7.D. Clases abstractas.

1. Clases abstractas.

1.2. Métodos abstractos.

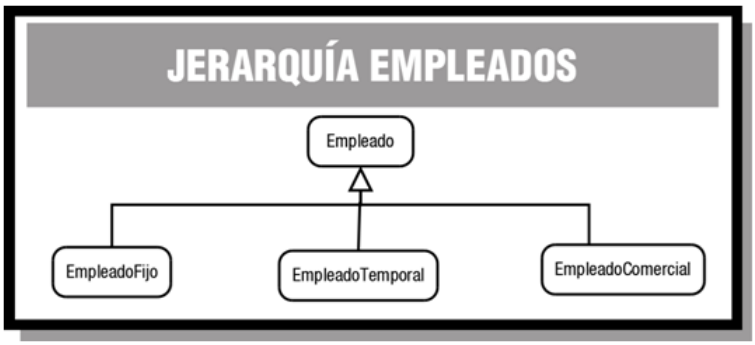
Un **método abstracto** es un método cuya implementación no se define, sino que se declara únicamente su **interfaz** (cabecera) para que su cuerpo sea implementado más adelante en una **clase derivada**.

Un método se declara como abstracto mediante el uso del modificador **abstract** (como en las **clases abstractas**):

```
[modificador_acceso] abstract  <tipo> <nombreMetodo> ([parámetros]) [excepciones];
```

Estos métodos tendrán que ser **obligatoriamente redefinidos** (en realidad "definidos", pues aún no tienen contenido) en las **clases derivadas**. Si en una **clase derivada** se deja algún **método abstracto sin implementar**, esa **clase derivada** será también una **clase abstracta**.

Cuando una clase contiene un **método abstracto** tiene que declararse como **abstracta** obligatoriamente.



Imagina que tienes una clase **Empleado** genérica para diversos tipos de empleado y tres **clases derivadas**: **EmpleadoFijo** (tiene un salario fijo más ciertos complementos), **EmpleadoTemporal** (salario fijo más otros complementos diferentes) y **EmpleadoComercial** (una parte de salario fijo y unas comisiones por cada operación). La clase **Empleado** podría contener un **método abstracto calcularNomina**, pues sabes que se método será necesario para cualquier tipo de empleado (todo empleado cobra una nómina). Sin embargo el cálculo en sí de la nómina será diferente si se trata de un empleado fijo, un empleado temporal o un empleado comercial, y será dentro de las clases especializadas de **Empleado** (**EmpleadoFijo**, **EmpleadoTemporal**, **EmpleadoComercial**) donde se implementen de manera específica el cálculo de las mismas.

Debes tener en cuenta al trabajar con métodos abstractos:

- Un **método abstracto** implica que la clase a la que pertenece tiene que ser **abstracta**, pero eso no significa que todos los métodos de esa clase tengan que ser abstractos.
- Un **método abstracto** no puede ser **privado** (no se podría implementar, dado que las **clases derivadas** no tendrían acceso a él).
- Los **métodos abstractos** no pueden ser **estáticos**, pues los **métodos estáticos** no pueden ser redefinidos (y los **métodos abstractos** necesitan ser redefinidos).

Autoevaluación

Los métodos de una clase abstracta tienen que ser también abstractos. ¿Verdadero o Falso?

- ○ Verdadero
- ○ Falso

Ejercicio resuelto

Basándote en la jerarquía de clases `Persona`, `Alumno`, `Profesor`, crea un método abstracto llamado mostrar para la clase `Persona`.

Dependiendo del tipo de persona (alumno o profesor) el método mostrar tendrá que mostrar unos u otros datos personales (habrá que hacer implementaciones específicas en cada clase derivada).

Una vez hecho esto, implementa completamente las tres clases (con todos sus atributos y métodos) y utilízalas en un pequeño programa de ejemplo que cree un objeto de tipo `Alumno` y otro de tipo `Profesor`, los rellene con información y muestre esa información en la pantallla a través del método `mostrar`.

Solución:

Dado que el método mostrar no va a ser implementado en la clase `Persona`, será declarado como abstracto y no se incluirá su implementación:

```
protected abstract void mostrar ();
```

Recuerda que el simple hecho de que la clase Persona contenga un método abstracto hace que sea clase sea abstracta (y deberá indicarse como tal en su declaración): `public abstract class Persona`.

En el caso de la clase `Alumno` habrá que hacer una implementación específica del método mostrar y lo mismo para el caso de la clase `Profesor`.

1. **Método `mostrar` para la clase `Alumno`.**

```
// Redefinición del método abstracto mostrar en la clase Alumno

public void mostrar () {

    SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");

    String Stringfecha= formatoFecha.format(this.fechaNacim.getTime());

    System.out.printf ("Nombre: %s\n", this.nombre);

    System.out.printf ("Apellidos: %s\n", this.apellidos);

    System.out.printf ("Fecha de nacimiento: %s\n", Stringfecha);

    System.out.printf ("Grupo: %s\n", this.grupo);

    System.out.printf ("Grupo: %5.2f\n", this.notaMedia);

}
```

2. **Método `mostrar` para la clase `Profesor`.**

```
// Redefinición del método abstracto mostrar en la clase Profesor

public void mostrar () {

    SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");

    String Stringfecha= formatoFecha.format(this.fechaNacim.getTime());

    System.out.printf ("Nombre: %s\n", this.nombre);

    System.out.printf ("Apellidos: %s\n", this.apellidos);

    System.out.printf ("Fecha de nacimiento: %s\n", Stringfecha);

    System.out.printf ("Especialidad: %s\n", this.especialidad);

    System.out.printf ("Salario: %7.2f euros\n", this.salario);

}
```

3. **Programa de ejemplo de uso.**

Un pequeño programa de ejemplo de uso del método mostrar en estas dos clases podría ser:

```
// Declaración de objetos
        Alumno alumno;
        Profesor profe;

        // Creación de objetos (llamada a constructores)
        alumno= new Alumno ("Juan", "Torres", new GregorianCalendar (1990, 10, 6), "1DAM-B", 7.5);
        profe= new Profesor ("Antonio", "Campos", new GregorianCalendar (1970, 8, 15), "Mates", 2000);

        // Utilización del método mostrar
        alumno.mostrar();
        profesor.mostrar();
```

Aquí tendríamos todo el código:

Persona.java

```
/*
 * Clase Persona
 */
package ejemploclaseabstractapersona;

import java.util.GregorianCalendar;

/**
 * Clase abstracta Persona
 */
public abstract class Persona {
    protected String nombre;
    protected String apellidos;
    protected GregorianCalendar fechaNacim;

    // Constructores
    // -------------

    // Constructor
    public Persona (String nombre, String apellidos, GregorianCalendar fechaNacim) {
       this.nombre= nombre;
       this.apellidos= apellidos;
       this.fechaNacim= (GregorianCalendar) fechaNacim.clone();
    }
```

```
    // Método getNombre
    protected String getNombre (){
       return nombre;
    }

    // Método getApellidos
    protected String getApellidos (){
       return apellidos;
    }

    // Método getFechaNacim
    protected GregorianCalendar getFechaNacim (){
       return this.fechaNacim;
    }

    // Método setNombre
    protected void setNombre (String nombre){
       this.nombre= nombre;
    }

    // Método setApellidos
    protected void setApellidos (String apellidos){
       this.apellidos= apellidos;
    }

    // Método setFechaNacim
    protected void setFechaNacim (GregorianCalendar fechaNacim){
       this.fechaNacim= fechaNacim;
    }

    // Métodos abstractos

    // Método mostrar
    protected abstract void mostrar (); // No se define: es abstracto. Ya lo harán sus subclases.

}
```

Alumno.java

```
/*
 * Clase Alumno.
 */
package ejemploclaseabstractapersona;

import java.util.*;
import java.text.*;

/**
 *
 * Clase Alumno
 */
public class Alumno extends Persona  {
        protected String grupo;
        protected double notaMedia;



        // Constructor
        // -----------
        public Alumno (String nombre, String apellidos, GregorianCalendar fechaNacim, String grupo, double notaMedia) {
           super (nombre, apellidos, fechaNacim);
           this.grupo= grupo;
           this.notaMedia= notaMedia;
        }



        // Método getGrupo
        public String getGrupo (){
           return grupo;
        }

        // Método getNotaMedia
        public double getNotaMedia (){
           return notaMedia;
        }




        // Método setGrupo
```

```
    public void setGrupo (String grupo){
        this.grupo= grupo;
    }

    // Método setNotaMedia
    public void setNotaMedia (double notaMedia){
        this.notaMedia= notaMedia;
    }

    // Redefinición de métodos abstractos heredados
    @Override
    public void mostrar () {
        SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");
        String Stringfecha= formatoFecha.format(this.fechaNacim.getTime());

        System.out.printf ("Nombre: %s\n", this.nombre);
        System.out.printf ("Apellidos: %s\n", this.apellidos);
        System.out.printf ("Fecha de nacimiento: %s\n", Stringfecha);
        System.out.printf ("Grupo: %s\n", this.grupo);
        System.out.printf ("Nota media: %5.2f\n", this.notaMedia);
    }
}
```

Profesor.java

```
/*
 * Clase Profesor
 */
package ejemploclaseabstractapersona;

/**

 */
import java.util.*;
import java.text.*;

/**
 *
 * Clase Profesor
 */
```

```
public class Profesor extends Persona {
    String especialidad;
    double salario;

  // Constructor
  // -----------

  public Profesor (String nombre, String apellidos, GregorianCalendar fechaNacim, String especialidad, double salario) {
    super (nombre, apellidos, fechaNacim);
    this.especialidad= especialidad;
    this.salario= salario;
  }



  // Método getEspecialidad
  public String getEspecialidad (){
    return especialidad;
  }

  // Método getSalario
  public double getSalario (){
    return salario;
  }



  // Método setSalario
  public void setSalario (double salario){
    this.salario= salario;
  }

  // Método setESpecialidad
  public void setESpecialidad (String especialidad){
    this.especialidad= especialidad;
  }

  // Redefinición de métodos abstractos heredados
  @Override
  public void mostrar () {
    SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");
```

```
        String Stringfecha= formatoFecha.format(this.fechaNacim.getTime());

        System.out.printf ("Nombre: %s\n", this.nombre);
        System.out.printf ("Apellidos: %s\n", this.apellidos);
        System.out.printf ("Fecha de nacimiento: %s\n", Stringfecha);
        System.out.printf ("Especialidad: %s\n", this.especialidad);
        System.out.printf ("Salario: %7.2f euros\n", this.salario);
    }
}
```

EjemploClaseAbstractaPersona.java

```
/**
 *  Ejemplo de uso de clases abstractas y métodos abstractos
 */
package ejemploclaseabstractapersona;
import java.util.GregorianCalendar;

/**
 * Programa principal
 */
public class EjemploClaseAbstractaPersona {

    /**
     *  Ejemplo de uso de clases abstractas y métodos abstractos
     */
    public static void main(String[] args) {
        // Declaración de objetos de las clases Persona, Profesor y Alumno
        Persona pers1, pers2;
        Alumno al1, al2;
        Profesor prof1, prof2;

        //pers1= new Persona (); // Error: una clase abstracta no puede ser instanciada

        al1= new Alumno ("Juan", "Torres", new GregorianCalendar (1990, 10, 6), "1DAM-B", 7.5);
        prof1= new Profesor  ("Antonio", "Campos", new GregorianCalendar (1970, 8, 15), "Electricidad", 2000);

        // Llamada a métodos abstractos en la clase Persona
```

```
        // Pero heredados y definidos en las clases Profesor y en Alummno
        al1.mostrar();
        prof1.mostrar();
    }
}
```