ETSIIT
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Ingeniería Informática + ADE

Universidad de Granada (UGR)

Autor: Ismael Sallami Moreno

Asignatura: Relación de Problemas Tema 3 : Herencia (PDOO)

Índice

1 Ejercicio 1

1.1. Enunciado

Dada la siguiente clase en Java:

```java
public class Vertebrado
{
    public ArrayList<String> partesDelAbdomen();
    public void desplazarse();
    public String comunicarse(Vertebrado vertebrado);
    protected Vertebrado obtenerCopia();
}
```

Indica con una cruz en la casilla correspondiente según si la declaración de los siguientes métodos de la clase `Mamifero`, subclase de `Vertebrado`, sobrecargan (*overloading*) o redefinen (*overriding*) a los de la clase `Vertebrado`.

1.2. Solución

Método	Sobrecarga	Redefinición
public ArrayList<String>partesDelAbdomen()		X
public void desplazarse(Modo m)	X	
public String comunicarse(Vertebrado vertebrado)		X
public String comunicarse(Mamifero mamifero)	X	
public Vertebrado obtenerCopia()		X
protected Mamifero obtenerCopia()		X

Nota:Al cambiar únicamente la visibilidad o el tipo hace que sea una redefición (Últimos dos casos)

2 Ejercicio 2

2.1. Enunciado

A partir de las siguientes clases:

```java
public class Persona {
    protected String nombre;
    public Persona(String nom) { this.setNombre(nom); }
    protected String getNombre() { return this.nombre; }
    public String hablar() { return "bla bla bla"; }
}
```

```java
public class Estudiante extends Persona {
    public String carrera;
    public int curso;
    public Estudiante(String nom, String carr, int cur) {
        super(nom);
```

```
        carrera = carr;
        curso = cur;
    }
    public void estudiar() { System.out.println("Estudiando"); }
}
```

Implementa la clase `EstudianteInformatica` que hereda de `Estudiante`. Debe tener:

- Una nueva variable de instancia que es una colección de `String` con los dispositivos que utiliza (por ejemplo, `PC`, `tablet`, `smartphone`).
- Métodos para consultar esa variable.
- Una redefinición del método `estudiar()` para que además de estudiar como los otros alumnos, diga que estudia con el último dispositivo de su colección.
- Un constructor para inicializar la clase.

Implementa este problema en Java y Ruby:

2.2. Solución

```
    public class EstudianteInformatica extends Estudiante {
        private ArrayList<String> dispositivos;
        public EstudianteInformatica(string nombre, string curso){
            super(nombre, "Informatica", curso);
            dispositivos = new ArrayList<String>();
        }

        @Override
        public void Estudiar(){
            System.out.println("Informatica con " + dispositivos.get(
                dispositivos.size() - 1));
        }
    } // Fin de la clase EstudianteInformatica
```

Listing 1: Clase Estudiante de Informática

```
    def EstudianteInformatica < Estudiante
        def initialize (nombre, curso)
            super(nombre, "Informatica", curso)
            @dispositivos = [] # @dispositivos = Array.new
        end

        def Estudiar
            super.Estudiar
            puts "Estudiando Informatica con #{@dispositivos[-1]}"
        end
    end
```

Listing 2: Clase Estudiante de Informática

3 Ejercicio 3

3.1. Enunciado

Nota: En las diapositivas de la Relación hay un error, donde pone 3, no es el ejercicio 3 es la parte 2 del ejercicio 2, es decir, otra clase, el ejercicio 3 es el marcado con el número 4.

Dada la siguiente clase abstracta en Java:

```java
abstract class Transporte {
    protected String marca;

    protected String getMarca() {
        return this.marca;
    }

    protected void setMarca(String marc) {
        this.marca = marc;
    }

    public abstract String hacerRuta(String origen, String destino);
}
```

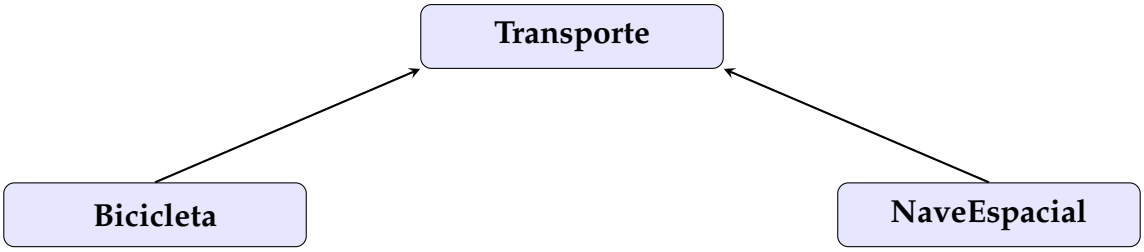
Crea dos nuevas clases, `Bicicleta` y `NaveEspacial`, que hereden de `Transporte` e implementen el método `hacerRuta` de forma diferente:

3.2. Solución

```java
    public class Bicicleta extends Transporte {
        @Override
        public String hacerRuta(String origen, String destino) {
            return "Pedaleando desde " + origen + " hasta " + destino;
        }
    }


    public class NaveEspacial extends Transporte {
        @Override
        public String hacerRuta(String origen, String destino) {
            return "Volando desde " + origen + " hasta " + destino + " por
                el espacio";
        }
    }
```

Listing 3: Clase Bicicleta y Clase Nave Espacial

Diagrama UML



4 Ejercicio 4

4.1. Enunciado

Hacer el ejercicio anterior, pero en Ruby.

4.2. Solución

```ruby
    def Bicicleta < Transporte
        def hacerRuta(origen, destino)
            "Pedaleando desde #{origen} hasta #{destino}"
        end
    end

    def NaveEspacial < Transporte
        def hacerRuta(origen, destino)
            "Volando desde #{origen} hasta #{destino} por el espacio"
        end
    end
```

Listing 4: Clase Bicicleta y Clase Nave Espacial

5 Ejercicio 5

5.1. Enunciado

Implementa en Java una clase paramétrica a partir de la cual se puedan definir clases de grupos de personas con un líder, como por ejemplo grupos de música con un cantante, o empresas con un jefe. Implementa también alguna de esas clases a partir de la clase paramétrica definida.

5.2. Solución

A continuación, se implementa una clase paramétrica en Java para definir grupos de personas con un líder. Posteriormente, se muestra un ejemplo de uso creando un grupo de música con un cantante.

```java
// Clase paramétrica genérica
public class Grupo<T> {
    private T lider;
    private List<T> miembros;

    public Grupo(T lider) {
        this.lider = lider;
        this.miembros = new ArrayList<>();
    }

    public T getLider() {
        return lider;
    }

    public void setLider(T lider) {
        this.lider = lider;
    }

    public void agregarMiembro(T miembro) {
        this.miembros.add(miembro);
    }

    public List<T> getMiembros() {
        return miembros;
    }

    @Override
    public String toString() {
        return "Líder: " + lider + ", Miembros: " + miembros;
    }
}

// Ejemplo de uso con un grupo de música
public class Musico {
    private String nombre;

    public Musico(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public String toString() {
        return nombre;
    }
}

public class Main {
    public static void main(String[] args) {
        Musico cantante = new Musico("Freddie Mercury");
        Grupo<Musico> banda = new Grupo<>(cantante);

        banda.agregarMiembro(new Musico("Brian May"));
        banda.agregarMiembro(new Musico("Roger Taylor"));
        banda.agregarMiembro(new Musico("John Deacon"));

```

```
        System.out.println(banda);
    }
}
```

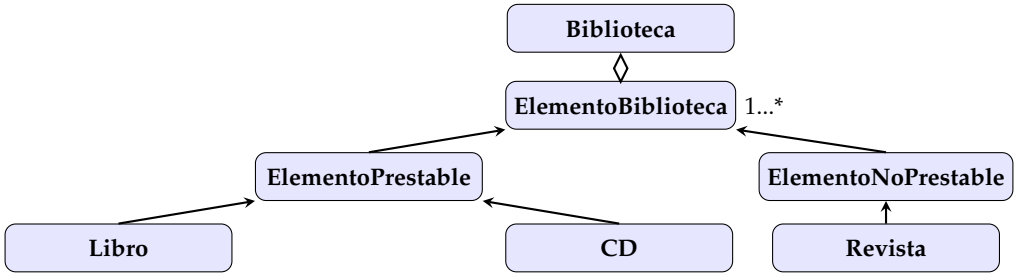
Si queremos implementar esta misma solución en Ruby, sería como sigue:

```ruby
# Clase paramétrica genérica
class Grupo
  attr_accessor :lider, :miembros

  def initialize(lider)
    @lider = lider
    @miembros = []
  end

  def agregar_miembro(miembro)
    @miembros << miembro
    # Otras formas de añadir un miembro en Ruby
    # @miembros.push(miembro)
    # @miembros += [miembro]
    # @miembros.concat([miembro])
  end

  def to_s
    "Líder: #{@lider}, Miembros: #{@miembros.join(', ')}"
  end
end

# Ejemplo de uso con un grupo de música
class Musico
  attr_accessor :nombre

  def initialize(nombre)
    @nombre = nombre
  end

  def to_s
    @nombre
  end
end

# Crear grupo de música
cantante = Musico.new("Freddie Mercury")
banda = Grupo.new(cantante)

banda.agregar_miembro(Musico.new("Brian May"))
banda.agregar_miembro(Musico.new("Roger Taylor"))
banda.agregar_miembro(Musico.new("John Deacon"))

puts banda
```

6 Ejercicio 6

6.1. Enunciado

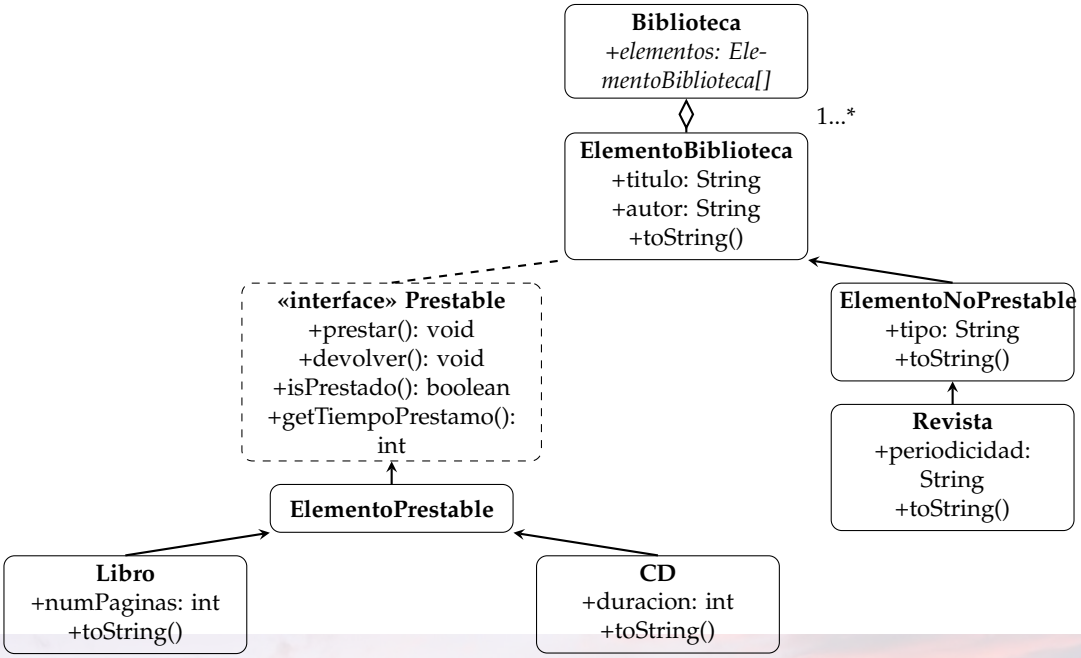
Partiendo del siguiente diagrama de clases, modifícalo añadiendo una interfaz Prestable implementada por la clase ElementoPrestable y sus subclases. Añade a este diagrama de clases los atributos y operaciones de las distintas clases y de la interfaz Prestable.



6.2. Solución

Detalles:

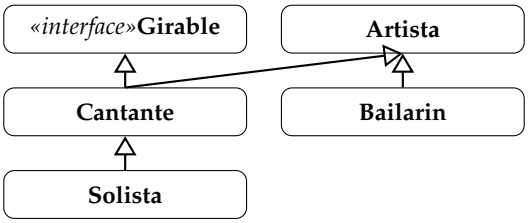
- Elemento Prestable hereda los métodos de la interfaz Prestable, por eso no se ponen en el diagrama.
- En el diagrama se sobreentiende el tipo de los toString que son de tipo String.



7 Ejercicio 7

7.1. Enunciado

Teniendo el cuenta el siguiente diagrama de clases, señala en la segunda columna de la tabla aquellas líneas de código que contienen un error de compilación por incompatibilidad de tipos (ECIT) y escribe en la tercera columna la corrección necesaria para evitarlos.



Código	ECIT	Corrección
Girable arti=new Artista();		
Cantante cant1=new Cantante();		
Cantante sol=new Solista();		
Solista cant2=new Cantante();		
Bailarin bail= new Artista();		

Figura 1: Tabla de código y corrección

7.2. Solución

Código	ECIT	Corrección
Girable arti=new Artista();	X	Girable arti = new Cantante()
Cantante cant1=new Cantante();	✓	
Cantante sol=new Solista();	✓	
Solista cant2=new Cantante();	X	Cantante cant2=new Cantante();
Bailarin bail= new Artista();	X	Bailarin bail= new Bailarin();

Figura 2: Tabla de código y corrección

7.2.1. Análisis línea por línea

- `Girable arti = new Artista();`
  - **Problema:** La clase `Artista` no implementa la interfaz `Girable`. Esto genera un error de compilación por incompatibilidad de tipos.
  - **Corrección:** Usar una clase que implemente la interfaz `Girable`, como `Cantante`.
  - **Resultado corregido:** `Girable arti = new Cantante();`
- `Cantante cant1 = new Cantante();`

- **Problema:** Ninguno. La asignación es válida porque la referencia y la instancia son de la misma clase.
- **Corrección:** No es necesario hacer ningún cambio.
- **Resultado corregido:** (Sin cambios)

- `Cantante sol = new Solista();`
  - **Problema:** Ninguno. `Solista` es una subclase de `Cantante`, por lo que esta asignación es válida.
  - **Corrección:** No es necesario hacer ningún cambio.
  - **Resultado corregido:** (Sin cambios)
- `Solista cant2 = new Cantante();`
  - **Problema:** Incompatibilidad de tipos. `Cantante` es la clase base de `Solista`, y no se puede asignar una instancia de una clase base a una referencia de una subclase.
  - **Corrección:** Cambiar el tipo de referencia a `Cantante`.
  - **Resultado corregido:** `Cantante cant2 = new Cantante();`
- `Bailarin bail = new Artista();`
  - **Problema:** `Artista` es una clase base abstracta y no se puede asignar directamente a una referencia de su subclase `Bailarin`.
  - **Corrección:** Crear una instancia de `Bailarin` en lugar de `Artista`.
  - **Resultado corregido:** `Bailarin bail = new Bailarin();`

8 Ejercicio 8

8.1. Enunciado

Teniendo en cuenta:

- El diagrama de clases del ejercicio anterior
- Que se han resuelto todos los errores del ejercicio
- Que en Java la primera posición de los contenedores es la 0
- Que todos los artistas actúan, pero sólo los cantantes cantan y sólo los solistas cantan solos

Marca las líneas donde se produce un error, indicando en la segunda columna si es de compilación (C) o de ejecución (E) y en la tercera el código correcto.

Código	Error (C/E)	Corrección
List<Artista>lista = new ArrayList();		
lista.add(arti);		
lista.add(cant1);		
lista.add(sol);		
lista.add(cant2);		
lista.add(bail);		
lista.get(1).canta();		
lista.get(0).actua();		
(Solista) lista.get(3).cantaSolo();		

Figura 3: Tabla de código y corrección

8.2. Solución

8.2.1. Tabla de diferencias

Criterio	Error de Compilación	Error de Ejecución
Cuándo ocurre	Durante la compilación.	Durante la ejecución del programa.
Tipo de problema	Problemas de sintaxis, tipos o reglas del lenguaje.	Problemas lógicos o de recursos.
Detección	Detectado por el compilador antes de ejecutar.	Ocurre mientras el programa está en ejecución.
Ejemplo común	Declaración de tipos incompatibles.	Dividir por cero o acceder a un índice inválido.

Cuadro 1: Principales diferencias entre errores de compilación y errores de ejecución

8.2.2. Solucion

Código	Error (C/E)	Corrección
List<Artista>lista = new ArrayList();	C	List<Artista>lista = new ArrayList<Artista>();
lista.add(arti);	No	Debido a que es un cantante y es una subclase de artista
lista.add(cant1);	No	Debido a que cantante esta por debajo en el diagrama de clases
lista.add(sol);	No	Debido a que es una subclase de Cantante y esta también lo es de Artista
lista.add(cant2);	No	”
lista.add(bail);	No	”
lista.get(1).canta();	No	Es un cantante
lista.get(0).actua();	E	Es un cantante
(Solista) lista.get(3).cantaSolo();	E	en la posición 3 es un cantante

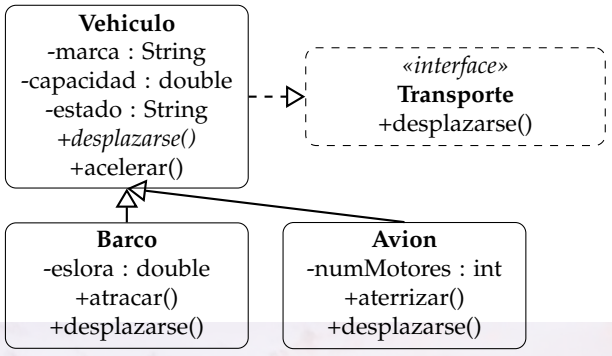
Figura 4: Tabla de código y corrección

9 Ejercicio 9

9.1. Enunciado

Dado el siguiente diagrama de UML, impleméntalo en Java. Presta atención a que el nombre de la clase Vehículo y su método desplazarse() aparecen en cursiva.

Nota: cuando en una clase no figura el tipo, es porque se trata de tipo void.



9.2. Solución

```
1
2  class Vehículo implements Transporte {
3      private String marca;
4      private double capacidad;
5      private String estado;
6
7      public void desplazarse() {
8          System.out.println("Desplazándose");
9      }
10
11     public void acelerar() {
12         System.out.println("Acelerando");
13     }
14 }
```

Listing 5: Clase Vehiculo

```
1
2  class Barco extends Vehículo {
3      private double eslora;
4
5      public void atracar() {
6          System.out.println("Atracando");
7      }
8
9      @Override
10     public void desplazarse() {
11         System.out.println("Desplazándose por el agua");
12     }
13 }
```

Listing 6: Clase Barco

```
1
2  class Avion extends Vehículo {
3      private int numMotores;
4
5      public void aterrizar() {
6          System.out.println("Aterrizando");
7      }
8
9      @Override
10     public void desplazarse() {
11         System.out.println("Desplazándose por el aire");
12     }
13 }
```

Listing 7: Clase Avion

```
1
2  interface Transporte {
3      void desplazarse();
4  }
```

Listing 8: Interfaz Transporte

10 Ejercicio 10

Teniendo presente el diagrama anterior, indica cuáles de los siguientes trozos de código son correctos y cuáles darían error de compilación o de ejecución. En caso de que den error, indica por qué y cómo lo solucionarías.

Código	**Error si procede**	**Corrección**
Transporte x = new Barco(); x.atracar();	Error de compilación, ya que el tipo de la variable x es Transporte y no tiene el método atracar.	Transporte x = new Barco(); ((Barco) x).atracar();
Avion av = new Avion(); av.acelerar();	✓.	
Vehiculo v = new Vehiculo(); v.desplazarse();	✓.	
Vehiculo v2 = new Vehiculo(); v2.acelerar();	✓.	
Transporte t = new Avion(): Barco b= new Barco(); t = b;	✓.	
Avion a = new Avion(); String est = a.estado;	Error de compilación, ya que el atributo estado es privado en la clase Vehiculo.	Implementar un consultor para que devuelva el valor estado.
Vehiculo v3 = new Barco(); ((Barco) v3).atracar();	✓.	
List<Transporte>listaTransportes = newArrayList(); listaTransportes.add(new Barco()); listaTransportes.add(new Avion()); listaTransportes.add(new Barco()); for(Transporte tr: listaTransportes) tr.acelerar(); tr.atracar();	Error de compilación, ya que el método atracar() no está definido en la interfaz Transporte.	No usar esos métodos.
List<Transporte>otraLista = new ArrayList(); otraLista.add(new Barco()); otraLista.add(new Avion()); otraLista.add(new Barco()); for(Transporte tr: otraLista) ((Barco) tr).acelerar(); ((Barco) tr).atracar();	Error de ejecución, ya que no se puede hacer un cast de un objeto de tipo Avion a Barco.	No usar esos métodos.

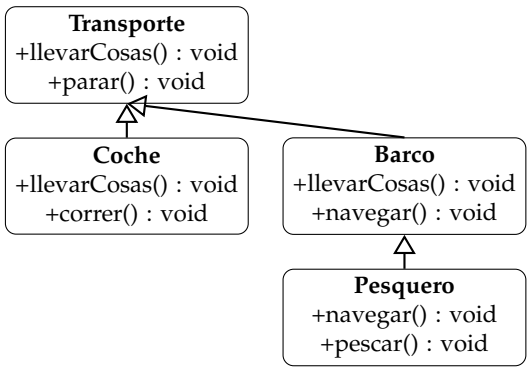
11 Ejercicio 11

11.1. Enunciado

A partir del siguiente diagrama de clases y del código proporcionado, detecta errores de compilación y de ejecución. Corrígelos para que funcione correctamente.

```
1   class PruebaLigadura {
2       public static void main ( String args[]){
3           Coche c1 = new Coche();
4           c1.llevarCosas();
5           c1.correr();
6           Barco b = new Barco();
7           b.llevarCosas();
8           b.navegar();
9           b.correr();
10          b=c1;
11          Pesquero p = new Pesquero();
12          p.navegar();
13          p.pescar();
14          p.llevarCosas();
15          p=b;
16          b=p;
17          b.navegar();
18          b.pescar();
19          ArrayList v= new ArrayList();
20          v.add(c1);
21          v.add(p);
22          (v.get(1)).navegar();
23          (v.get(1)).llevarCosas();
24      }
25  }
```

Listing 9: Ejercicio del Enunciado



11.2. Solución

```
1  class PruebaLigadura {
2      public static void main ( String args[]){
3          Coche c1 = new Coche();
4          c1.llevarCosas();
5          c1.correr();
```

```
        Barco b = new Barco();
        b.llevarCosas();
        b.navegar();
        b.correr(); // Error: Barco no tiene el método correr
        b=c1; // Error: No se puede asignar un Coche a un Barco
        Pesquero p = new Pesquero();
        p.navegar();
        p.pescar();
        p.llevarCosas();
        p=b; // Error: No se puede asignar un Barco a un Pesquero
        b=p;
        b.navegar();
        b.pescar(); // Error: Barco no tiene el método pescar
        ArrayList<Transporte> v= new ArrayList<>();
        v.add(c1);
        v.add(p);
        ((Barco) v.get(1)).navegar();
        v.get(1).llevarCosas();
    }
}

//-----------Implementación de las clases-----------------

class Transporte {
    public void llevarCosas() {
        System.out.println("Llevando cosas");
    }
    public void parar() {
        System.out.println("Parando");
    }
}

class Coche extends Transporte {
    public void correr() {
        System.out.println("Corriendo");
    }
}

class Barco extends Transporte {
    public void navegar() {
        System.out.println("Navegando");
    }
}

class Pesquero extends Barco {
    public void pescar() {
        System.out.println("Pescando");
    }
}
```

Listing 10: Ejercicio del Enunciado

12 Ejercicio 12

12.1. Enunciado

A partir de los siguientes diagramas de clases, resuelve las cuestiones que se plantea (siempre que sean de codificación hazlo en Java y en Ruby):

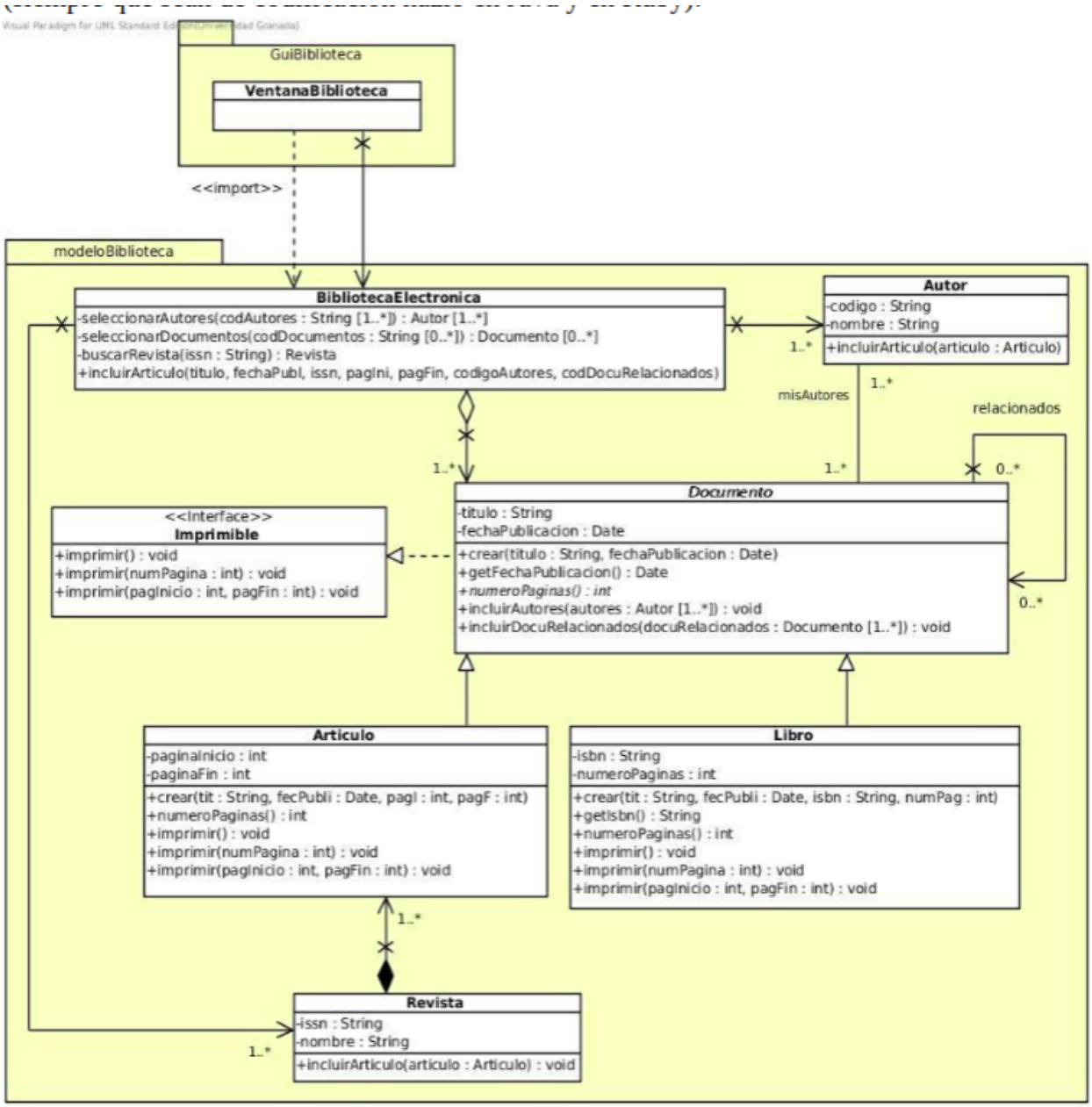


Figura 5: Diagrama de clases

- Define la clase Documento (cabecera, atributos y cabeceras de los métodos).
- Define la clase Articulo (cabecera, atributos y cabeceras de los métodos).

- Implementa el constructor que se indica en la clase Artículo.
- Define la interfaz Imprimible.
- Indica los atributos que definen el estado de la clase BibliotecaElectronica.
- La clase Documento figura en cursiva, lo que indica que es abstracta. Indica los dos motivos por los que lo es.
- En este modelo, ¿puede haber artículos que no estén ligados a una revista?
- Indica sobre las relaciones del diagrama de clases con cuál de los siguientes tipos se corresponden: asociación (AS), agregación (AG), composición (C), realización (R), herencia (H), dependencia (D).
- Escribe el contenido del fichero VentanaBiblioteca completo (pero sin añadir nada que no aparezca en el diagrama).
- En la clase Articulo, indica cuáles de sus métodos están sobrecargados o redefinidos. Justifica tu respuesta.
- Corrige el código:

```
1    Imprimible docu = new Documento("Intemperie", fecha); //
         suponiendo que fecha está inicializada
2    docu.imprimir();
```

- Corrige el código:

```
1    Documento docu = new Libro("ISBN10102030");
2    String codigo = docu.getIsbn();
```

- Rellena la siguiente tabla indicando el tipo estático y dinámico de la variable docu en las siguientes líneas de código:

Código	Tipo estático	Tipo dinámico
Documento docu = new Articulo(titulo, fecha, pagIni, pagFin);		
docu.imprimir(pagIni, pagFin);		
docu = new Libro(titulo, fecha, isbn, pags);		
docu.imprimir();		

- Indica si hay errores de compilación o ejecución en el código anterior (suponiendo que las variables titulo, fecha, pagIni, pagFin, isbn y pags han sido declaradas e inicializadas convenientemente con anterioridad). Justifica tu respuesta.
- Rellena la siguiente tabla marcando con una “X” la situación que corresponda a cada una de las líneas del siguiente bloque de código (suponiendo que las variables titulo, fecha, pagIni y pagFin han sido declaradas e inicializadas convenientemente con anterioridad).

Código	Ningún error	Sólo error de compilación	Sólo error de ejecución
Imprimible imp = new Articulo(titulo, fecha, pagIni, pagFin);			
Documento docu = imp;			
imp.numeroPaginas();			
((Libro) docu).getIsbn();			

12.2. Solución en Java

```
1   abstract class Documento implements Imprimible{
2       private String titulo;
3       private Date fechaPublicacion;
4       private Autor misAutores[];
5       private Documento relacionados[];
6
7
8       public Documento(String titulo, Date fechaPublicacion);
9
10      public Date getFechaPublicacion();
11
12      public abstract int numeroPaginas();
13
14      public void public incluirAutores(Autor[] autores);
15
16      public void incluirDocuRelacionados(Documento[] docuRelacionados);
17
18
19  }
```

Listing 11: lase Documento (cabecera, atributos y cabeceras de los métodos)

```
1   class Articulo extends Documento{
2       private int paginaInicio;
3       private int paginaFin;
4
5       public Articulo(String tit, Date fecPubli, int pagI, int pagF){
6           super(tit, fecPubli);
7           this.paginaInicio = pagI;
8           this.paginaFin = pagF;
9       }
10      public int numeroPaginas();
11      public void imprimir();
12      public void imprimir(int numeroPagina);
13      public void imprimir(int pagInicio, int pagFin);
14  }
```

Listing 12: clase Articulo (cabecera, atributos y cabeceras de los métodos), constructor de la clase Artículo

```java
interface Imprimible {
    void imprimir();
    void imprimir(int numeroPagina);
    void imprimir(int pagInicio, int pagFin);
}
```

Listing 13: Intefaz Imprimible

```java
class BibliotecaElectronica{
    private Autor autores[];
    private Documento documentos[];
    private Revista resvistas[];
}
```

Listing 14: Atributos que definen el estado de la clase BibliotecaElectronica

Motivos por los que la clase Documento es abstracta

1. Sirve como clase plantilla para las clases Libro y Revista que son clases hijas según el diagrama de clases.
2. Contiene al menos un método abstracto, que en este caso es el método `numeroPaginas()`.

¿Puede haber artículos que no estén ligados a una revista?

En este modelo si, lo que no puede haber son revistas que no estén ligadas a un Artículo debido al diagrama de clases y la lógica del rombo de la flecha de composición de la clase Artículo (parte de esta).

Pintar en el diagrama de clases las uniones de: asociación (AS), agregación (AG), composición (C), realización (R), herencia (H), dependencia (D)

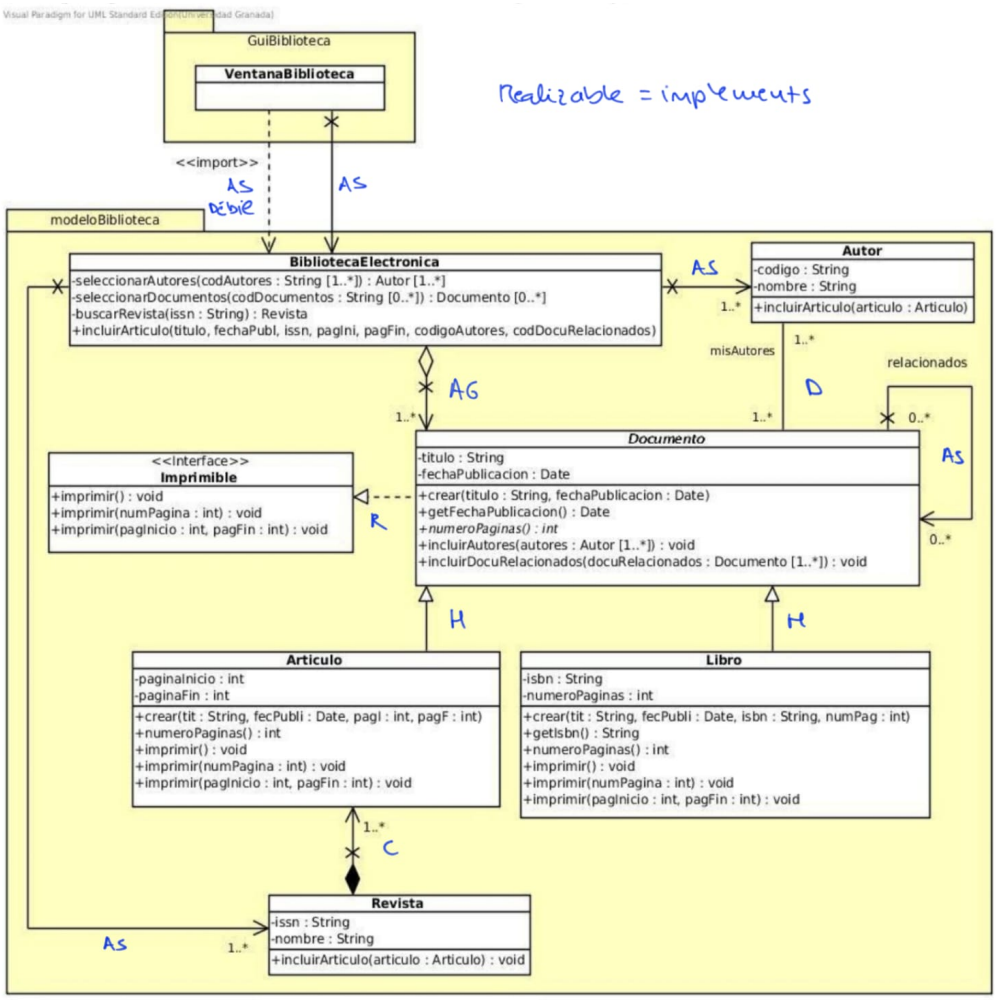


Figura 6: Diagrama de clases

Contenido del fichero VentanaBiblioteca completo (pero sin añadir nada que no aparezca en el diagrama)

```
1    class VentanaBiblioteca{
2        private BibliotecaElectronica biblioteca;
3
4        public VentanaBiblioteca(BibliotecaElectronica biblioteca){
5            this.biblioteca = biblioteca;
6        }
7    }
```

Listing 15: VentanaBiblioteca, el constructor se puede obviar, ya que usaríamos el por defecto

En la clase Articulo, indica cuáles de sus métodos están sobrecargados o redefinidos. Justifica tu respuesta.

- El método `imprimir()` está redefinido porque aparece en la clase `Documento`, el cual lo implementa de la interfaz `Imprimible`.
- El método `imprimir(int numeroPagina)` está redefinido porque aparece en la clase `Documento`, el cual lo implementa de la interfaz `Imprimible`.
- El método `imprimir(int pagInicio, int pagFin)` está redefinido porque aparece en la clase `Documento`, el cual lo implementa de la interfaz `Imprimible`.
- El método `numeroPaginas()` está redefinido porque sobrescribe el método de la clase `Documento`. Aunque sea abstracto, al cambiar únicamente el tipo se considera una redefinición[1].

Corrige el código

```
1    Imprimible docu = new Documento("Intemperie", fecha); // suponiendo que
         fecha está inicializada
2    docu.imprimir();
```

Listing 16: Código a corregir

Corrección:

```
1    Documento docu = new Articulo("Intemperie", fecha, 1, 10);
2    docu.imprimir();
```

Listing 17: Corrección del código

Corrige el código:

```
1    Documento docu = new Libro("ISBN10102030");
2    String codigo = docu.getIsbn();
```

Listing 18: Código a corregir

Corrección:

```
1    Libro docu = new Libro("ISBN10102030");
2    String codigo = docu.getIsbn();
```

Listing 19: Corrección del código

[1] Mirar ejercicio 1.

Rellena la siguiente tabla indicando el tipo estático y dinámico de la variable docu en las siguientes líneas de código:

Código	Tipo estático	Tipo dinámico
Documento docu = new Articulo(titulo, fecha, pagIni, pagFin);	Documento	Articulo
docu.imprimir(pagIni, pagFin);	Documento	Articulo
docu = new Libro(titulo, fecha, isbn, pags);	Documento	Libro
docu.imprimir();	Documento	Libro

Indica si hay errores de compilación o ejecución en el código anterior (suponiendo que las variables titulo, fecha, pagIni, pagFin, isbn y pags han sido declaradas e inicializadas convenientemente con anterioridad). Justifica tu respuesta.

- En la primera línea de código no hay errores de compilación ni de ejecución.
- Si, debido a que no puedes usar imprimir de Documento, ya que aunque derive de la interfaz que lo tenga, no esta implementado en Documento. Vemos que esta en Artículo, pero para usarlo debemos de realizar un cast: `((Articulo)docu).imprimir();`
  Nota: en java solo se puedes usar las funciones de los tipos estáticos, por eso en este caso da error de compilación y lo tenemos que hacer de la manera indicada.
- En la tercera línea de código no hay errores de compilación ni de ejecución.
- Pasa lo mismo que en la 2º explicación de este.

Rellena la siguiente tabla marcando con una "X" la situación que corresponda a cada una de las líneas del siguiente bloque de código (suponiendo que las variables titulo, fecha, pagIni y pagFin han sido declaradas e inicializadas convenientemente con anterioridad).

Código	Ningún error	Sólo error de compilación	Sólo error de ejecución
Imprimible imp = new Articulo(titulo, fecha, pagIni, pagFin);	X		
Documento docu = imp;		X	
imp.numeroPaginas();		X	
((Libro) docu).getIsbn();			X

12.3. Solución en Ruby

```ruby
class Documento
    attr_reader :titulo, :fecha_publicacion

    def initialize(titulo, fecha_publicacion)
        @titulo = titulo
        @fecha_publicacion = fecha_publicacion
    end

    def numero_paginas
        raise NotImplementedError, 'Debe ser implementado en una
           subclase'
    end

    def incluir_autores(autores)
        # Implementación
    end

    def incluir_documentos_relacionados(documentos)
        # Implementación
    end
end
```

Listing 20: Clase Documento (cabecera, atributos y cabeceras de los métodos)

```ruby
class Articulo < Documento
    attr_reader :pagina_inicio, :pagina_fin

    def initialize(titulo, fecha_publicacion, pagina_inicio, pagina_fin
       )
        super(titulo, fecha_publicacion)
        @pagina_inicio = pagina_inicio
        @pagina_fin = pagina_fin
    end

    def numero_paginas
        @pagina_fin - @pagina_inicio + 1
    end

    def imprimir
        # Implementación
    end

    def imprimir_por_pagina(numero_pagina)
        # Implementación
    end

    def imprimir_por_rango(pagina_inicio, pagina_fin)
        # Implementación
    end
end
```

Listing 21: Clase Articulo (cabecera, atributos y cabeceras de los métodos), constructor de la clase Artículo

```ruby
module Imprimible
```

```ruby
        def imprimir
            # Implementación
        end

        def imprimir_por_pagina(numero_pagina)
            # Implementación
        end

        def imprimir_por_rango(pagina_inicio, pagina_fin)
            # Implementación
        end
    end
```

Listing 22: Interfaz Imprimible

```ruby
    class BibliotecaElectronica
        attr_accessor :autores, :documentos, :revistas
    end
```

Listing 23: Atributos que definen el estado de la clase BibliotecaElectronica

13 Ejercicio 13

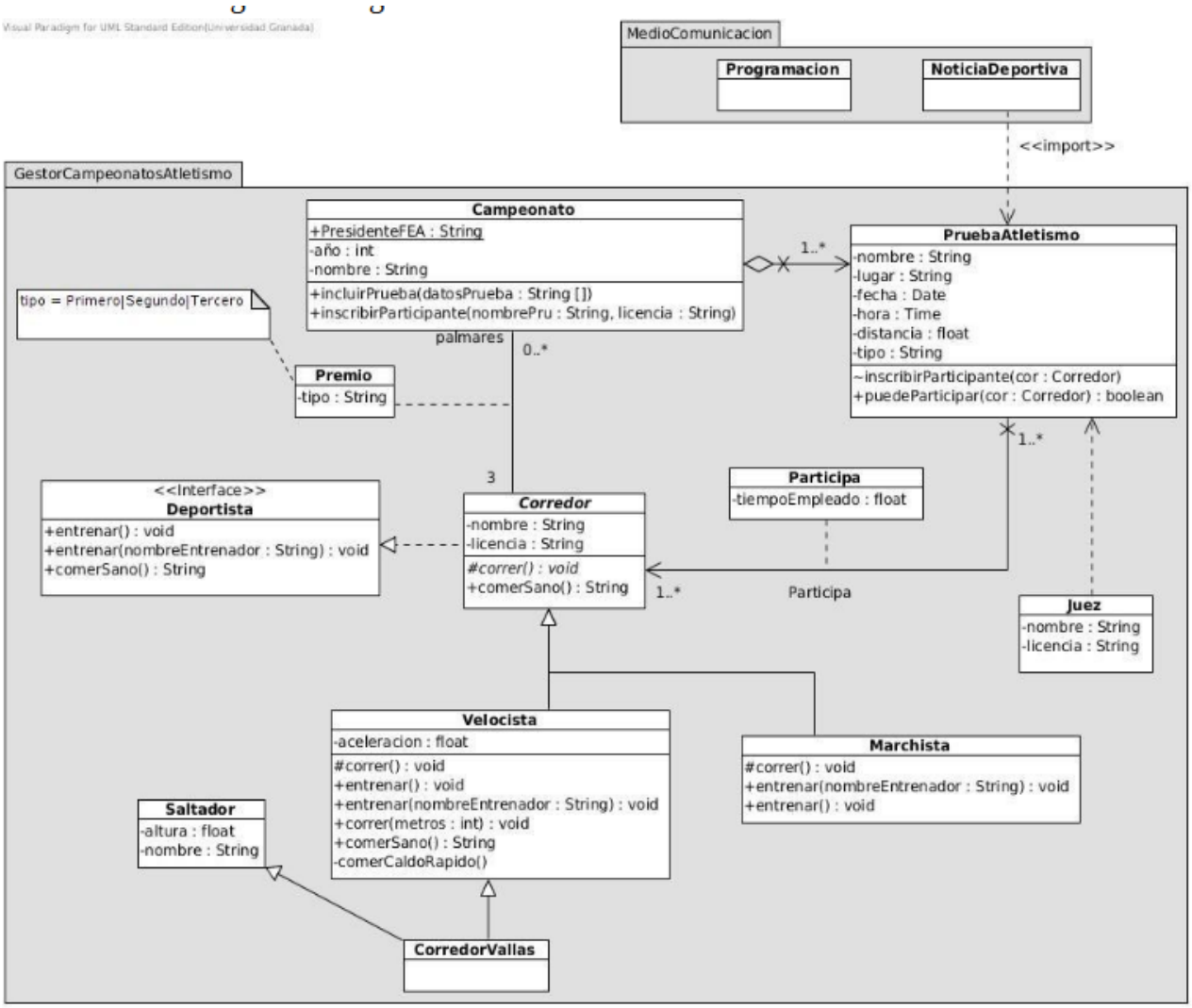


Figura 7: Diagrama de clases

13.1. Enunciado Y Solución

Cuestión	Respuesta
Desde la clase NoticiaDeportiva se puede acceder a todos los elementos públicos del paquete GestorCampeonatosAtletismo directamente	F
Un CorredorVallas es un Velocista y Saltador	V
El estado de un objeto de la clase Juez no viene determinado por el estado de un objeto de la clase PruebaAtletismo	V
Una PruebaAtletismo puede existir sin estar asociada a la clase Campeonato	V
Un Velocista es un Corredor	V
Un Deportista es un Corredor	F
La clase Corredor tiene 4 métodos, 3 abstractos y 1 concreto	F
La clase Marchista está mal representada en el diagrama, debe ser abstracta	F
La clase CorredorVallas presenta un conflicto de nombres	F
Todas las instancias de la clase Campeonato tienen una copia de la variable PresidenteFEA	V

Cuadro 2: Cuestiones Verdaderas o Falsas

Método	Abstracto
correr()	No, porque es una redefinición de Corredor
comerSano()	No, es una redefinición de la clase Corredor
entrenar()	No, porque es una redefinición de la interfaz Deportista
entrenar(nombreEntrenador:String)	No, porque es una redefinición de la interfaz Deportista

Cuadro 3: Métodos abstractos en Marchista

Método	Redefinido	Sobrecargado
correr()	X, porque cambia la cabecera	
comerSano()	X	
entrenar()	X, aunque sea de la interfaz	

Cuadro 4: Métodos redefinidos o sobrecargados en Velocista

	Variable	Tipo Estático	Tipo Dinámico
Deportista c = new Velocista();	c	Deportista	Velocista
Corredor d = new Marchista();	d	Corredor	Marchista
c = new Marchista();	c	Deportista	Marchista
d = c;	d	Corredor	Marchista

Cuadro 5: Tipo estático y dinámico de variables

Código	**Corrección del error en Compilación**	**Error en ejecución**
Corredor c = new Velocista();	Ninguno	Ninguno
Deportista d = new Marchista();	Ninguno	Ninguno
d.comerSano();	Ninguno	Ninguno
Marchista m = (Marchista) d;	Ninguno	Ninguno
d = c;	Ninguno	Ninguno
d.correr(150);	El método correr(int) no está definido en la interfaz Deportista	Ninguno
Velocista v = (Velocista) d;	Ninguno	Error en ejecución si d no es una instancia de Velocista
ArrayList<Corredor>corredores = new ArrayList<>();	Ninguno	Ninguno
corredores.add(c);	Ninguno	Ninguno
corredores.add(m);	Ninguno	Ninguno
corredores.get(0).correr(10);	El método correr(int) no está definido en Corredor	Ninguno
corredores.get(1).correr(10);	El método correr(int) no está definido en Corredor	Ninguno
c = new Corredor();	No se puede instanciar una clase abstracta	Ninguno

Cuadro 6: Corrección de errores de compilación y ejecución

Para una explicación más detallada pincha aquí.

```
public class Club<T> {
private Map<String, T> miembros; // key = número de licencia
private T lider;

// Constructor
public Club(T lider) {
    this.lider = lider;
    this.miembros = new HashMap<>();
}

// Consultor de un miembro del club a partir del numero de licencia
```

```
    public T getMiembro(String numeroLicencia) {
        return miembros.get(numeroLicencia);
    }

    // Incluir un nuevo miembro en el club
    public void incluirMiembro(String numeroLicencia, T miembro) {
        miembros.put(numeroLicencia, miembro);
    }

    // Cambiar el líder
    public void setLider(T lider) {
        this.lider = lider;
    }

    // Obtener el líder
    public T getLider() {
        return lider;
    }
}
```

Listing 24: Código en java

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

public class Main {
    public static void main(String[] args) {
        // Creamos una instancia de Velocista
        Velocista lider = new Velocista("Usain Bolt", "001", 9.58f);

        // Creamos un club de Velocistas, con Usain Bolt como líder inicial
        Club<Velocista> clubDeVelocistas = new Club<>(lider);

        // Creamos más miembros del club
        Velocista miembro1 = new Velocista("Carl Lewis", "002", 9.86f);
        Velocista miembro2 = new Velocista("Tyson Gay", "003", 9.69f);

        // Añadimos miembros al club
        clubDeVelocistas.incluirMiembro("002", miembro1);
        clubDeVelocistas.incluirMiembro("003", miembro2);

        // Consultamos los miembros
        System.out.println("Líder del club: " + clubDeVelocistas.getLider()
            .getNombre());
        System.out.println("Miembro con licencia 002: " + clubDeVelocistas.
            getMiembro("002").getNombre());
        System.out.println("Miembro con licencia 003: " + clubDeVelocistas.
            getMiembro("003").getNombre());

        // Cambiamos el líder del club
        clubDeVelocistas.setLider(miembro1);
        System.out.println("Nuevo líder del club: " + clubDeVelocistas.
            getLider().getNombre());
    }
}

```

```java
// Clase Velocista
class Velocista {
    private String nombre;
    private String licencia;
    private float mejorMarca;

    // Constructor
    public Velocista(String nombre, String licencia, float mejorMarca) {
        this.nombre = nombre;
        this.licencia = licencia;
        this.mejorMarca = mejorMarca;
    }

    // Getters
    public String getNombre() {
        return nombre;
    }

    public String getLicencia() {
        return licencia;
    }

    public float getMejorMarca() {
        return mejorMarca;
    }
}
```

Listing 25: Main

Listing 26: Salida del Main

```
Líder del club: Usain Bolt
Miembro con licencia 002: Carl Lewis
Miembro con licencia 003: Tyson Gay
Nuevo líder del club: Carl Lewis
```

14 Ejercicio 14

14.1. Enunciado

Dado el siguiente diagrama de clases:

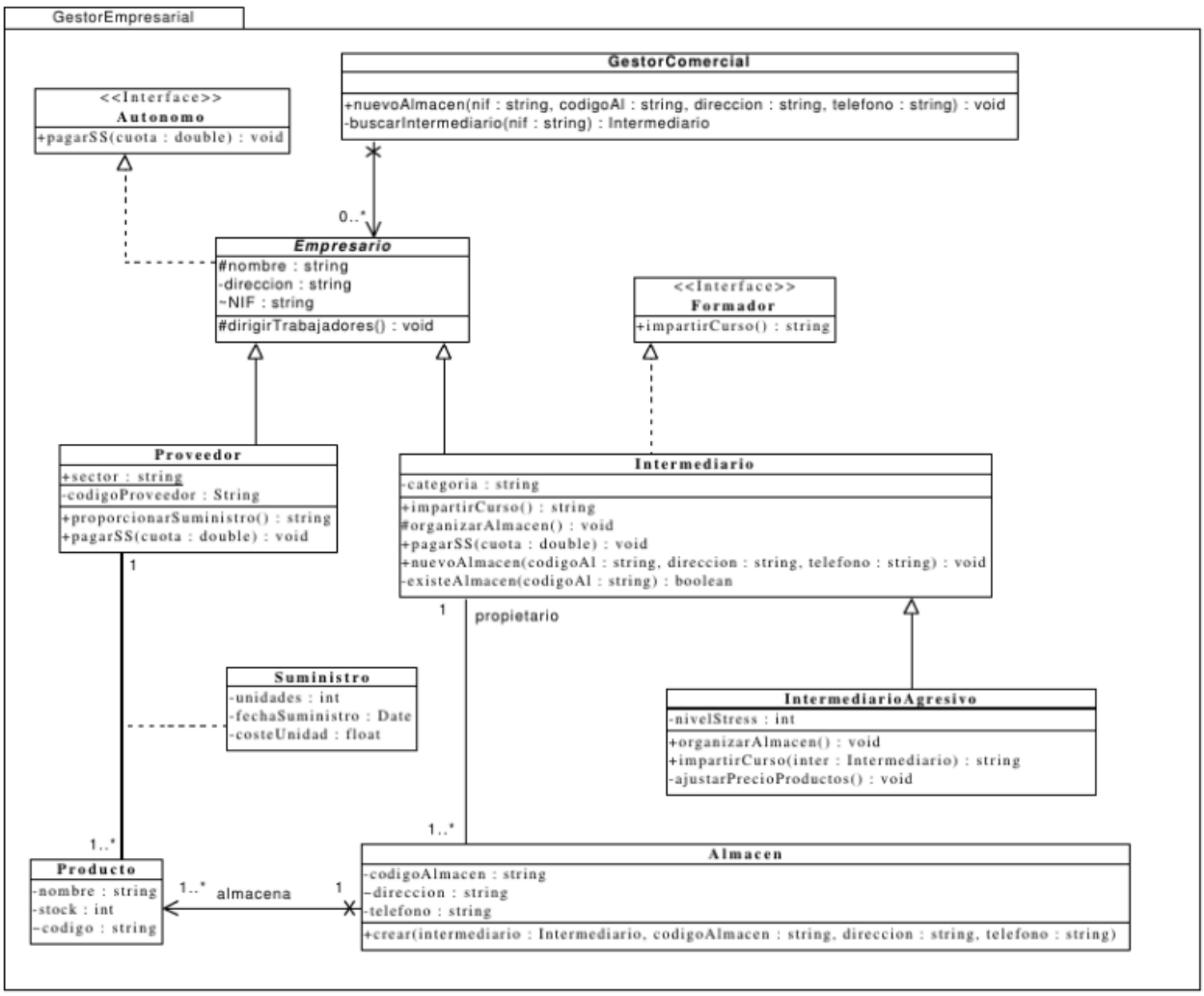


Figura 8: Diagrama de clases

Indica si las siguientes líneas de código Java producen error de compilación, de ejecución, ambos o ninguno. Supón que están en un `main` en una clase nueva dentro del mismo paquete. Si hay error explica cómo lo arreglarías y si no hay error, indícalo explícitamente. Considera para este ejercicio que todas las clases tienen un constructor válido que no recibe atributos.

Código	Error de compilación	Error de ejecución
Formador f = new Intermediario();	Ninguno	Ninguno
f.pagarSS(23.4);	Ninguno	Ninguno
Autonomo auto1 = f;	Sí	Ninguno
Autonomo auto2 = (Proveedor) f;	Sí	Ninguno
IntermediarioAgresivo emp1 = new IntermediarioAgresivo();	Ninguno	Ninguno
emp1.ajustarPrecioProductos();	Ninguno	Ninguno
Empresario emp2 = new Empresario();	Ninguno	Ninguno
ArrayList<Formador>formadores = new ArrayList();	Sí	Ninguno
formadores.add(f);	Ninguno	Ninguno
formadores.add(emp1);	Ninguno	Ninguno
formadores.get(1).impartirCurso();	Ninguno	Ninguno
formadores.get(1).impartirCurso(f);	Sí	Ninguno
((IntermediarioAgresivo)formadores.get(0)).impartirCurso(emp1);	Ninguno	Ninguno

14.2. Solución

Código	Error de compilación	Error de ejecución
Formador f = new Intermediario();	Ninguno	Ninguno
f.pagarSS(23.4);	Ninguno	Ninguno
Autonomo auto1 = f;	Sí	Ninguno
Autonomo auto2 = (Proveedor) f;	Sí	Ninguno
IntermediarioAgresivo emp1 = new IntermediarioAgresivo();	Ninguno	Ninguno
emp1.ajustarPrecioProductos();	Ninguno	Ninguno
Empresario emp2 = new Empresario();	Ninguno	Ninguno
ArrayList<Formador>formadores = new ArrayList();	Sí	Ninguno
formadores.add(f);	Ninguno	Ninguno
formadores.add(emp1);	Ninguno	Ninguno
formadores.get(1).impartirCurso();	Ninguno	Ninguno
formadores.get(1).impartirCurso(f);	Sí	Ninguno
((IntermediarioAgresivo)formadores.get(0)).impartirCurso(emp1);	Ninguno	Ninguno

Para una explicación más detallada pincha aquí.