Deportistas y conductores

```
interface Sancionable(){
  void sancionar(int tiempo);
}


class Deportista{
  protected String nombre;
  public Deportista(){
       System.out.println("Se crea el deportista");
  }
  public Deportista(String n){
       nombre = n;
       System.out.println("Se crea el deportista " + nombre);
  }
  public void retirar(){
       System.out.println("Se retira el deportista " + nombre);
  }
}

class Tenista extends Deportista implements Sancionable {
  public Tenista(String n){
    super(n);
    System.out.println("Se crea el tenista " + nombre);
  }
  public void jugar(String tipo){
    System.out.println("El tenista juega un " + tipo);
  }
  public void sancionar(int t){
    System.out.println("Tenista sancionado" + t+ " dias");
  }
  public void retirar(){
    System.out.println("Se retira el tenista " + nombre);
  }
}

class Golfista extends Deportista {
  public Golfista(String n){
    nombre = n;
    System.out.println("Se crea el golfista " + nombre);
  }
  public void jugar(int num){
    System.out.println("El golfista juega + num + " hoyos");
  }
}

class Conductor implements Sancionable{
      public void sancionar(int t){
       System.out.println("Conductor sancionado" + t + " meses");
      }
}
```

Aulas

La siguiente jerarquía de clases representan aulas y dos subtipos: aulas de proyección y aulas de informática.

```
public abstract class Aula {
	private double longitud;
	private double anchura;

	public Aula(double longitud, double anchura){
		this.longitud = longitud;
		this.anchura = anchura;
	}

	public abstract int capacidad();

	private double area(){
		return longitud*anchura;
	}

	@Override public String toString(){
		return  "Superficie: " + area() + " metros cuadrados\n " +
			"Capacidad: " + capacidad() + " alumnos.";
	}
}

public class AulaProyeccion extends Aula{
	private String marcaProy; //Marca del proyector

	public AulaProyeccion(){
		super(10,10);
		this.marcaProy = "EPSON";
	}

	public AulaProyeccion(double longitud, double anchura, String marca){
		super(longitud, anchura);
		this.marcaProy = marca;
	}

	@Override public int capacidad(){
		//Un alumno cada dos metros cuadreados
		return area()/2;
	}
}

public class AulaInformatica extends AulaProyeccion {
	private int numOrd; //nº de ordenadores

	public AulaInformatica (double longitud, double anchura, String marca, int numOrd){
		super(longitud, anchura, marca);
		this.numOrd = numOrd;
	}
	@Override public final int capacidad(){
		//Dos alumnos por cada ordenador
		return numOrd * 2;
	}

}
```

Ejercicio 1.- **(2 puntos) Basándote en los interfaces y clases de que hay en la página de título "*Deportistas y conductores*", indica qué líneas del siguiente programa son correctas (marcando la columna OK), tiene errores de compilación (EC) o errores de ejecución (EE). Si no tienen errores indica qué salida producen al ejecutarse el programa y si contienen errores, cuál es el motivo.**

Nombre: _____

	OK	EC	EE	Salida / Por qué hay error
`Sancionable s1 = new Tenista("Nadal");`				
`s1.sancionar(5);`				
`s1.jugar("Play offs");`				
`Sancionable s2 = new Golfista("Tiger");`				
`s2.sancionar(5);`				
`Deportista d1 = new Tenista("Federer");`				
`d1.sancionar(5);`				
`d1.retirar();`				
`d1 = new Conductor();`				
`d1 = new Golfista("Tiger");`				
`((Sancionable)d1).sancionar(5);`				
`((Golfista)d1).jugar(18);`				
`Conductor c1 = new Conductor();`				
`c1.sancionar(5);`				

Ejercicio 2.- (2 puntos) Responde a las siguientes preguntas referentes a la jerarquía Aulas **justificando brevemente la respuesta.**

a) Si en la clase ***AulaProyección*** eliminámos la línea que llama `super(longitud, anchura);` del constructor, ¿se produciría error de compilación?

b) Si en el constructor de ***AulaInformática*** eliminámos la línea que llama `super(longitud, anchura, marca);`, ¿se produciría error de compilación?

c) ***capacidad()*** es abstracto en *Aula*. ¿Qué ocurriría si eliminasemos el método ***capacidad( )*** de la clase ***AulaProyección*** ?

d) ¿Qué ocurriría si eliminasemos el método ***capacidad( )*** de la clase ***AulaInfomática***?

e) ¿Sería posible crear una subclase de AulaInformática en la que la capacidad fuese de tres alumnos por ordenador?

Ejercicio 3 (3.5 puntos): **Se quiere realizar la gestión que un Centro de Alto Rendimiento para deportistas hace de sus asociados. Existen cuatro tipos de deportistas:**

> **Nota: Se valorará la reutilización, en las clase hija, del código realizado en la clase padre, así como la correcta elección entre reescrituras totales y parciales, clases abstractas, …**

- **Deportistas externos**. Deportistas que acuden al centro únicamente a entrenar. y pagan por ello una mensualidad

- **Deportistas internos:** Son deportistas que, además de entrenar, viven en el centro de alto rendimiento. Éste les proporciona alojamiento y, según el régimen de pensión, también la comida. Un deportista interno puede estar en régimen de "SOLO_ALOJAMIENTO", "ALOJAMIENTO+DESAYUNO" o en régimen de "PENSION_COMPLETA".
Los deportistas internos pagan una cantidad que asciende a:
  - 300 € si están en régimen de solo alojamiento.
  - 500 € si están en régimen de alojamiento más desayuno.
  - 800 € si están en régimen de pensión completa.

- **Deportistas becados**: Se trata de **deportistas internos** que disfrutan de una beca. La beca consiste en que se les reduce en un porcentaje lo que tienen que pagar. Por ejemplo una beca del 20% implicaría que tienen que pagar un 20% menos que si no disfrutaran de beca. Cada deportista becado puede tener un porcrentaje de beca distinto.

- **Gimnastas**: Se trata de deportistas becados que practican el deporte "gimnasia deportiva". Como es difícil obtener ingresos por esta disciplina y requiere dedicación completa, los gimnastas gozan siempre de una beca del 100%.

Diseñar la clase **Deportista** y sus subclases **DeportistaExterno, DeportistaInterno, DeportistaBecado y Gimnasta y relacionarlas mediante herencia**

Los atributos serán los siguientes:

- Para todos los **deportistas**: el *dni* (String), el *nombre* del deportista (String), el deporte que practica (String)
- Los deportistas **externos,** además, la mensualidad (double) que paga.
- Para los **internos,** además, el *regimen* de pensión (String)
- Para los **becados**, además,el *porcentajeBeca* (double): Porcentaje de beca que tiene, es decir porcentaje de descuento que disfruta.

Se implementarán los siguientes métodos:

- El constructor de cada clase, de manera que todos los atributos queden convenientemente inicializados. En la clase **DeportistaInterno**, si el régimen de pensión no coincide con alguno de los que se han descrito anteriormente, se lanzará la excepción IllegalArgumentException

- toString() con los siguientes aspectos:

> **Nota: El tipo de pensión aparecerá como "Solo alojamiento", "Media" o "Completa".**

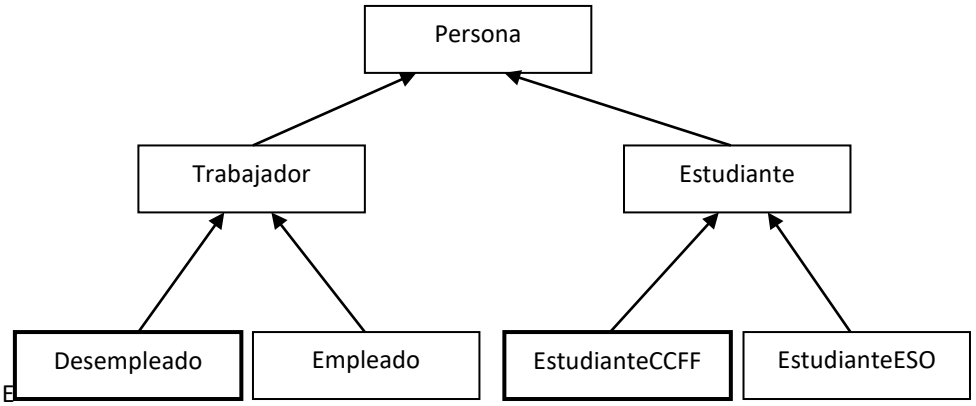
(En dep externos)	*(En los internos)*	*(En los becados)*
EXTERNO	INTERNO	BECADO
Nombre: Javi	Nombre: Miguel	INTERNO
DNI: 111111	DNI: 222222	Nombre: Luis
Deporte: futbol	Deporte: natación	DNI: 111111
	Pensión: Media	Deporte: baloncesto
		Pensión: Completa
		Beca: 20%

- double calcularImporteAPagar() de cada clase: devuelve el importe que ha de pagar el deportista cada mes.
- void mostrarRecibo(): para todos los deportistas se muestra el nombre del deportista y el importe que ha de pagar el en el recibo mensual**, excepto en el caso de los Gimnastas, para los que se mostrará el nombre y "CUOTA GRATUITA"**

  NOMBRE: Javi

  TOTAL A PAGAR: xxxxx €

Ejercicio 4: (1.5 puntos) En el proyecto se encuentra, ya creada, la siguiente jerarquía de clases: En ella, se aprecian distintas especializaciones de la clase Persona



Se está considerando que tanto a los Desempleado como a los EstudiantesCCFF se les puede dar formación por lo que se quiere añadir a estas dos clases el siguiente método

void formar (int horas)

Por ello, **debes añadir** el método formar a ambas clases. El método ***formar*** recibe el número de horas de formación y debe mostrar un mensaje del tipo:

- En la clase Desempleado: "Desempleado recibe formación de XXX horas ",
- En la clase EstudianteCCFF: "Estudiante de CCFF recibe formación de XXX horas"

En el método main de la clase DarFormación, que se encuentra en el proyecto, se ha definido un array que contiene objetos Desempleado y objetos EstudianteCCFF. Se pide realizar un bucle que recorra el array y "dé formación" de 100 horas a cada elemento del array, es decir que llame al método *formar* de cada elemento del array, **haciendo los cambios que consideres necesarios en la jerarquía.**

Ejercicio 5 (1 punto): En la clase Lista se ha implementado un método genérico que recorre un ArrayList y modifica todos sus elementos. Para ello, hace uso del interfaz Modificador, que contiene un único método, llamado modificar

Se pide, completar el método main de la clase Ejercicio5 para que, usando el método Lista.modificarElementosLista, convierta a mayúsculas todos los elementos de la lista.