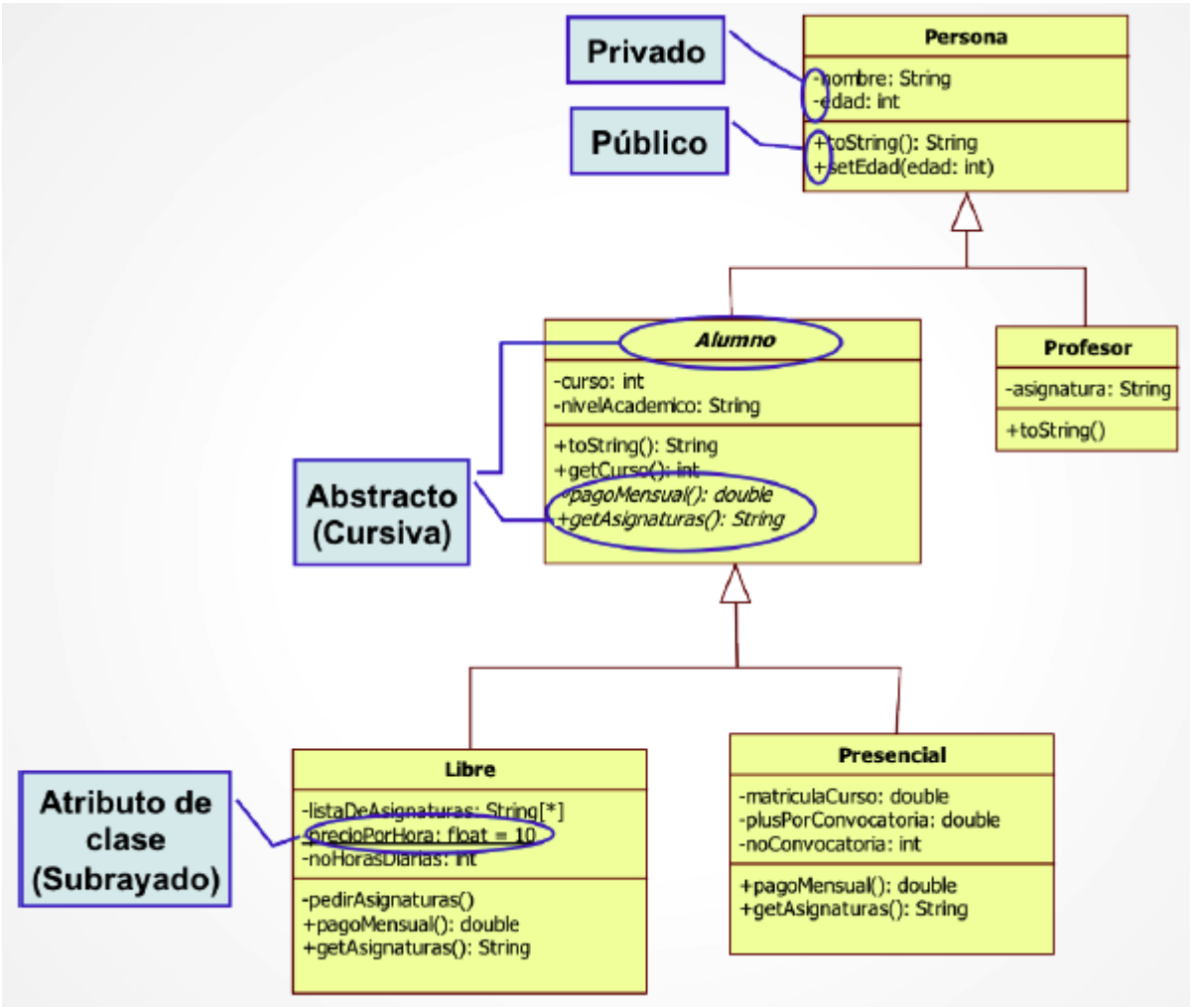
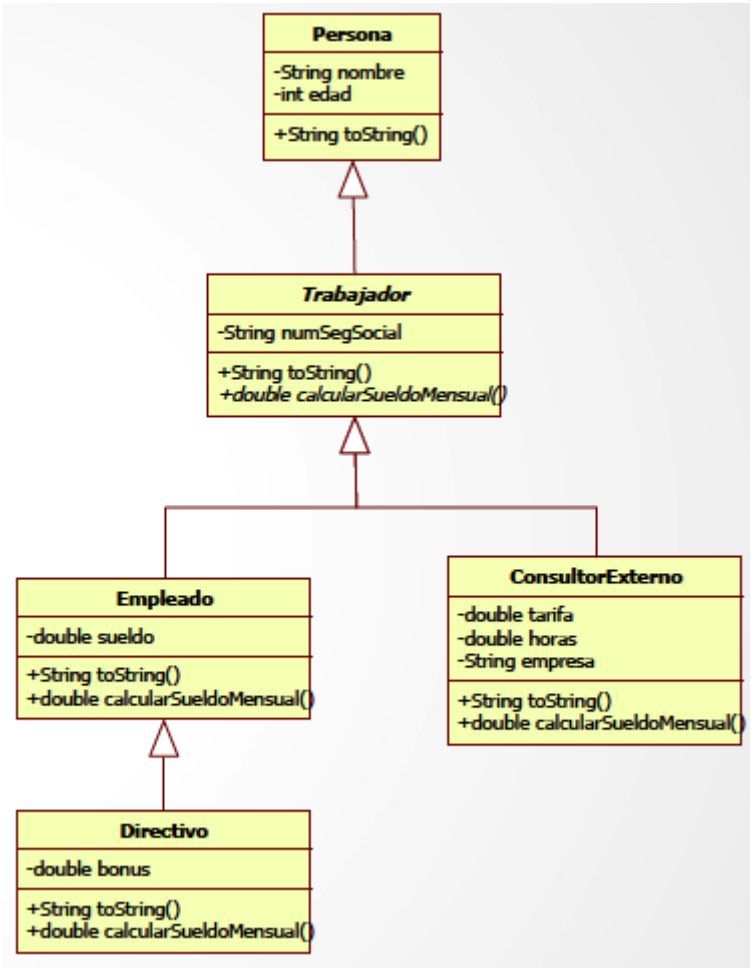
TEMA 5: HERENCIA ABSTRACTA

Ejercicio2:
Crear las clases Persona, Alumno, Profesor, Libre y Presencial con los atributos y métodos de la figura siguiente. La clase Alumno tiene dos métodos, pagoMensual y getAsignaturas que son comunes a todos los objetos de tipo Alumno y cuya implementación depende de si es de tipo Libre o Presencial. El alumno Libre paga mensualmente por las horas de clase diarias de las asignaturas de las que está matriculado, mientras que el alumno Presencial paga el precio de la matrícula más un precio fijo por convocatoria (plusConvocatoria) multiplicado por el número de la convocatoria, todo ello dividido en los 12 meses del año.



Ejercicio3:
En el ejercicio del tema anterior, añadir a la clase Trabajador un método abstracto double calcularSueldoMensual(), a la clase Empleado un método double calcularSueldoMensual() que devuelve el sueldo dividido entre 14, a la clase Directivo un método double calcularSueldoMensual() que devuelve el sueldo dividido entre 14 y le suma su bonus, a la clase ConsultorExterno un método double calcularSueldoMensual() que devuelve tarifa*horas.

Implementar, además, un programa principal (main) que haga lo siguiente:
→ Crear un vector de 4 trabajadores que contenga: 1 directivo, 2 empleados y 1 consultor externo.
→ Calcular el sueldo mensual total de las personas del vector y mostrarlo en pantalla.



TEMA 5: INTERFACES

Podría suceder que varias clases no relacionadas necesariamente compartan un mismo conjunto de operaciones. Para ello utilizamos las **interfaces**, ya que nos permiten especificar un conjunto de operaciones, y dependiendo de la clase, cada operación se realizará de una manera diferente. **Ejemplo**: las clases Parcela, Foto, Cuadro, EspejoCircular, … incluyen los métodos calcularArea, calcularPerimetro, etc. En este caso podríamos definir una **interfaz** que agrupe todos los métodos comunes que contenga tan solo las cabeceras de estos métodos y luego definir varias clases que implementen los métodos de la interfaz.

Una interfaz se declara como ***public interface*** *Nombre {...}*. Suele incluir un conjunto de cabeceras de métodos abstractos que deben ser implementados (todos) en la clase que implemente dicha interfaz. **IMPORTANTE**: una interfaz NO tiene atributos, aunque sí puede incluir definiciones constantes públicas (static final).

NOTA: una misma clase puede implementar más de una interfaz → *herencia múltiple de interfaces*

Ejemplo:

```
public interface Figura {   // Define una interfaz
   double area();
   double perimetro();
}

public class EspejoCircular implements Figura {  // Implementa una interfaz
   private double radio;
   ..............
   public double area() { return Math.PI*radio*radio; }
   public double perimetro() { return 2*Math.PI*radio; }
}

public class Foto implements Figura { // Implementa una interfaz
   private double lado1, lado2;
   ..............
   public double area() { return lado1*lado2; }
   public double perimetro() { return 2*(lado1+ lado2); }
}
```

Se pueden declarar referencias a objetos que implementen una cierta interfaz. Esto permite definir un método que sea aplicable a todos los objetos de clases que implementen una cierta interfaz.

```
double totalArea( Figura v[] ) {
   double t=0;                                  // Array de instancias de clases que
   for (int i=0; i<v.length; i++)                    // implementan la interfaz figura
   t += v[i].area(); // enlace dinámico
   return t;
}
```

Para poder ordenar un array es necesario saber cómo comparar 2 elementos para averiguar cuál es mayor o si son iguales. Java proporciona la interfaz **Comparable<T>** Es necesario que una clase A que quiera poder comparar objetos de la propia clase A implemente esa interfaz así: ***public class A implements Comparable<A> {…}*** Esto obliga a tener implementado el método ***public int compareTo(A parametro)*** que retorna < 0 si this es menor que parametro, > 0 si this es mayor que parametro e = 0 si son iguales.

NOTA: El método de Java que permite ordenar un array es: **Arrays.sort (Array de objetos)**

Ejercicio: → PRÓXIMO DÍA
Dada una clase Persona y una clase FechaComparable que implementa la interfaz **Comparable<FechaComparable>**, modificar la clase Persona de forma que implemente la interfaz **Comparable<Persona>** (la comparación se hará teniendo en cuenta sólo la edad).

Escribir, además, un programa principal que ejecute:
→ Crear un array de 5 personas y ordenar el array (con **Arrays.sort**)

Diferencias básicas y más destacables entre:

Interfaces	Clases Abstractas
Una clase puede implementar múltiples interfaces	Una clase solo puede extender a otra clase (un único padre)
Las clases que lo implementan no tienen porqué estar relacionadas entre sí	La clase que extienda a la clase abstracta tiene una relación es-un con ella y es-hermano con otras subclases de la abstracta
Solo permite constantes públicas (static final)	Permite atributos de clase y de instancia
Todos los métodos son públicos	Sus métodos no tienen por qué ser públicos