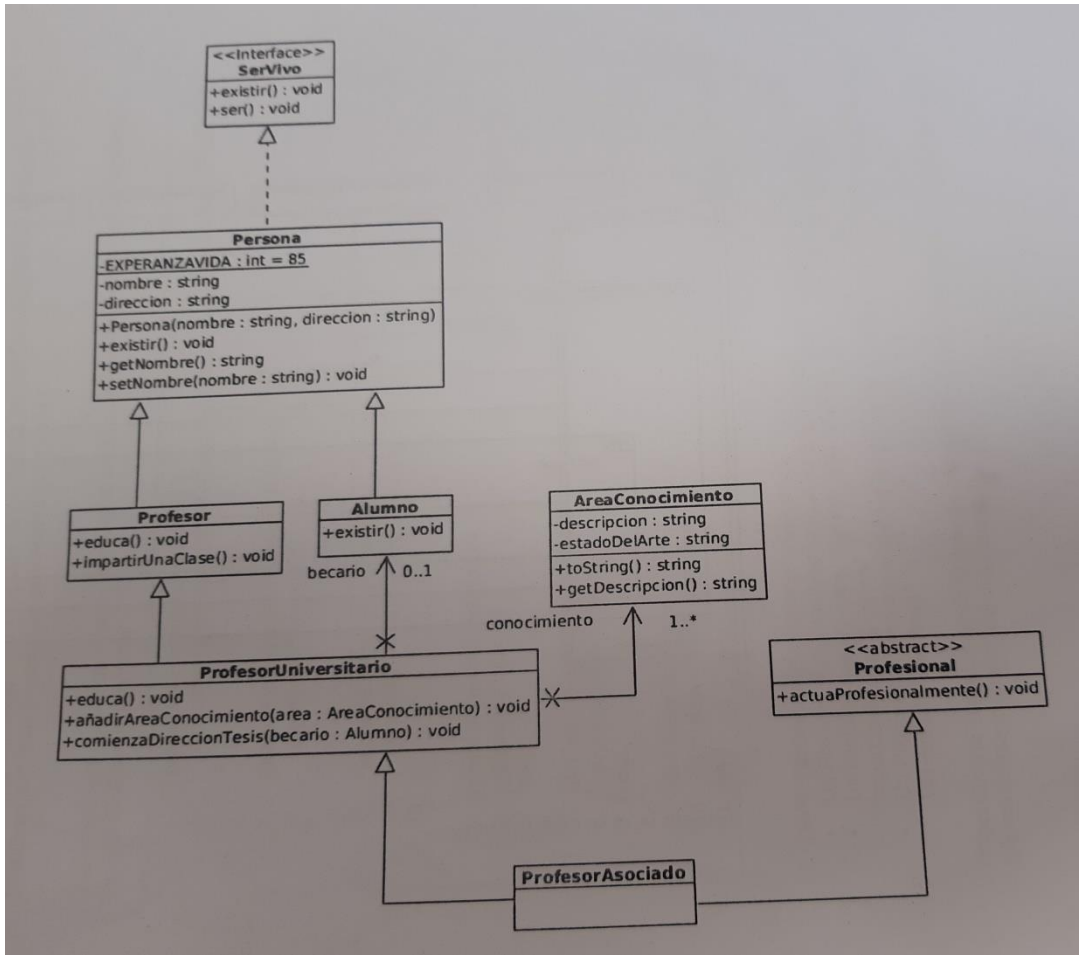
Profesor prof1 = new ProfesorUniversitario(…);

((Profesor) prof1).impartirUnaClase();

Igual

Profesor educando
Profesor universitario educando
Profesor impartiendo clase

Ejercicio 4




```
//En Profesor

public void educa(){
  System.out.println("Profesor educando");
}

public void impartirUnaClase(){
  this.educa();
  System.out.println("Profesor impartiendo
clase");
}

//En ProfesorUniversitario
@Override
public void educa(){
   super.educa();
   System.out.println("Profesor
Universitario educando");
}
```

¿Cuál es la salida del siguiente código?

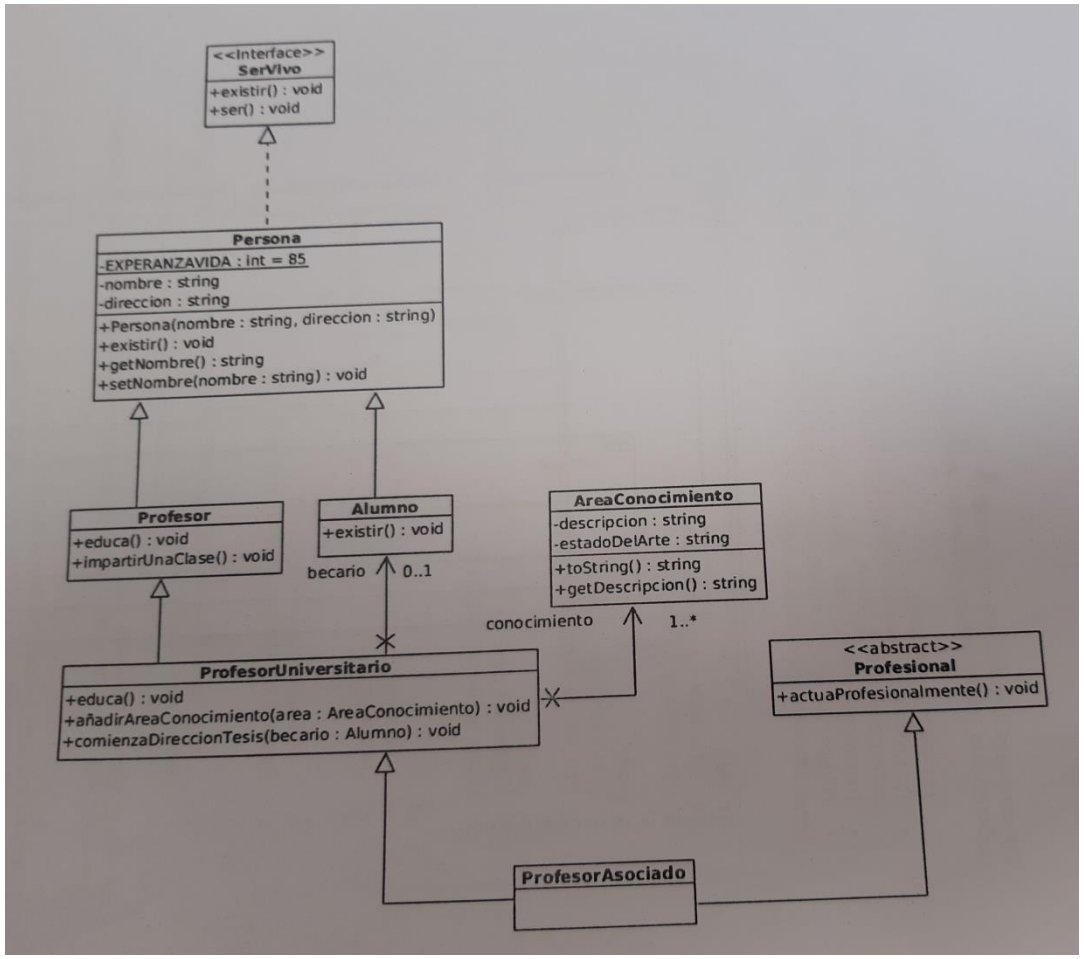
Profesor prof1 = new ProfesorUniversitario(…);

((ProfesorUniversitario) prof1).impartirUnaClase();

Igual

Profesor educando
Profesor universitario educando
Profesor impartiendo clase

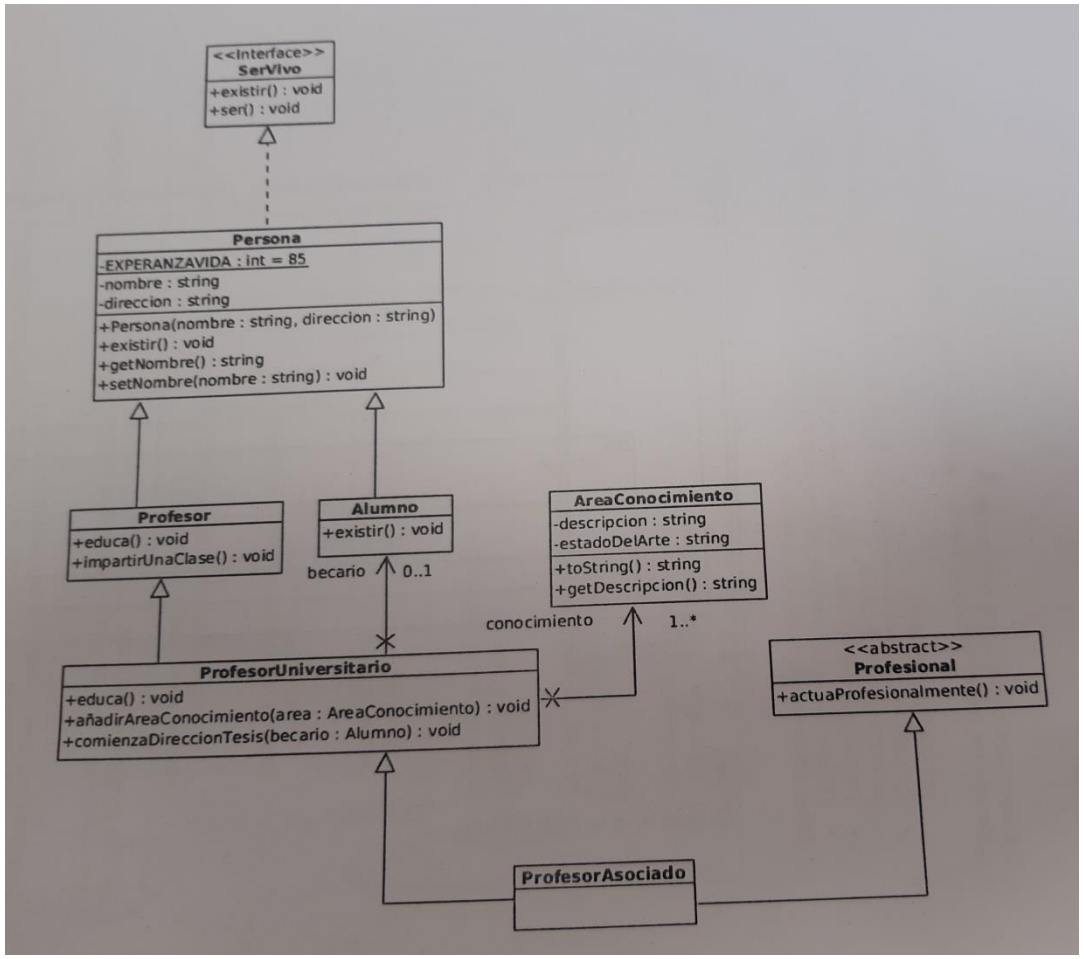
Ejercicio 5



Indica los errores y el tipo de los mismos en el siguiente código Java:

```
AreaConocimiento a4 = new
AreaConocimiento("Informática",
"Turing");

Profesor profUni1 = new
ProfesorUniversitario(…);

Profesor prof3 = new Profesor(…);

profUni1.añadirAreaConocimiento(a4);

ProfesorUniversitario profUni2 =
profUni1;

ProfesorUniversitario profUni3 = prof3;
Alumno alum1 = (Alumno) ((Persona)
prof3);
```

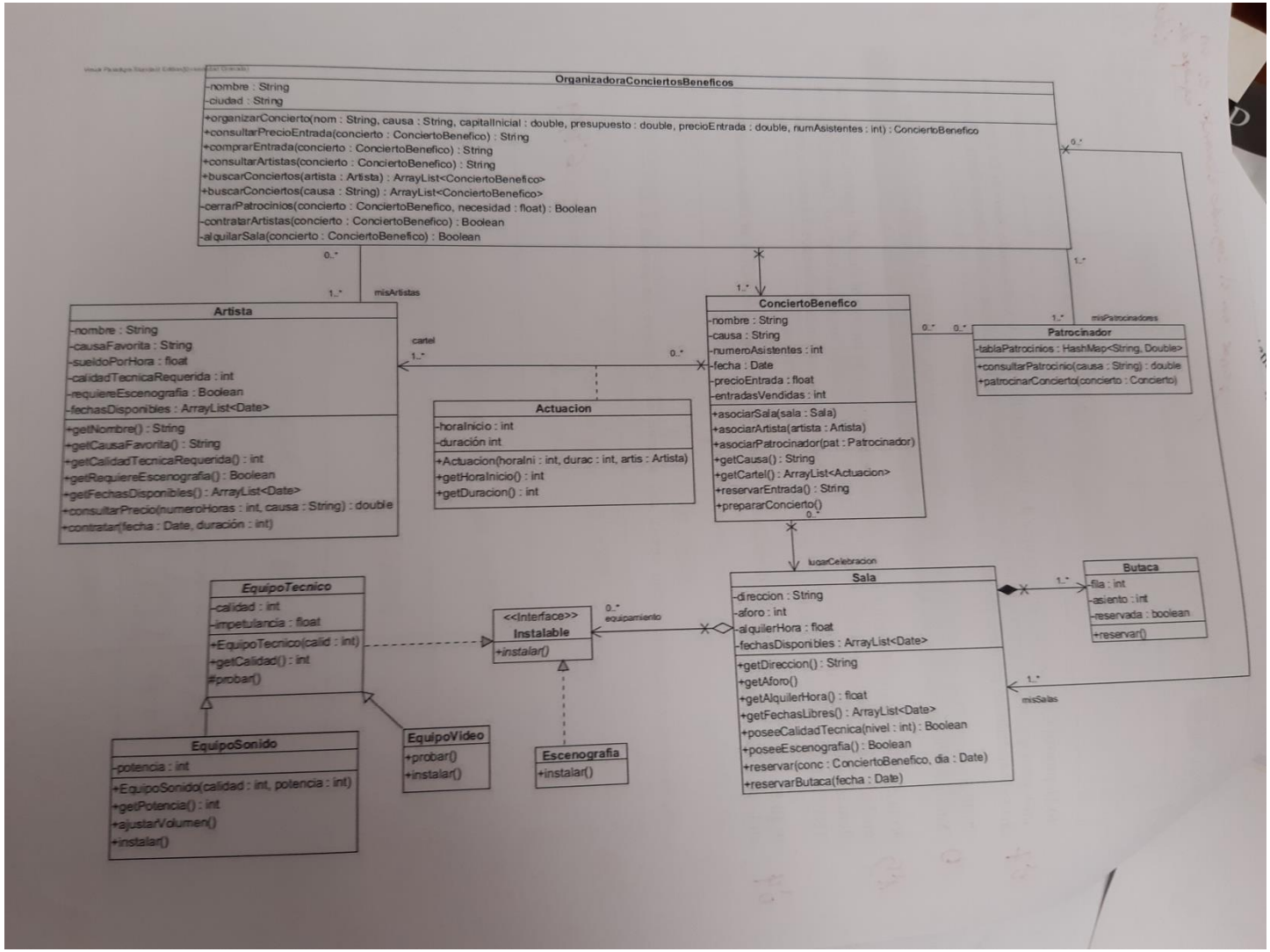
Ejercicio 5



Indica los errores y el tipo de los mismos en el siguiente código Java:

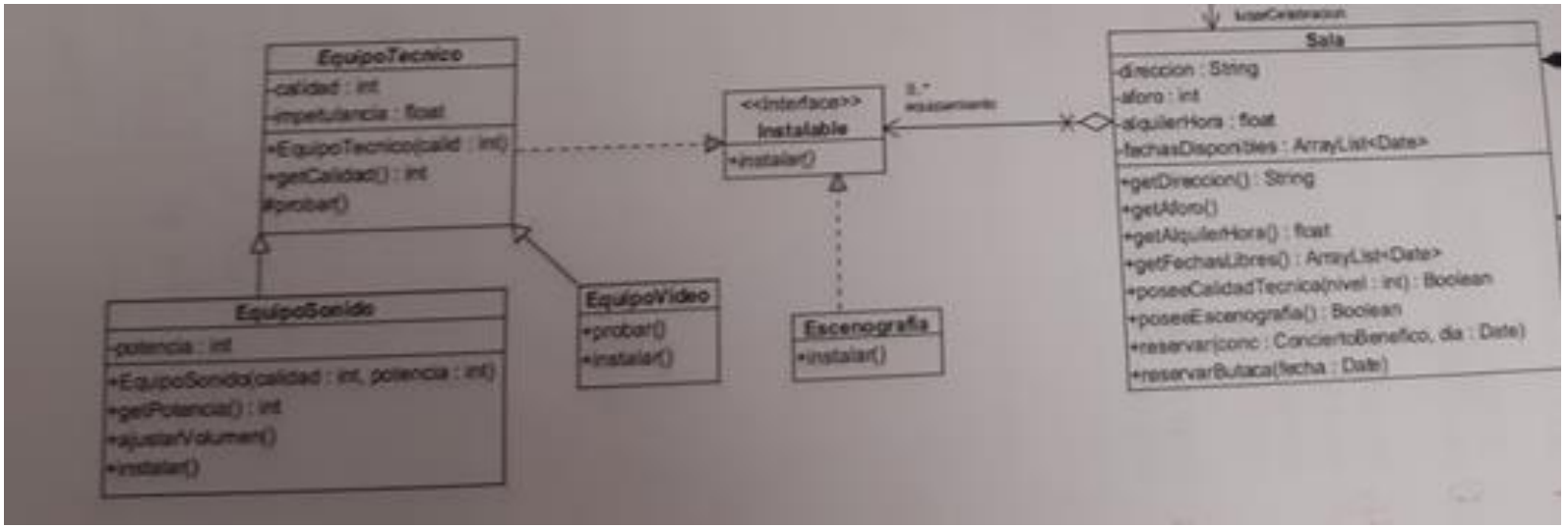
```
AreaConocimiento a4 = new
AreaConocimiento("Informática",
"Turing");

Profesor profUni1 = new
ProfesorUniversitario(…);

Profesor prof3 = new Profesor(…);

((ProfesorUniversitario)profUni1).añadi
rAreaConocimiento(a4);

ProfesorUniversitario profUni2 =
(ProfesorUniversitario)profUni1;

~~ProfesorUniversitario profUni3 = prof3;~~
//Error de compilación

~~Alumno alum1 = (Alumno) ((Persona)~~
~~prof3);~~
//Error de ejecución
```

Ejercicio 6



Implementa en Java la clase EquipoTecnico y la clase EquipoSonido:

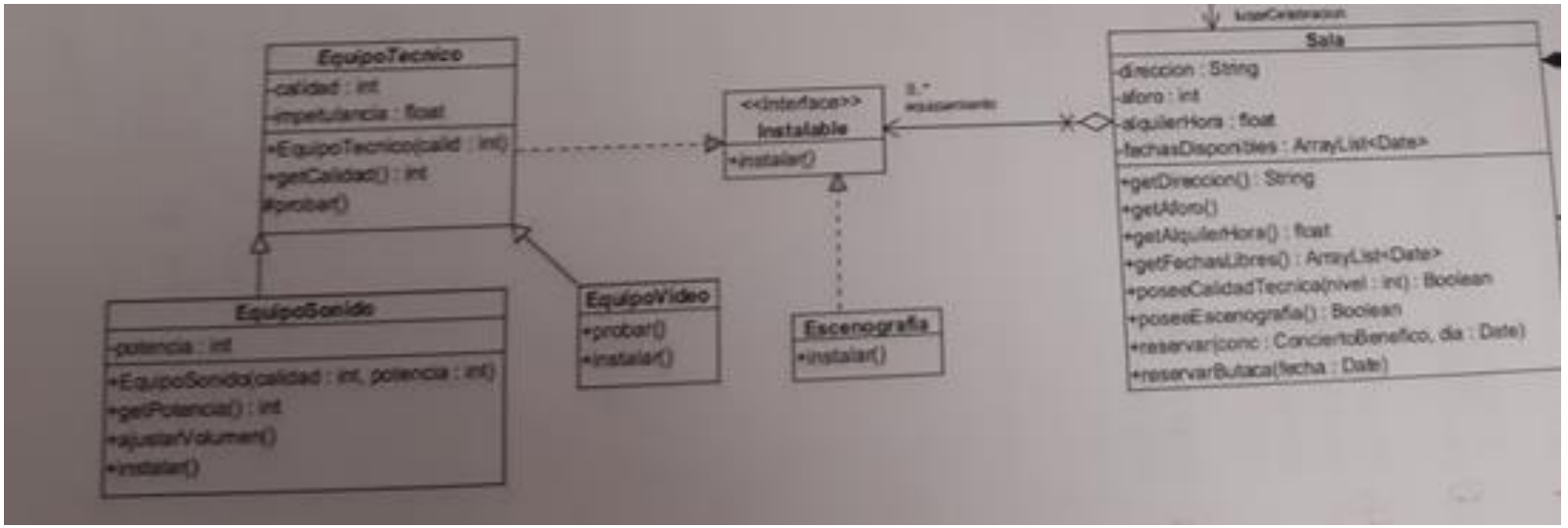
Ejercicio 6



Implementa en Java la clase EquipoTecnico y la clase EquipoSonido:

```
public class EquipoTecnico implements Instalable{
  private int calidad;
  private floar impetulancia;
  public EquipoTecnico(int calidad){…}
  public getCalidad(){…}
  protected void probar(){…}

}
```

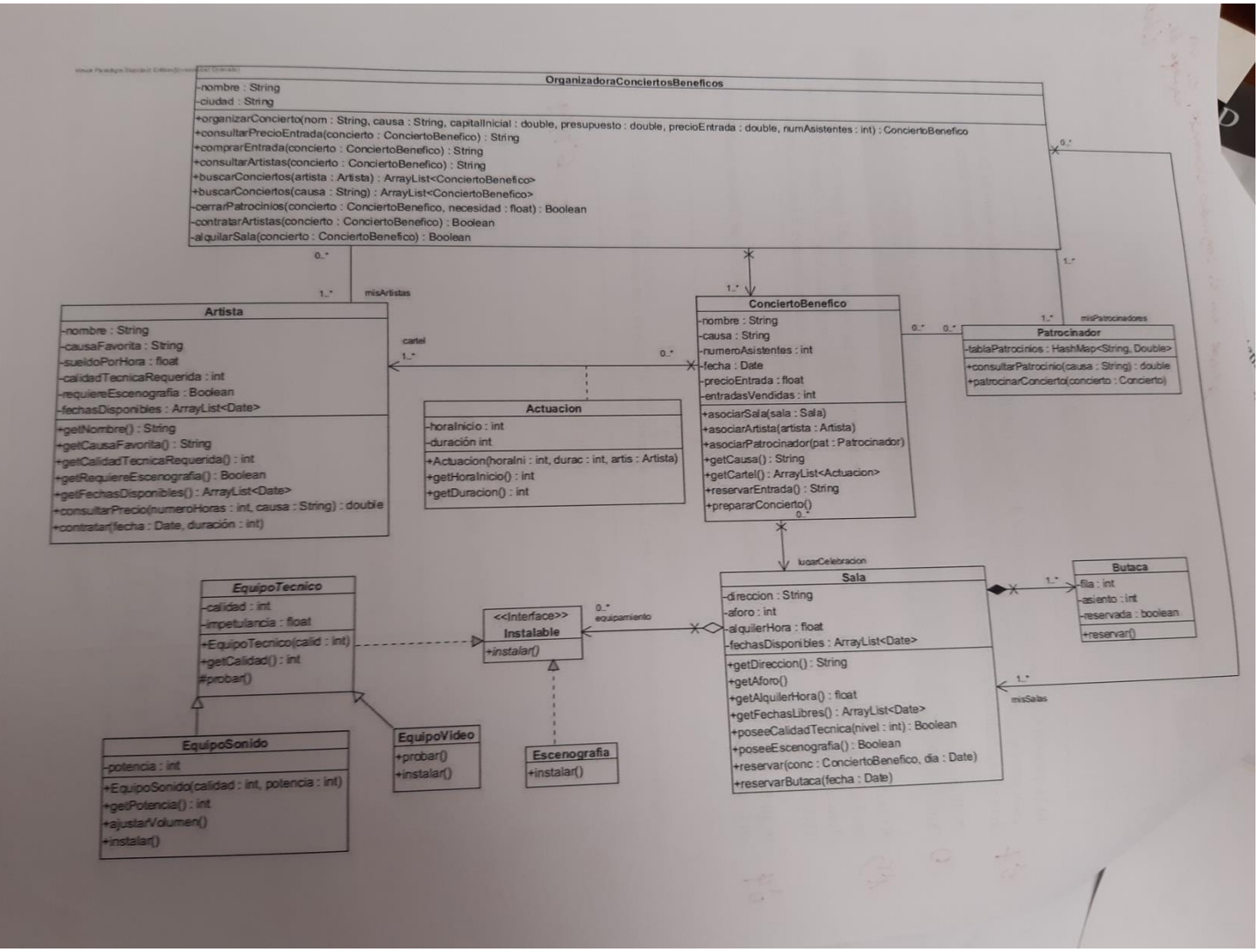
Ejercicio 6



Implementa en Java la clase EquipoTecnico y la clase EquipoSonido:

```
public class EquipoSonido extends EquipoTecnico{
  private int potencia;

  public EquipoSonido(int calidad, int pot){
    super(calidad);
    potencia = pot;
  }
  public int getPotencia() { return potencia;}
  public void ajustarVolumen() {…}
  @Override
  public void instalar(){…}
}
```

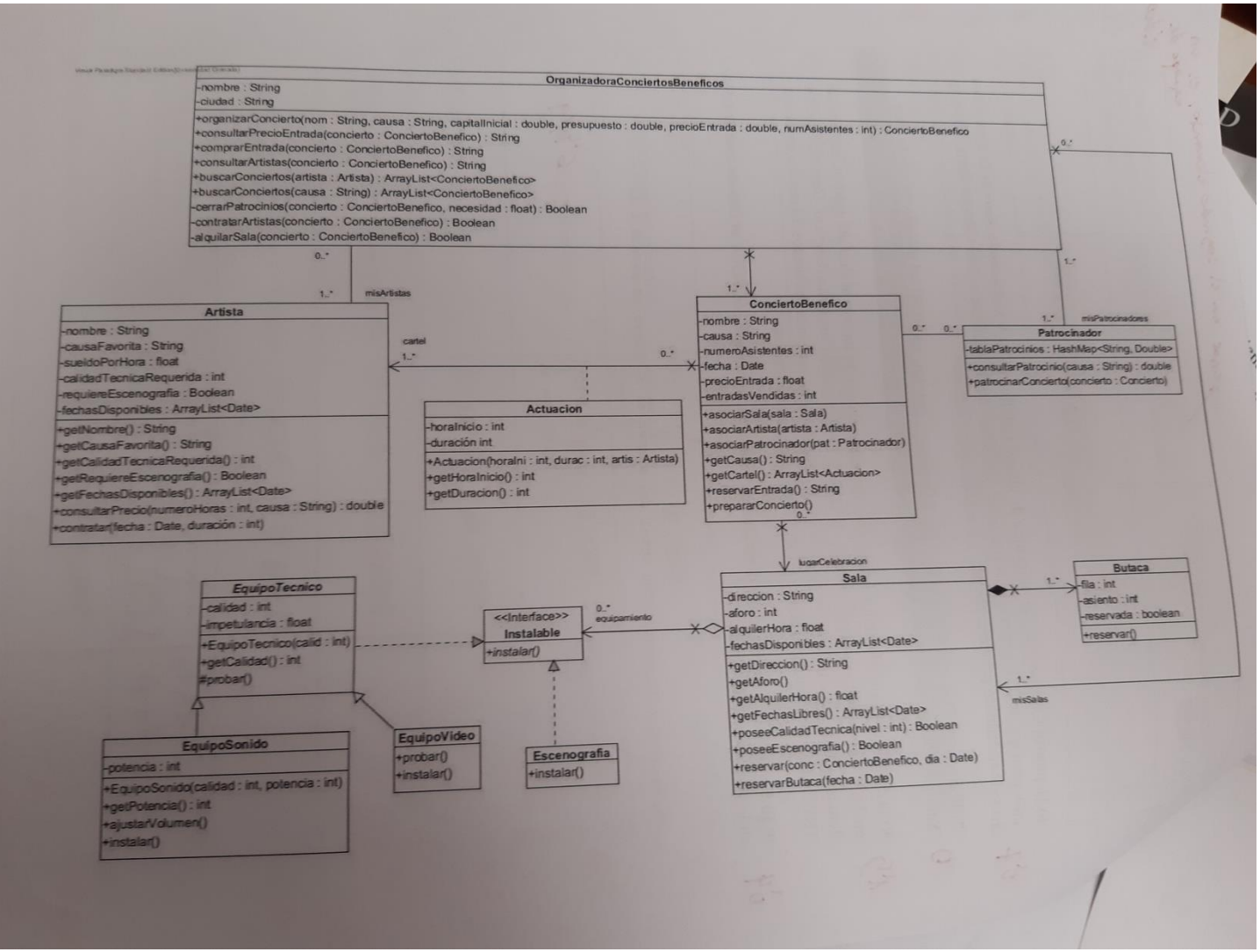
Ejercicio 7



Indica los errores en el código:

```
Instalable i1, i2, i3;

i1 = new Instalable();

i2 = new EquipoSonido(300);

i3 = new EquipoSonido(9,
7.8, 300);

i3.ajustarVolumen();

EquipoTecnico eV = new
EquipoVideo (9, 5.3);

ArrayList<EquipoTecnico>
equipos = new ArrayList();

equipos.add(i3);
equipos.add(eV);

for (EquipoTecnico
e:equipos){

  e.getCalidad();
((EquipoSonido)
e).adjustarVolumen();
}
```

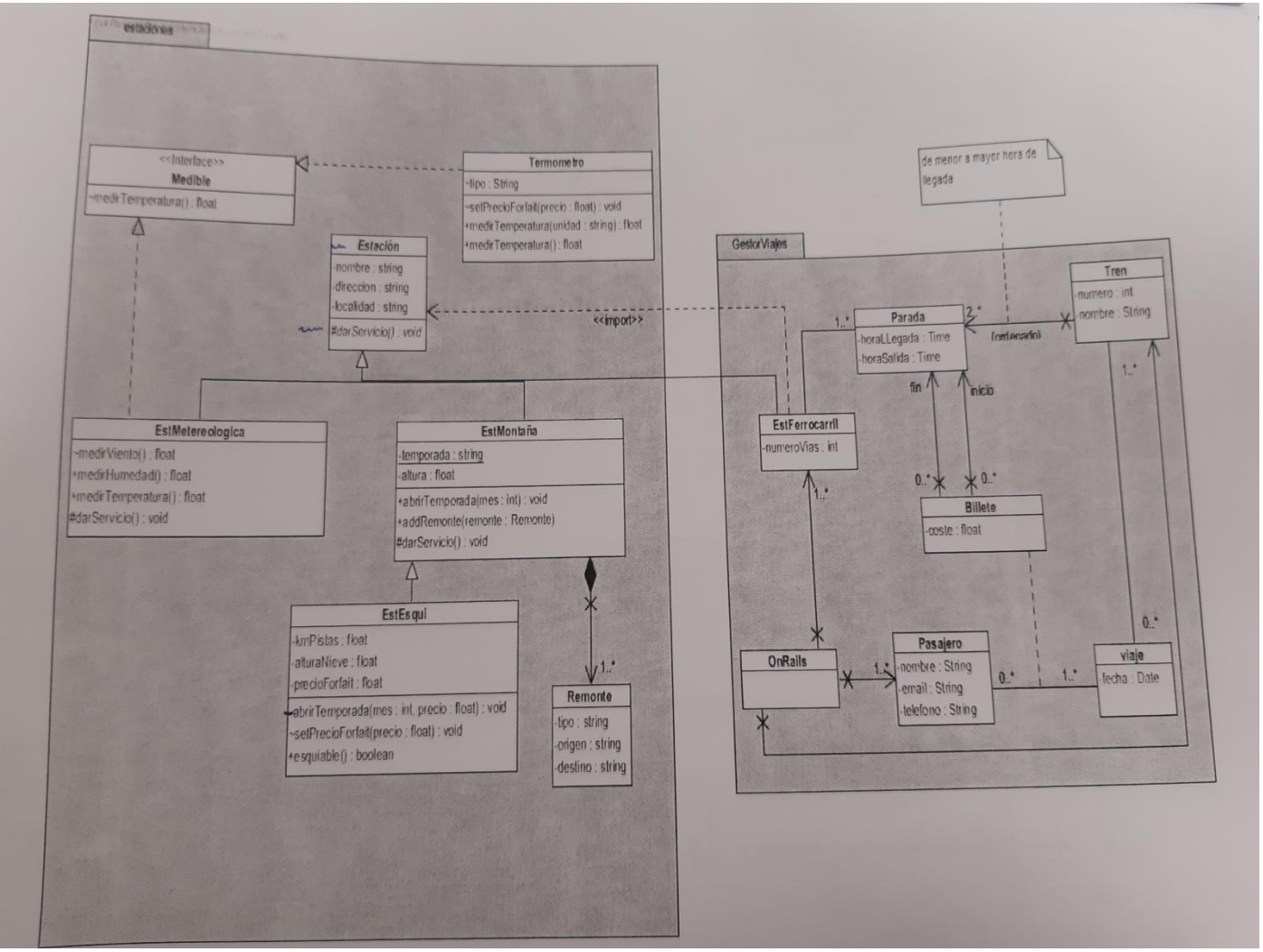
Ejercicio 7



Indica los errores en el código:

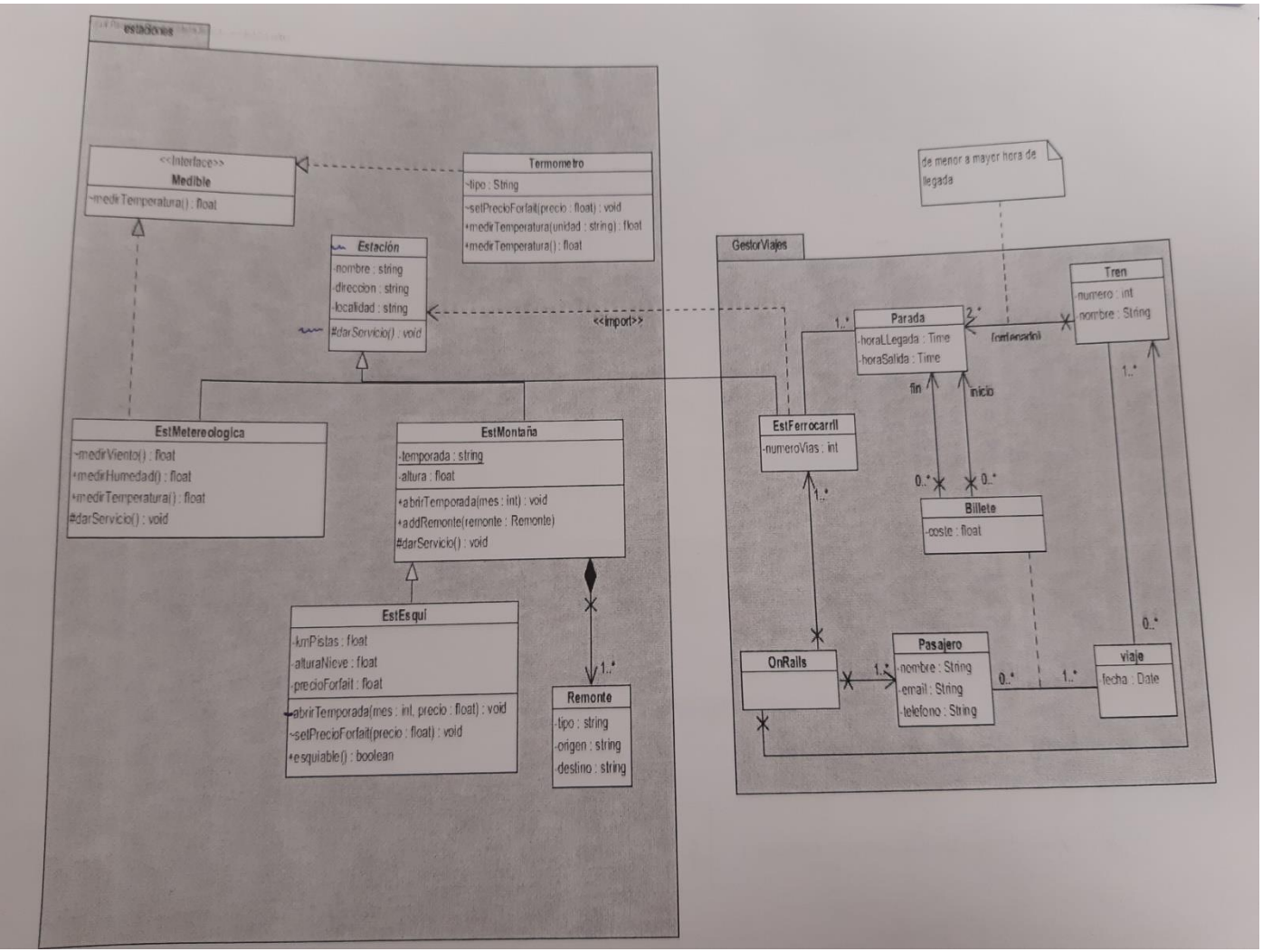
```
Instalable i1, i2, i3;

i1 = new Instalable();

i2 = new EquipoSonido(300,
5, 7);

i3 = new EquipoSonido(9,
7.8, 300);

((EquipoSonido)
i3).ajustarVolumen();

EquipoTecnico eV = new
EquipoVideo (9, 5.3);

ArrayList<EquipoTecnico>
equipos = new ArrayList();

equipos.add( (EquipoSonido)
i3);
equipos.add(eV);

for (EquipoTecnico
e:equipos){

  e.getCalidad();
((EquipoSonido)
e).adjustarVolumen();
//Error de ejecución
}
```

Ejercicio 8



Implementa en Ruby la clase EstMontaña:

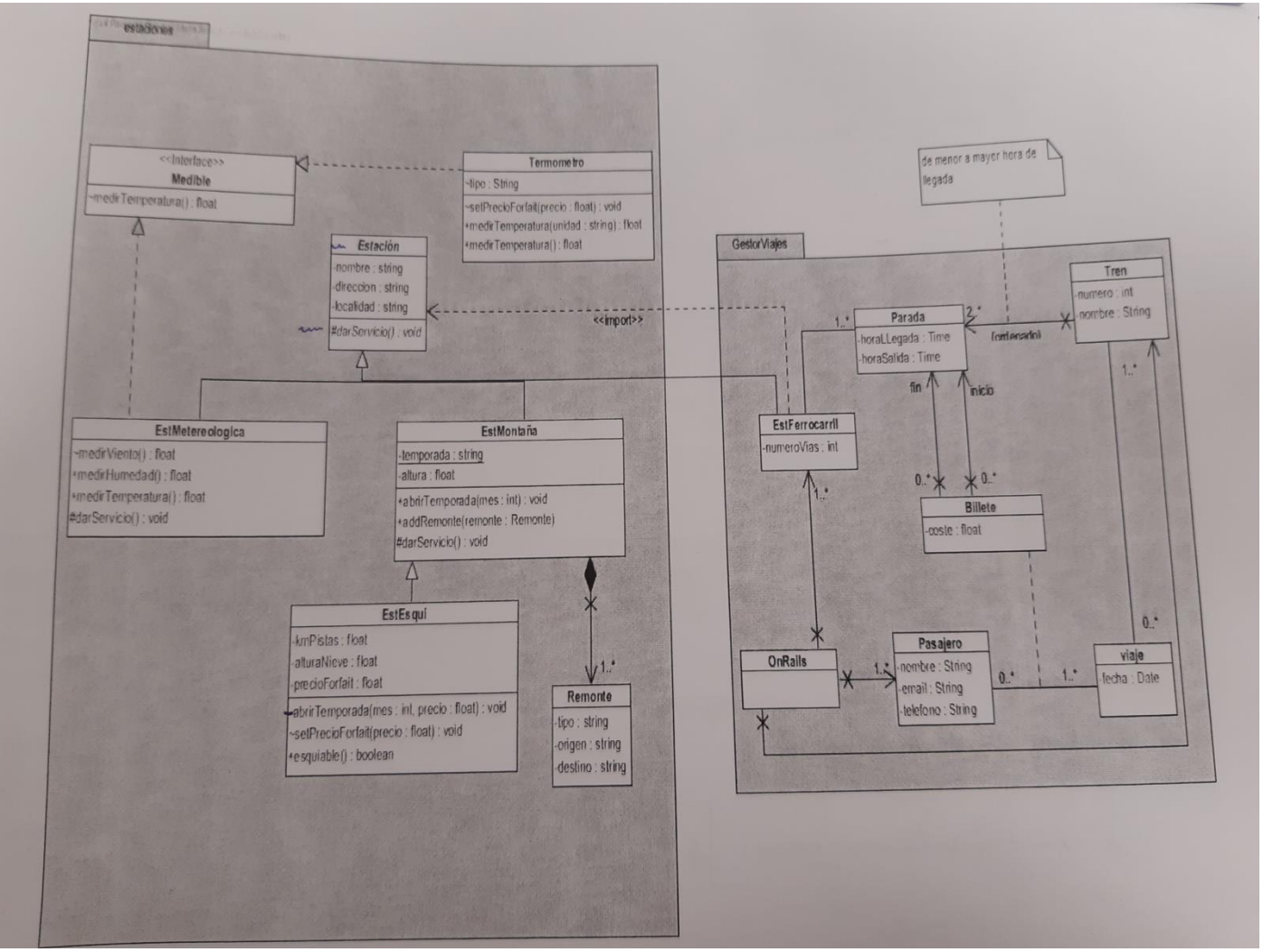
Ejercicio 8



Implementa en Ruby la clase EstMontaña:

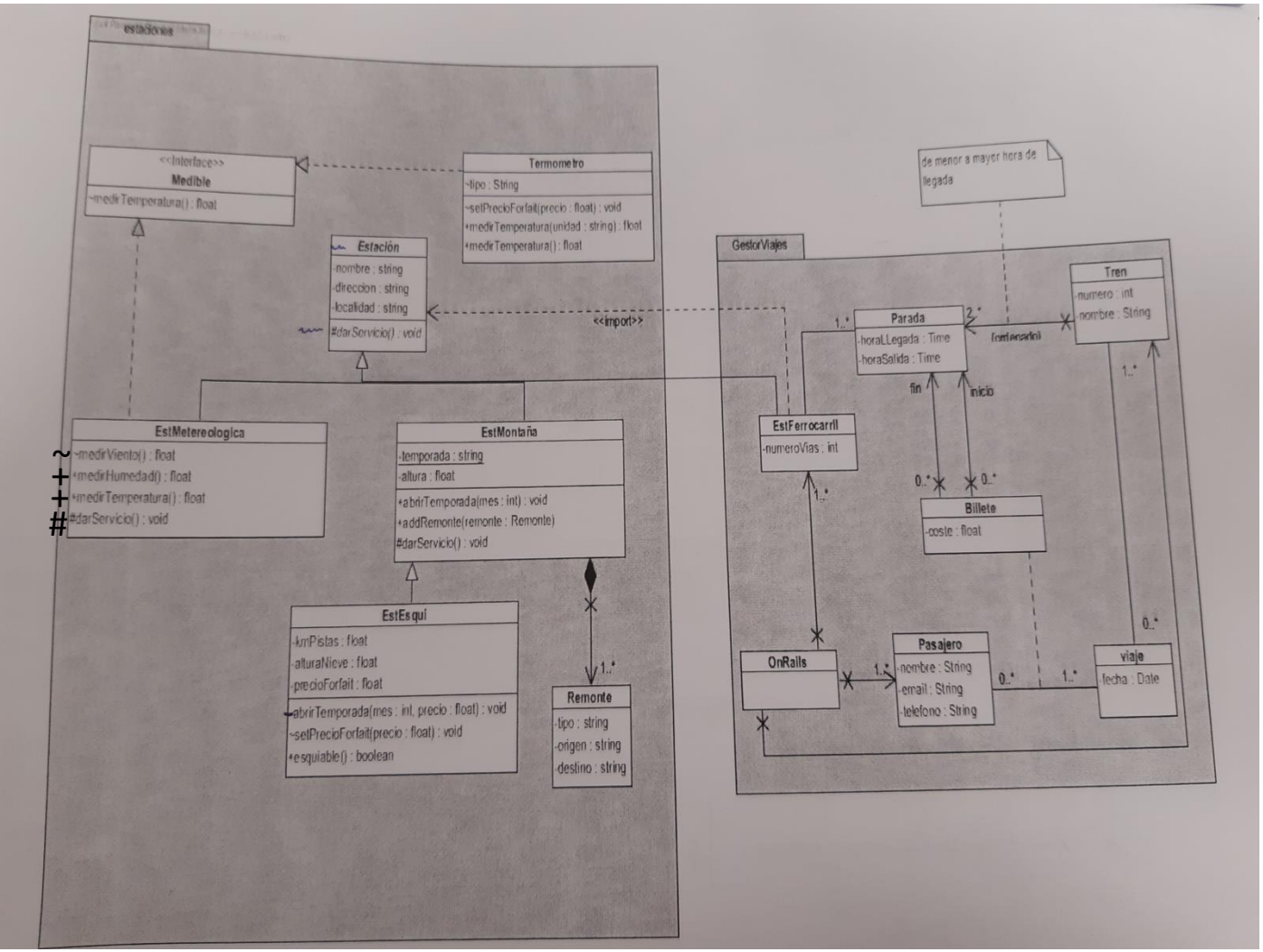
```
requite_relative
´estacion.rb´

module Estaciones

class EstMontaña <
Estacion

@@temporada
def initialize(n, d, l,
al)
  @altura = al
  super(n,d,l)
end

def abrirTemporada(mes)
   ….
end

def addRemonte (remonte)
 ….
end

protected
def darServicio()
 ….
end

end

end
```

Ejercicio 9



Implementa en Java la clase EstMetereologica:

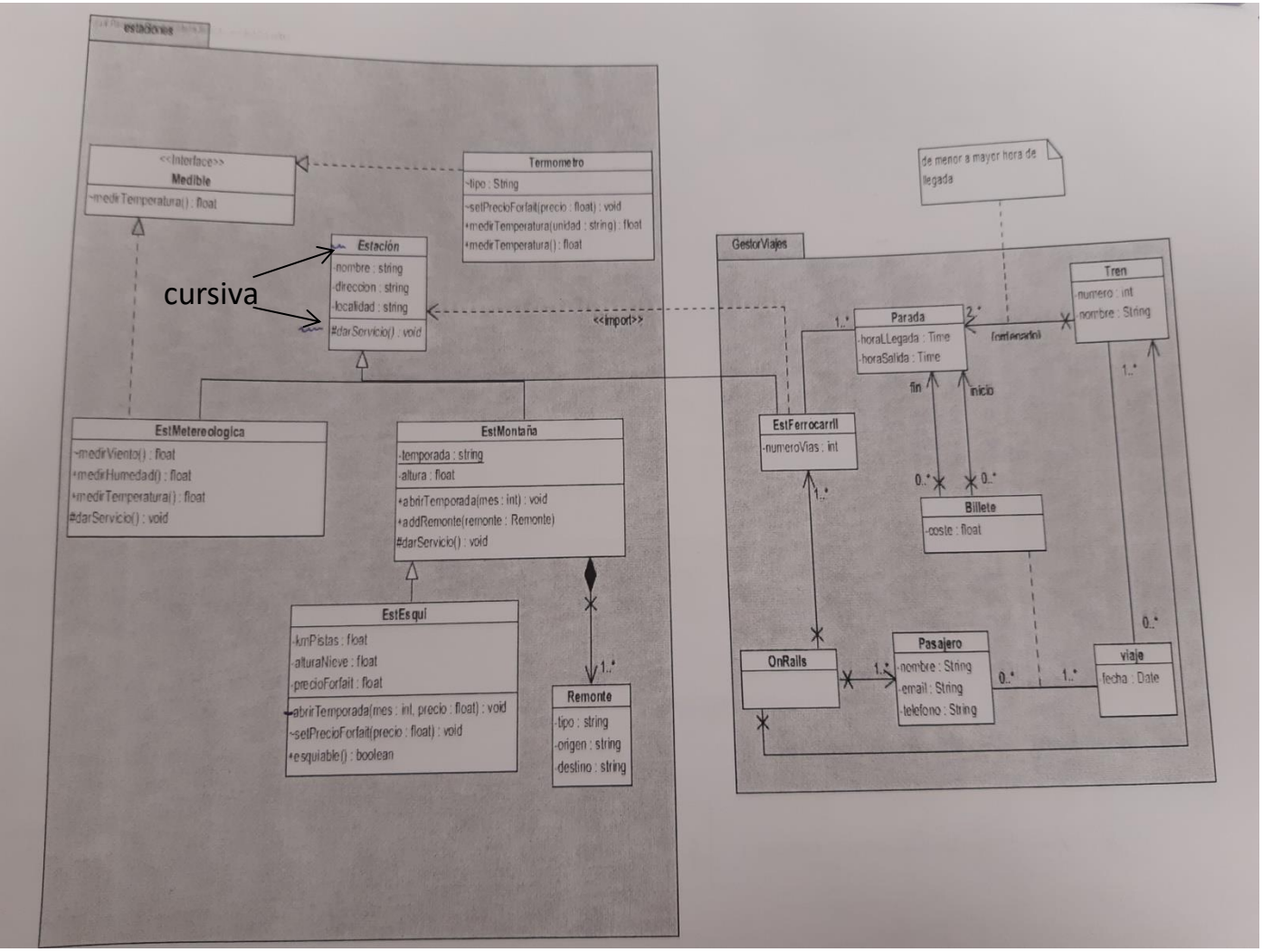
Ejercicio 9



Implementa en Java la clase EstMetereologica:

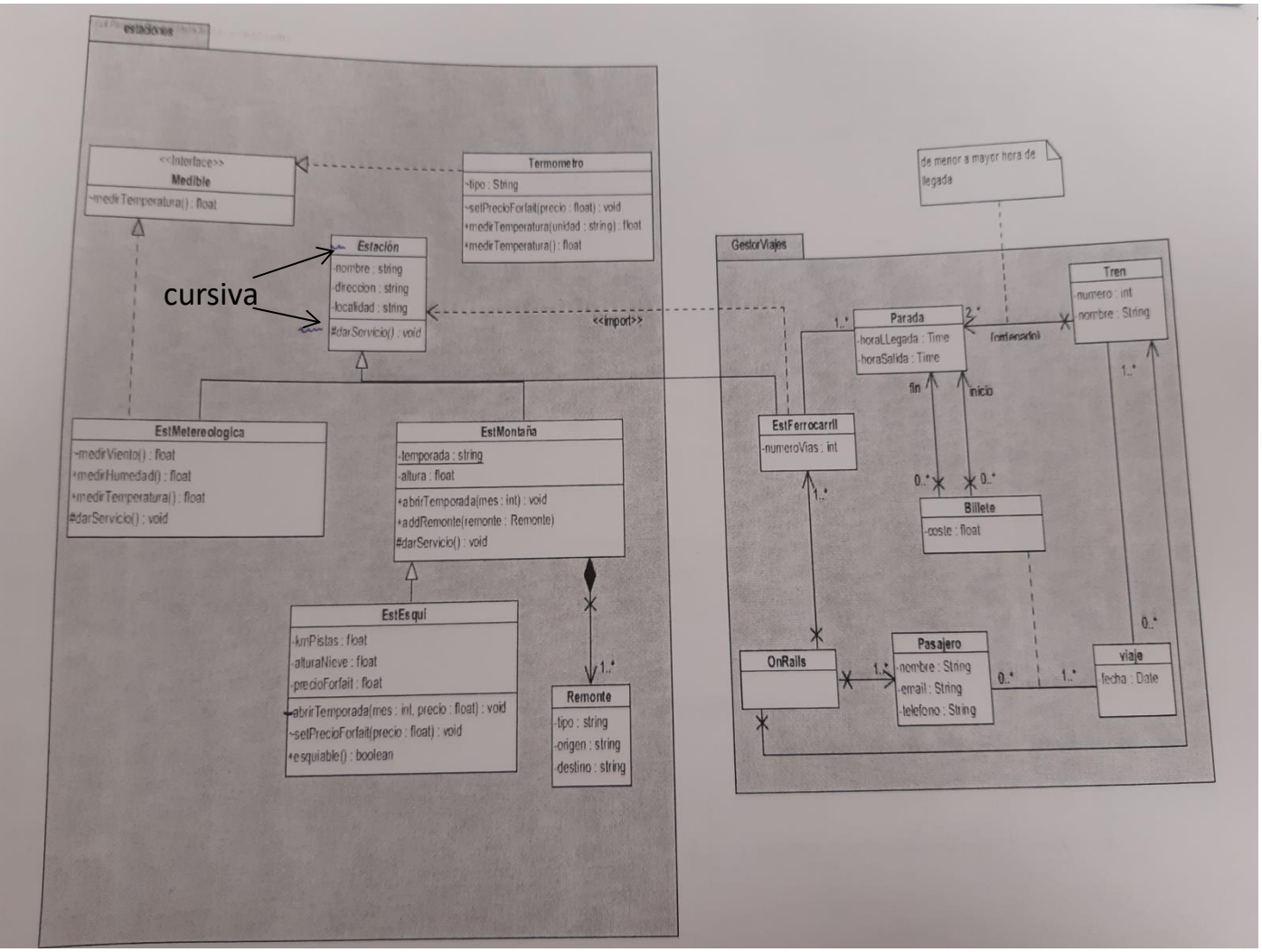
```
package estaciones;

class EstMetereologica
extends Estacion implements
Medible{

public estMetereologica(
String n, String d, String l){
  super(n, d, l);
}

float medirViento(){…}

public float
medirHumedad(){…}

@Override
public float
medirTemperatura(){…}

@Override
protected void
darServicio(){…}



}
```

Ejercicio 10



Implementa en Java la clase Estación:

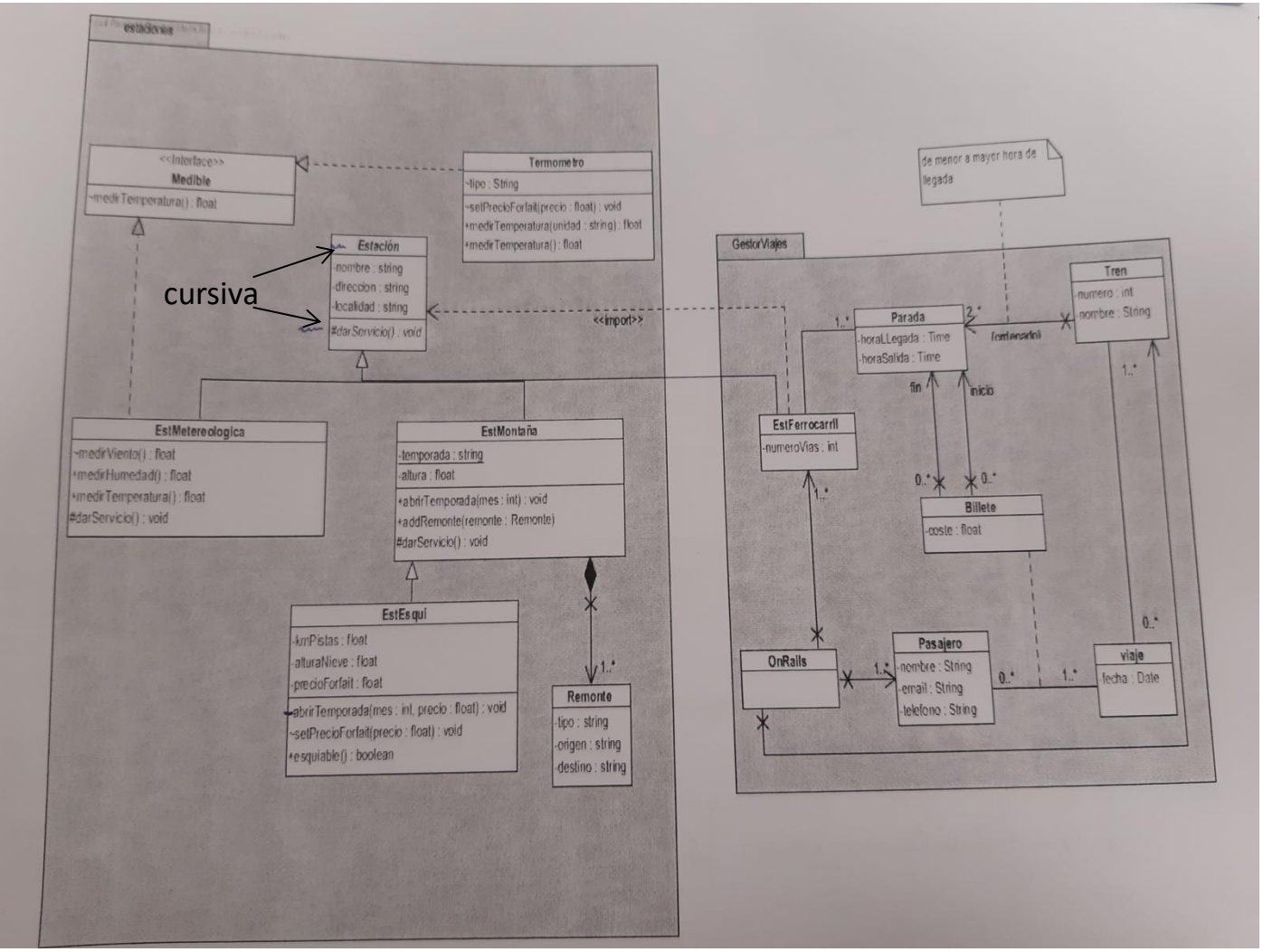
Ejercicio 10



Implementa en Java la clase Estación:

```
package estaciones;

class abstract Estacion{

  private String nombre;
  private String direccion;
  private String localidad;

  public Estacion(String n,
Strind d, string l){

  nombre = n;
  direccion = d;
  localidad = l;
  }

  protected abstract void
darServicio();


}
```

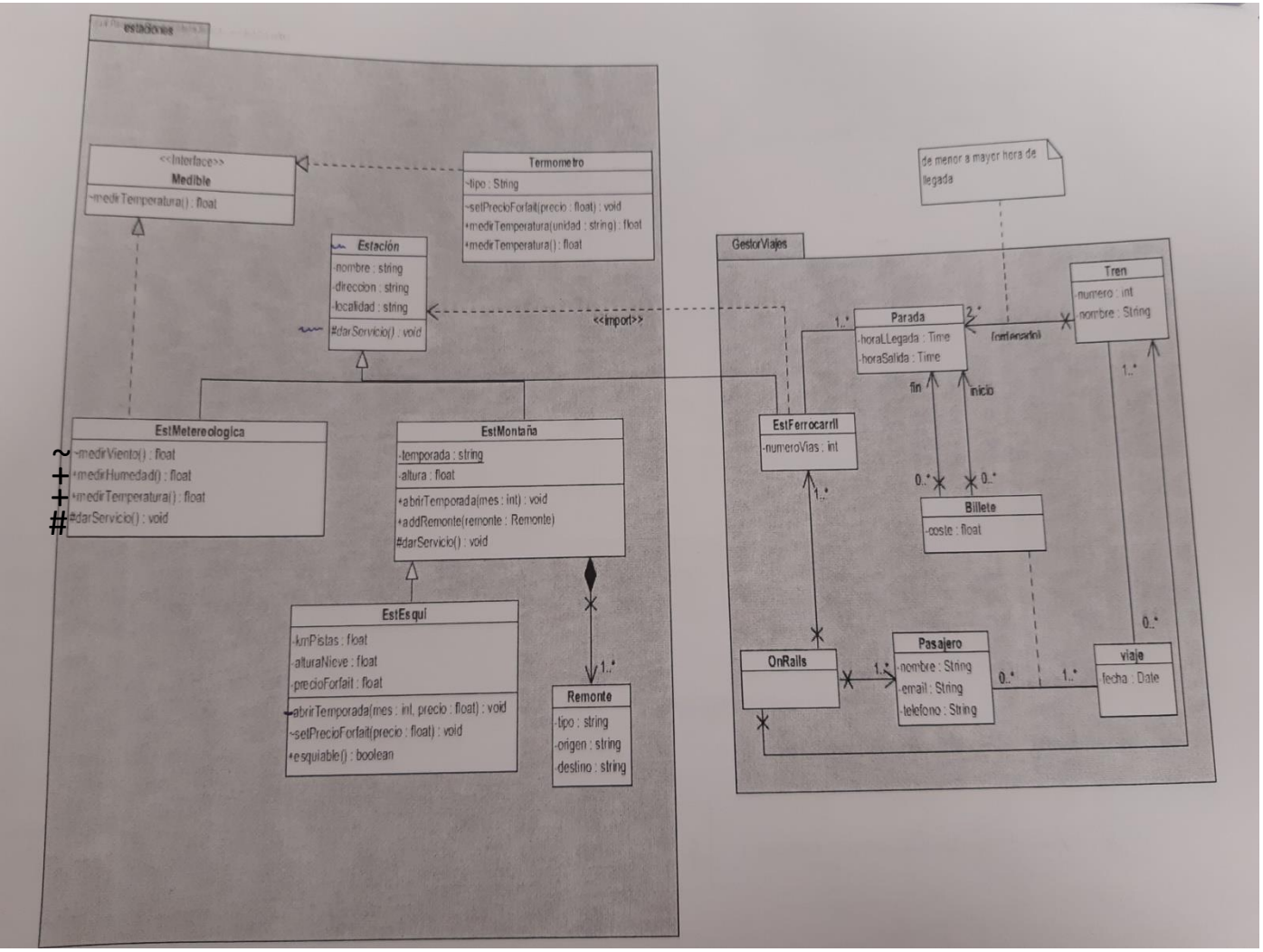
Ejercicio 10



Implementa en Ruby la clase Estación:

```
module Estaciones

class Estacion

   def initialize (m, d, l)
     @nombre = n
     @direccion = d
     @localidad = l
   end

   protected
   def darSerivicio
   end

   private_class_method :new
end

end
```

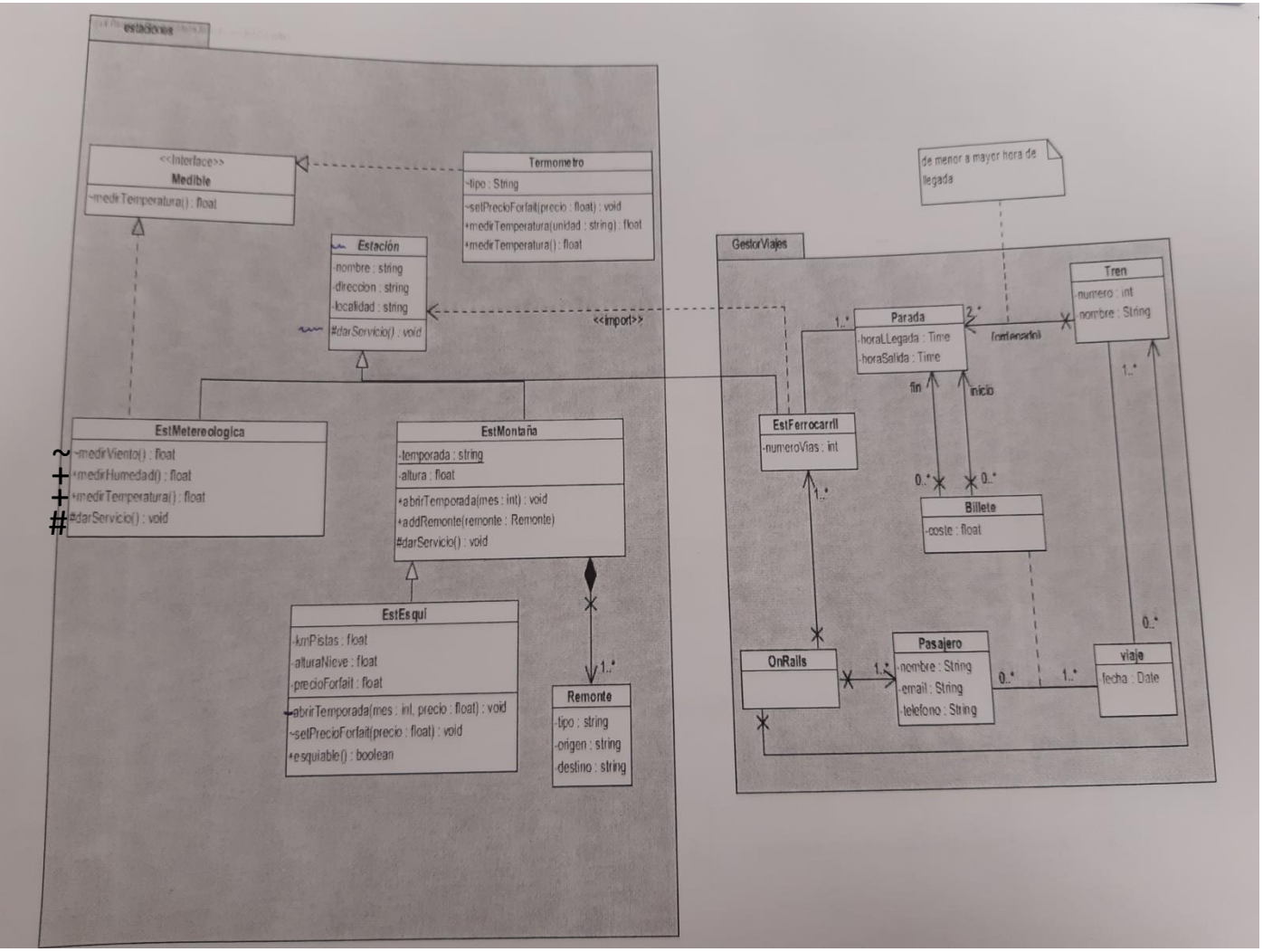
Ejercicio 11



Indica los errores de este código:

```
Medible v1;
Estacion v2;
EstMontaña v3;

----
v1 = new EstMontaña();

v1= new EstMetereologica();

v2= v1;

v1.medirViento();

v3 = new EstEsqui();

v3.darServicio();

List<Estacion> estaciones =
new ArrayList();

estaciones.add(v1);

estaciones.add(v3);

((EstMetereologica)
estaciones.get(1)).medirVie
nto();
```

Ejercicio 11



Indica los errores de este código:

```
Medible v1;
Estacion v2;
EstMontaña v3;

----
~~v1 = new EstMontaña();~~

v1= new EstMetereologica();

v2= (EstMetereologica) v1;

((EstMetereologica)
v1).medirViento();

v3 = new EstEsqui();

v3.darServicio();

List<Estacion> estaciones =
new ArrayList();

estaciones.add(v1);

estaciones.add(v3);

~~((EstMetereologica)
estaciones.get(1)).medirVie
nto();~~ //Fallo en ejecución
```