Programación

Act08_Herencia

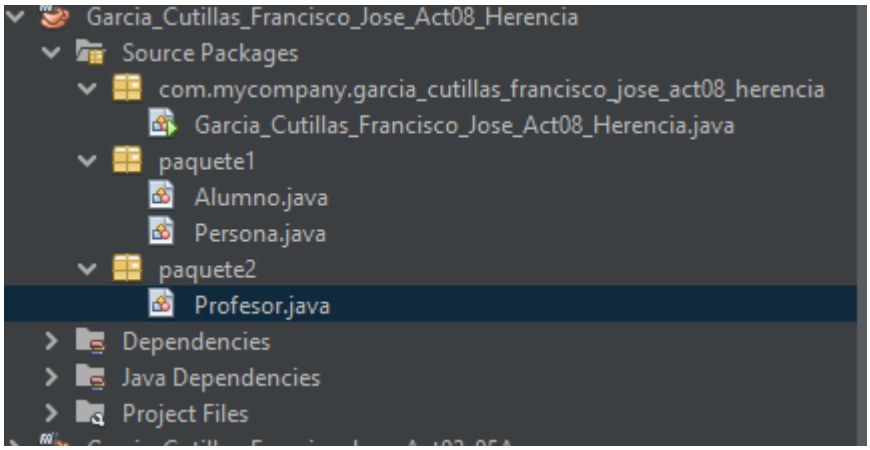
Francisco José García Cutillas | 1FPGS_DAM

Índice

Ejercicio A01

Aplicación para un instituto.

- **Crea un nuevo proyecto que esté formado por dos paquetes: paquete1 y paquete2 (adicionales al paquete que haya creado el proyecto).**
- **Dentro del paquete1, vamos a meter dos ficheros java: uno que guardará las clases Persona y Alumno y segundo paquete que guardará la clase Profesor.**



- **La clase Persona tendrá las propiedades dni, nombre, apellidos y Fecha de Nacimiento que tendrán el tipo que tú consideres.**
    - **Los atributos deben ser privados y tener todos un getter y un setter.**

```java
package paquete1;

import java.time.LocalDate;
import java.time.Period;
import java.util.Objects;
import utilidades.Dni;

/**
 * Clase Persona
 *
 * @author Fran
 */
public class Persona {

    private static final String CODIGO_CENTRO = "44443333";
    private String dni;
    private String nombre;
    private String apellidos;
    private LocalDate fechaNacimiento;

    /**
     * Constructor de la clase. Crea objetos de tipo Persona
     *
     * @param _dni Dni de la persona
     * @param _nombre Nombre de la persona
     * @param _apellidos Apellidos de la persona
     * @param _fechaNacimiento Fecha de nacimiento de la persona
     */
    public Persona(String _dni, String _nombre, String _apellidos, LocalDate _fechaNacimiento) {

        //Comprueba que el dni sea correcto
        if (Dni.compruebaDni( dni: _dni)) {

            this.dni = _dni;

        } else {

            this.dni = null;
            System.out.println( x: "Dni incorrecto");

        }

        this.nombre = _nombre;
        this.apellidos = _apellidos;
        this.fechaNacimiento = _fechaNacimiento;

    }
```

```java
/**
 * Método para obtener el dni de la persona
 *
 * @return Dni de la persona de tipo String
 */
public String getDni() {
    return dni;
}

/**
 * Método para establecer el dni de la persona
 *
 * @param dni Nuevo dni a establecer
 */
public void setDni(String dni) {
    this.dni = dni;
}

/**
 * Método para obtener el nombre de la persona
 *
 * @return Devuelve el nombre de la persona de tipo String
 */
public String getNombre() {
    return nombre;
}

/**
 * Método para establecer el nombre de la persona
 *
 * @param nombre Nombre a establecer a la persona de tipo String
 */
public void setNombre(String nombre) {
    this.nombre = nombre;
}

/**
 * Método para obtener los apellidos de la persona
 *
 * @return Apellidos de la persona de tipo String
 */
public String getApellidos() {
    return apellidos;
}
```

```java
/**
 * Método para establecer apellidos a la persona
 *
 * @param apellidos Apellidos a establecer a la persona de tipo String
 */
public void setApellidos(String apellidos) {
    this.apellidos = apellidos;
}

/**
 * Método para obtener la fecha de nacimiento de la persona
 *
 * @return Fecha de nacimiento de la persona de tipo LocalDate (aaaa-mm-dd)
 */
public LocalDate getFechaNacimiento() {
    return fechaNacimiento;
}

/**
 * Método para establecer la fecha de nacimiento de la persona
 *
 * @param fechaNacimiento Fecha de nacimiento de la persona de tipo
 * LocalDate y formato (aaaa-mm-dd)
 */
public void setFechaNacimiento(LocalDate fechaNacimiento) {
    this.fechaNacimiento = fechaNacimiento;
}
```

- **Debes desarrollar el método que devuelva el nombre completo de la persona en un String.**

```java
/**
 * Método para obtener el nombre y apellidos de la persona
 *
 * @return Devuelve el nombre y apellidos de la persona en una cadena
 */
public String getNombreCompleto() {

    return getNombre() + " " + getApellidos();

}
```

- **Todas las personas comparten el mismo código de centro. Debemos crear un atributo para ello del tipo que creas necesario, así como un método para consultar este código. En nuestro caso el valor del código debe ser 44443333.**

```java
public class Persona {

    private static final String CODIGO_CENTRO = "44443333";
```

- **Crea el método getEdad que devolverá un entero correspondiente a la edad. Esta edad se calculará a partir de la fecha de nacimiento (NO se debe tener un atributo edad, se calcula y se devuelve en el método)**

```java
/**
 * Método para obtener la edad de la persona
 *
 * @return Edad de la persona de tipo int
 */
public int getEdad() {

    LocalDate actual = LocalDate.now();
    Period diferencia = Period.between( startDateInclusive: this.fechaNacimiento, endDateExclusive: actual);

    return diferencia.getYears();

}
```

- **Crea el método toString y equals correspondiente (dos personas son iguales si tienen el mismo dni)**

```java
/**
 * Método toString() de la clase
 *
 * @return Devuelve la información de la persona en una cadena
 */
@Override
public String toString() {
    return "dni: " + dni + ", nombre: " + nombre + ", apellidos: "
            + apellidos + ", fecha de nacimiento: " + fechaNacimiento;
}

/**
 * Método equals de la clase
 *
 * @param obj Objeto a comparar
 * @return Devuelve true si dos personas tienen el mismo dni, false en caso
 * contrario
 */
@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    Persona other = (Persona) obj;
    return Objects.equals( a: this.dni,  b: other.dni);
}
```

- **La clase Alumno heredará de Persona y tendrá adicionalmente el nre y un atributo mail que una vez establecido no se podrá cambiar (lo indicamos de tipo final).**
    - **Encapsulamiento básico: atributos privados y constructor público.**

```java
package paquete1;

import java.time.LocalDate;
import java.util.Objects;

/**
 * Clase Alumno que hereda de Persona
 *
 * @author Fran
 */
public class Alumno extends Persona {

    private String nre;
    private final String MAIL;

    /**
     * Constructor para crear objetos de tipo Alumno
     *
     * @param _dni Dni del alumno
     * @param _nombre Nombre del alumno
     * @param _apellidos Apellidos del alumno
     * @param _fechaNacimiento Fecha de nacimiento del alumno de tipo LocalDate
     * @param _nre Nre del alumno
     * @param _mail Mail del alumno
     */
    public Alumno(String _dni, String _nombre, String _apellidos, LocalDate _fechaNacimiento, String _nre, String _mail) {

        super(_dni, _nombre, _apellidos, _fechaNacimiento);
        this.nre = _nre;
        this.MAIL = _mail;

    }
```

```java
/**
 * Método para obtener el nre del alumno
 *
 * @return Devuelve el nre del alumno de tipo String
 */
public String getNre() {
    return nre;
}

/**
 * Método para establecer un nuevo nre al alumno
 *
 * @param nre Nre que se vaya a establecer de tipo String
 */
public void setNre(String nre) {
    this.nre = nre;
}
```

- **Crea los métodos toString y equals (un alumno es igual a otro si tiene el mismo nre).**

```java
/**
 * Método toString() de la clase
 *
 * @return Información del alumno en forma de cadena
 */
@Override
public String toString() {

    return super.toString() + ", nre: " + this.nre + ", mail: " + this.MAIL;

}

/**
 * Método equals de la clase
 *
 * @param obj Objeto a comparar
 * @return True si dos objetos tienen el mismo Nre, false en caso contrario
 */
@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    Alumno other = (Alumno) obj;
    return Objects.equals( a: this.nre, b: other.nre);
}
```

- **La clase Profesor tendrá el atributo especialidad.**
  - **Encapsulamiento básico.**

```java
package paquete2;

import java.time.LocalDate;
import java.util.Objects;
import java.util.Scanner;
import paquete1.Persona;

/**
 * Clase Profesor que hereda de Persona
 *
 * @author Fran
 */
public class Profesor extends Persona {

    private String especialidad;

    /**
     * Constructor para crear objetos de tipo Profesor
     *
     * @param _dni Dni del profesor
     * @param _nombre Nombre del profesor
     * @param _apellidos Apellidos del profesor
     * @param _fechaNacimiento Fecha de nacimiento del Profesor de tipo
     * LocalDate
     * @param _especialidad Especialidad del profesor
     */
    public Profesor(String _dni, String _nombre, String _apellidos, LocalDate _fechaNacimiento, String _especialidad) {

        super(_dni, _nombre, _apellidos, _fechaNacimiento);
        this.especialidad = _especialidad;

    }
```

- **Crear un método que sea cambiarEspecialidad que deberá pedir por terminal la especialidad y que la cambie para su objeto.**

```java
/**
 * Método para cambiar la especialidad del profesor. Ésta se introduce a
 * través de la consola
 */
public void cambiarEspecialidad() {

    Scanner sc = new Scanner( source: System.in);
    System.out.println( x: "Introduce nueva especialidad:");
    this.especialidad = sc.nextLine();

}
```

- **Crea los métodos toString y equals (un profesor es igual a otro si tiene el mismo dni).**

```java
/**
 * Método toString() de la clase
 *
 * @return Devuelve en una cadena los datos del profesor
 */
@Override
public String toString() {

    return super.toString() + ", especialidad: " + this.especialidad;

}

/**
 * Método equals de la clase
 *
 * @param obj Objeto a comparar
 * @return True si el Dni de los dos objetos coincide, false en caso
 * contrario
 */
@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    Profesor other = (Profesor) obj;
    return Objects.equals( a: this.getDni(), b: other.getDni());
}
```

- **En la clase principal crea un método en el que cree varios objetos de cada tipo (Alumno y Profesor) y en el que llames a los distintos métodos.**

```
public static void creaAlumnoProfesor() {

    Alumno a1 = new Alumno( _dni: "11111111A", _nombre: "Juan", _apellidos: "Rodríguez Carretero",
            _fechaNacimiento: LocalDate.of( year: 2000, month: 10, dayOfMonth: 9), _nre: "111111", _mail: "juan.r@gmail.com");
    Alumno a2 = new Alumno( _dni: "22222222B", _nombre: "Antonio", _apellidos: "Ruiz Martínez",
            _fechaNacimiento: LocalDate.of( year: 1998, month: 1, dayOfMonth: 25), _nre: "2222222", _mail: "antonio.r@gmail.com");
    Alumno a3 = new Alumno( _dni: "33333333C", _nombre: "Pepi", _apellidos: "García Domínguez",
            _fechaNacimiento: LocalDate.of( year: 2001, month: 5, dayOfMonth: 28), _nre: "3333333", _mail: "pepi.g@gmail.com");

    Profesor p1 = new Profesor( _dni: "44444444D", _nombre: "Cecilia", _apellidos: "Palazón Giménez",
            _fechaNacimiento: LocalDate.of( year: 1988, month: 6, dayOfMonth: 12), _especialidad: "Inglés");
    Profesor p2 = new Profesor( _dni: "55555555E", _nombre: "Evaristo", _apellidos: "Buendía Gil",
            _fechaNacimiento: LocalDate.of( year: 1979, month: 8, dayOfMonth: 21), _especialidad: "Sistemas");
    Profesor p3 = new Profesor( _dni: "66666666F", _nombre: "María", _apellidos: "García Pérez",
            _fechaNacimiento: LocalDate.of( year: 1988, month: 6, dayOfMonth: 12), _especialidad: "Programación");

    System.out.println("nre A1: " + a1.getNre());
    System.out.println("dni A1: " + a1.getDni());
    System.out.println("Datos A1: " + a1.toString());
    System.out.println("Edad A2: " + a2.getEdad());
    System.out.println("Código centro A1: " + a1.getCodigoCentro());
    System.out.println("Código centro A2: " + a2.getCodigoCentro());
    System.out.println("Nombre completo A2: " + a2.getNombreCompleto());
    System.out.println("¿Es alumno A1 igual que alumno A2?: " + a1.equals( obj: a2));
    System.out.println("¿Es alumno A1 igual que alumno A1?: " + a1.equals( obj: a1));

    System.out.println("¿Es profesor P1 igual que profesor P2?: " + p1.equals( obj: p2));
    System.out.println("¿Es profesor P1 igual que profesor P1?: " + p1.equals( obj: p1));
    System.out.println("Datos profesor P1: " + p1.toString());
    p1.cambiarEspecialidad();
    System.out.println("Especialidad P1: " + p1.getEspecialidad());
```

- **Crea un array de objetos de tipo Persona y crea un método que pasándole ese array muestre por pantalla los que son de un tipo o de otro (Persona, Alumno y Profesor). Debes utiliza getClass() y getName()**

```
Persona arrayPersonas[] = {a1, p2, a3, p3};
tipoObjeto( entrada: arrayPersonas);
```

```java
    /**
     * Método que muestra por pantalla la cantidad de objetos creados de cada
     * clase (Persona, Alumno o Profesor)
     *
     * @param entrada Recibe un array de Persona, Alumno o Profesor
     */
    public static void tipoObjeto(Persona[] entrada) {

        Persona p = new Persona( _dni: "llllllllA",  _nombre: "aaaa",  _apellidos: "aaaa aaaa",
                _fechaNacimiento: LocalDate.of( year: 2000,  month: 1,  dayOfMonth: 1));
        Alumno a = new Alumno( _dni: "llllllllA",  _nombre: "aaaa",  _apellidos: "aaaa aaaa",
                _fechaNacimiento: LocalDate.of( year: 2000,  month: 1,  dayOfMonth: 1),  _nre: "lllllll",  _mail: "aaaa");
        Profesor pro = new Profesor( _dni: "llllllllA",  _nombre: "aaaa",  _apellidos: "aaaa aaaa",
                _fechaNacimiento: LocalDate.of( year: 2000,  month: 1,  dayOfMonth: 1),  _especialidad: "Especialidad");

        int persona = 0;
        int alumno = 0;
        int profesor = 0;

        for (int i = 0; i < entrada.length; i++) {

            if (entrada[i].getClass().getName().equals( anObject: p.getClass().getName())) {

                persona++;

            } else if (entrada[i].getClass().getName().equals( anObject: a.getClass().getName())) {

                alumno++;

            } else if (entrada[i].getClass().getName().equals( anObject: pro.getClass().getName())) {

                profesor++;

            }

        }

        System.out.println( x: "*** Tipos de objetos en el array ***");
        System.out.println("Objetos de tipo Persona: " + persona);
        System.out.println("Objetos de tipo Alumno: " + alumno);
        System.out.println("Objetos de tipo Profesor: " + profesor);

    }
}
```

```
--< com.mycompany:Garcia_Cutillas_Francisco_Jose_Act08_Herencia_Ej01 >--
Building Garcia_Cutillas_Francisco_Jose_Act08_Herencia_Ej01 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Garcia_Cutillas_Francisco_Jose_Act08_Herencia_Ej01 ---
nre A1: 1111111
dni A1: llllllllA
Datos A1: dni: llllllllA, nombre: Juan, apellidos: Rodriguez Carretero, fecha de nacimiento: 2000-10-09, nre: 1111111, mail: juan.r@gmail.com
Edad A2: 25
Código centro A1: 44443333
Código centro A2: 44443333
Nombre completo A2: Antonio Ruiz Martinez
¿Es alumno A1 igual que alumno A2?: false
¿Es alumno A1 igual que alumno A1?: true
¿Es profesor P1 igual que profesor P2?: false
¿Es profesor P1 igual que profesor P1?: true
Datos profesor P1: dni: 44444444D, nombre: Cecilia, apellidos: Palazón Giménez, fecha de nacimiento: 1988-06-12, especialidad: Inglés
Introduce nueva especialidad:
Programación
Especialidad P1: Programaci?n

*** Tipos de objetos en el array ***
Objetos de tipo Persona: 0
Objetos de tipo Alumno: 2
Objetos de tipo Profesor: 2
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  5.950 s
Finished at: 2023-03-23T17:23:10+01:00
------------------------------------------------------------------------
```

Ejercicio A02

Instituto versión 2.

- **Haz una copia del proyecto de la actividad 1.**
- **La clase Persona va a definir un método que se va a llamar mostrarInfo que NO va a tener código, es decir, va a ser abstracta. Esta clase la tenemos que implementar en cada una de las subclases. Este cambio puede suponer diferentes cambios en el resto de las clases que has realizado. Realiza todos los cambios que consideres comentando el motivo de porque se realizan los mismos.**

```java
/**
 * Clase Persona
 * @author Fran
 */

//Se ha tenido que hacer la clase abstracta, debido a que contiene un método abstracto
public abstract class Persona {

    private static final String CODIGO_CENTRO = "44443333";
    private String dni;
    private String nombre;
    private String apellidos;
    private LocalDate fechaNacimiento;
```

```java
 //Se crea un método abstracto que debemos desarrollar en las subclases que heredan de Persona
 /**
  * Método mostrarInfo de la persona
  */
 public abstract void mostrarInfo();
```

```java
//Para que no dé error y no tener que hacer la clase Alumno abstracta, debemos desarrollar el método
//mostrar info de su clase Padre

/**
 * Método mostrarInfo(). Muestra en forma de lista la información del alumno
 */
@Override
public void mostrarInfo(){

    System.out.println(x:"*** Información del Alumno ***");
    System.out.println("Dni: " + this.getDni());
    System.out.println("Nombre y apellidos: " + this.getNombreCompleto());
    System.out.println("Fecha nacimiento: " + this.getFechaNacimiento());
    System.out.println("Nre: " + this.getNre());
    System.out.println("Mail: " + MAIL);
}
```

```java
//Para que no dé error y no tener que hacer la clase Profesor abstracta, debemos desarrollar el método
//mostrar info de su clase Padre

/**
 * Método mostrarInfo de profesor. Muestra en forma de lista todos los atributos del objeto
 */
@Override
public void mostrarInfo(){

    System.out.println( x: "*** Información del Profesor ***");
    System.out.println("Dni: " + this.getDni());
    System.out.println("Nombre y apellidos: " + this.getNombreCompleto());
    System.out.println("Fecha nacimiento: " + this.getFechaNacimiento());
    System.out.println("Especialidad: " + this.getEspecialidad());

}
```

```java
package com.mycompany.garcia_cutillas_francisco_jose_act08_herencia;

import java.time.LocalDate;
import paquete1.Alumno;
import paquete1.Persona;
import paquete2.Profesor;

/**
 *
 * @author Fran
 */
public class Garcia_Cutillas_Francisco_Jose_Act08_Herencia {

    public static void main(String[] args) {

        creaAlumnoProfesor();

    }

    /**
     * Método utilizado para crear objetos de tipo Alumno y Profesor
     */
    public static void creaAlumnoProfesor() {

        Alumno a1 = new Alumno( _dni: "11111111A", _nombre: "Juan", _apellidos: "Rodriguez Carretero",
                _fechaNacimiento: LocalDate.of( year: 2000, month: 10, dayOfMonth: 9), _nre: "1111111", _mail: "juan.r@gmail.com");
        Alumno a2 = new Alumno( _dni: "22222222B", _nombre: "Antonio", _apellidos: "Ruiz Martinez",
                _fechaNacimiento: LocalDate.of( year: 1998, month: 1, dayOfMonth: 25), _nre: "2222222", _mail: "antonio.r@gmail.com");
        Alumno a3 = new Alumno( _dni: "33333333C", _nombre: "Pepi", _apellidos: "Garcia Domínguez",
                _fechaNacimiento: LocalDate.of( year: 2001, month: 5, dayOfMonth: 28), _nre: "3333333", _mail: "pepi.g@gmail.com");

        Profesor p1 = new Profesor( _dni: "44444444D", _nombre: "Cecilia", _apellidos: "Palazón Giménez",
                _fechaNacimiento: LocalDate.of( year: 1988, month: 6, dayOfMonth: 12), _especialidad: "Inglés");
        Profesor p2 = new Profesor( _dni: "55555555E", _nombre: "Evaristo", _apellidos: "Buendia Gil",
                _fechaNacimiento: LocalDate.of( year: 1979, month: 8, dayOfMonth: 21), _especialidad: "Sistemas");
        Profesor p3 = new Profesor( _dni: "66666666F", _nombre: "Maria", _apellidos: "Garcia Pérez",
                _fechaNacimiento: LocalDate.of( year: 1988, month: 6, dayOfMonth: 12), _especialidad: "Programación");
```

```java
        System.out.println("nre A1: "+ a1.getNre());
        System.out.println("dni A1: "+ a1.getDni());
        System.out.println("Datos A1: "+ a1.toString());
        System.out.println("Edad A2: "+ a2.getEdad());
        System.out.println("Código centro A1: "+ a1.getCodigoCentro());
        System.out.println("Código centro A2: "+ a2.getCodigoCentro());
        System.out.println("Nombre completo A2: "+ a2.getNombreCompleto());
        System.out.println("¿Es alumno A1 igual que alumno A2?: "+ a1.equals( obj: a2));
        System.out.println("¿Es alumno A1 igual que alumno A1?: "+ a1.equals( obj: a1));

        System.out.println("¿Es profesor P1 igual que profesor P2?: "+ p1.equals( obj: p2));
        System.out.println("¿Es profesor P1 igual que profesor P1?: "+ p1.equals( obj: p1));
        System.out.println("Datos profesor P1: "+ p1.toString());
        p1.cambiarEspecialidad();
        System.out.println("Especialidad P1: "+ p1.getEspecialidad());

        Persona arrayPersonas[] = {a1, p2, a2, a3, p3};
        tipoObjeto( entrada: arrayPersonas);
        p1.mostrarInfo();

    }

    /**
     * Método para determinar de qué clase son los objetos de un array que recibe por parámetro
     * @param entrada Array de tipo Persona
     */
    public static void tipoObjeto(Persona[] entrada){

        int tipoPersona = 0;
        int tipoAlumno = 0;
        int tipoProfesor = 0;

        for (int i = 0; i < entrada.length; i++) {

            if (entrada[i].getClass().getSimpleName().equals( anObject: "Persona")) {

                tipoPersona++;

            } else if (entrada[i].getClass().getSimpleName().equals( anObject: "Alumno")) {

                tipoAlumno++;

            } else {

                tipoProfesor++;

            }

        }

        System.out.println( x: "*** Tipos de objetos en el array ***");
        System.out.println("Persona: " + tipoPersona);
        System.out.println("Alumno: " + tipoAlumno);
        System.out.println("Profesor: " + tipoProfesor);

    }
```

```
--< com.mycompany:Garcia_Cutillas_Francisco_Jose_Act08_Herencia_Ej02 >--
Building Garcia_Cutillas_Francisco_Jose_Act08_Herencia_Ej02 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Garcia_Cutillas_Francisco_Jose_Act08_Herencia_Ej02 ---
nre A1: 1111111
dni A1: 11111111A
Datos A1: dni: 11111111A, nombre: Juan, apellidos: Rodríguez Carretero, fecha de nacimiento: 2000-10-09, nre: 1111111, mail: juan.r@gmail.com
Edad A2: 25
Código centro A1: 44443333
Código centro A2: 44443333
Nombre completo A2: Antonio Ruiz Martínez
¿Es alumno A1 igual que alumno A2?: false
¿Es alumno A1 igual que alumno A1?: true
¿Es profesor P1 igual que profesor P2?: false
¿Es profesor P1 igual que profesor P1?: true
Datos profesor P1: dni: 44444444D, nombre: Cecilia, apellidos: Palazón Giménez, fecha de nacimiento: 1988-06-12, especialidad: Inglés
Introduce nueva especialidad:
Bases de datos
Especialidad P1: Bases de datos
*** Tipos de objetos en el array ***
Persona: 0
Alumno: 3
Profesor: 2
*** Información del Profesor ***
Dni: 44444444D
Nombre y apellidos: Cecilia Palazón Giménez
Fecha nacimiento: 1988-06-12
Especialidad: Bases de datos
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  10.436 s
Finished at: 2023-03-23T17:03:46+01:00
------------------------------------------------------------------------
```

Ejercicio A03

Película

- **Crea un nuevo Proyecto de nombre Película**
- **Añade la clase Persona de la actividad A02, pero elimina toda la información relacionada con el código y también con el dni (una Persona en este caso no tiene dni).**

```java
package com.mycompany.garcia_cutillas_francisco_jose_act08_herencia_ej03;

import java.time.LocalDate;
import java.time.Period;


/**
 * Clase Persona
 * @author Fran
 */

//Se ha tenido que hacer la clase abstracta, debido a que contiene un método abstracto
public abstract class Persona {

    private String nombre;
    private String apellidos;
    private LocalDate fechaNacimiento;

    /**
     * Constructor para crear objetos de tipo Persona
     * @param _nombre Nombre de la persona
     * @param _apellidos Apellidos de la persona
     * @param _fechaNacimiento Fecha de nacimiento de la persola de tipo LocalDate
     */
    public Persona (String _nombre, String _apellidos, LocalDate _fechaNacimiento){

        this.nombre = _nombre;
        this.apellidos = _apellidos;
        this.fechaNacimiento = _fechaNacimiento;

    }
```

- **Aunque no se desarrolle el método (abstracto), Persona debe tener un método que indique a qué se dedica esa persona, por lo que devolverá un String. Este método se llamará getRol()**

```java
/**
 * Método que devuelve el rol de una persona
 * @return rol de una persona
 */
public abstract String getRol();
```

- **A partir de la clase Persona debes crear la clase Actor que deberá tener los atributos altura que se indicará en centímetros, su idioma principal que por defecto será castellano si en el constructor no se indica información para este atributo, el teléfono de su representante y un código de actor.**

```java
package com.mycompany.garcia_cutillas_francisco_jose_act08_herencia_ej03;

import java.time.LocalDate;
import java.util.Objects;

/**
 * Clase Actor que hereda de Persona
 * @author Fran
 */
public class Actor extends Persona {

    private int altura;
    private String idiomaPrincipal;
    private String telefonoRepresentante;
    private String codigoActor;

    /**
     * Constructor para crear objetos de tipo Actor
     * @param _nombre Nombre del actor
     * @param _apellidos Apellidos del actor
     * @param _fechaNacimiento Fecha de nacimiento del actor de tipo LocalDate
     * @param _altura Altura del actor en centímetros
     * @param _idiomaPrincipal Idioma principal del actor
     * @param _telefonoRepresentante Teléfono del representante del actor
     * @param _codigoActor Código del actor
     */
    public Actor(String _nombre, String _apellidos, LocalDate _fechaNacimiento, int _altura,
                 String _idiomaPrincipal, String _telefonoRepresentante, String _codigoActor){

        super(_nombre, _apellidos, _fechaNacimiento);
        this.altura = _altura;
        this.idiomaPrincipal = _idiomaPrincipal;
        this.telefonoRepresentante = _telefonoRepresentante;
        this.codigoActor = _codigoActor;

    }
```

```java
/**
 * Constructor para crear objetos de tipo Actor. No recibe el idioma principal, por lo que por defecto
 * asigna castellano como idioma principal del actor
 * @param _nombre Nombre del actor
 * @param _apellidos Apellidos del actor
 * @param _fechaNacimiento Fecha de nacimiento del actor de tipo LocalDate
 * @param _altura Altura del actor en centímetros
 * @param _telefonoRepresentante Teléfono del representante del actor
 * @param _codigoActor Código del actor
 */
public Actor(String _nombre, String _apellidos, LocalDate _fechaNacimiento, int _altura,
             String _telefonoRepresentante, String _codigoActor){

    super(_nombre, _apellidos, _fechaNacimiento);
    this.altura = _altura;
    this.idiomaPrincipal = "Castellano";
    this.telefonoRepresentante = _telefonoRepresentante;
    this.codigoActor = _codigoActor;

}
```

- **Encapsulamiento básico.**
-

```java
/**
 * Método que devuelve la altura del actor
 * @return Devuelve la altura del actor en centímetros
 */
public int getAltura() {
    return altura;
}

/**
 * Método que establece la altura del actor
 * @param altura Altura del actor en centímetros
 */
public void setAltura(int altura) {
    this.altura = altura;
}

/**
 * Método que devuelve el idioma principal del actor
 * @return Idioma principal del actor en tipo String
 */
public String getIdiomaPrincipal() {
    return idiomaPrincipal;
}

/**
 * Método que establece el idioma principal del actor
 * @param idiomaPrincipal Idioma principal
 */
public void setIdiomaPrincipal(String idiomaPrincipal) {
    this.idiomaPrincipal = idiomaPrincipal;
}
```

```java
/**
 * Método que devuelve el teléfono del representante del actor
 * @return Teléfono del representante del actor en tipo String
 */
public String getTelefonoRepresentante() {
    return telefonoRepresentante;
}

/**
 * Método para establecer el teléfono del representante del actor
 * @param telefonoRepresentante Teléfono del representante de tipo String
 */
public void setTelefonoRepresentante(String telefonoRepresentante) {
    this.telefonoRepresentante = telefonoRepresentante;
}

/**
 * Método que devuelve el código del actor
 * @return Devuelve el código del actor en forma de String
 */
public String getCodigoActor() {
    return codigoActor;
}

/**
 * Método que establece el código de actor
 * @param codigoActor Código de actor en forma de String
 */
public void setCodigoActor(String codigoActor) {
    this.codigoActor = codigoActor;
}
```

- **Desarrolla los métodos toString y equals. Dos actores son iguales si tienen el mismo código.**

```java
/**
 * Método toString() de la clase
 * @return Devuelve la información del Actor en forma de cadena
 */
@Override
public String toString(){

    return super.toString() + ", Altura: " + this.getAltura() + ", Idioma principal: " + this.getIdiomaPrincipal() +
            ", Teléfono representante: " + this.getTelefonoRepresentante() + ", Código actor: " + this.getCodigoActor();

}

/**
 * Método equals de la clase. Dos actores son iguales si tienen el mismo código de actor
 * @param obj Objeto a comparar
 * @return True si son el mismo actor, false en el caso contrario
 */
@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    Actor other = (Actor) obj;
    return Objects.equals( a: this.codigoActor, b: other.codigoActor);
}
```

- **El método getRol() devolverá la cadena “Actor”.**

```java
/**
 * Método que devuelve el rol
 * @return Devuelve la cadena "Actor"
 */
@Override
public String getRol(){

    return "Actor";

}
```

- **A partir de la clase Persona debes crear la clase Director, que deberá tener los atributos dni y género de películas favorito (cadena de texto).**

```java
package com.mycompany.garcia_cutillas_francisco_jose_act08_herencia_ej03;

import java.time.LocalDate;
import java.util.Objects;
import utilidades.Dni;

/**
 * Clase Director que hereda de Persona
 * @author Fran
 */
public class Director extends Persona{

    private String dni;
    private String generoFavorito;

    /**
     * Constructor de la clase. Crea objetos de tipo Director
     * @param _nombre Nombre del director
     * @param _apellidos Apellidos del director
     * @param _fechaNacimiento Fecha de nacimiento del director de tipo LocalDate
     * @param _dni Dni del director
     * @param _generoFavorito Género favorito del director
     */
    public Director(String _nombre, String _apellidos, LocalDate _fechaNacimiento, String _dni,
                    String _generoFavorito){

        super(_nombre, _apellidos, _fechaNacimiento);

        if (Dni.compruebaDni( dni: _dni )) {

            this.dni = _dni;

        } else {

            this.dni = null;
            System.out.println( x: "Dni incorrecto");

        }

        this.generoFavorito = _generoFavorito;


    }
```

  - **Encapsulamiento básico.**

```java
/**
 * Método que devuelve el dni del director
 * @return Dni del director de tipo String
 */
public String getDni() {
    return dni;
}

/**
 * Método que devuelve el género favorito del director
 * @return Género favorito del director de tipo String
 */
public String getGeneroFavorito() {
    return generoFavorito;
}

/**
 * Método para establecer el género favorito del director
 * @param generoFavorito Género favorito del director en forma de String
 */
public void setGeneroFavorito(String generoFavorito) {
    this.generoFavorito = generoFavorito;
}
```

- **Desarrolla los métodos toString y equals. Dos directores son iguales si tienen el mismo dni**

```java
/**
 * Método toString() de la clase
 * @return Información del director en una cadena
 */
@Override
public String toString(){

    return super.toString() + ", Dni: " + this.getDni() + ", Género Favorito: " + this.getGeneroFavorito();

}

/**
 * Método equals de la clase. Dos directores son iguales si tienen el mismo dni
 * @param obj Objeto a comparar
 * @return True si coinciden los dos dni, false en el caso contrario
 */
@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    Director other = (Director) obj;
    return Objects.equals( a: this.dni, b: other.dni);
}
```

- **El método getRol() devolverá la cadena “Director”.**

```java
/**
 * Método que devuelve el rol
 * @return Devuelve la cadena "Director" de tipo String
 */
@Override
public String getRol(){

    return "Director";

}
```

- **Desarrolla la clase Película que tendrá los atributos Nombre, año, duración, Un array de hasta 3 actores y un director.**

```java
package com.mycompany.garcia_cutillas_francisco_jose_act08_herencia_ej03;

import java.util.Arrays;
import java.util.Objects;

/**
 * Clase Pelicula
 * @author Fran
 */
public class Pelicula {

    private String nombre;
    private int año;
    private int duracion;
    private Actor[] actores;
    private Director director;

    /**
     * Constructor para construir objetos de tipo Película
     * @param _nombre Nombre de la película
     * @param _año Año de la película
     * @param _duracion Duración de la película en minutos
     * @param _actores Array de actores partícipes en la película de tipo Actor y máximo índice 3
     * @param _director Director de la película de tipo Director
     */
    public Pelicula(String _nombre, int _año, int _duracion, Actor[] _actores, Director _director) {

        this.nombre = _nombre;
        this.año = _año;
        this.duracion = _duracion;

        if (_actores.length <= 3) {

            this.actores = Arrays.copyOf(original: _actores, newLength: 3);

        } else {

            this.actores = Arrays.copyOf(original: _actores, newLength: 3);

        }

        this.director = _director;

    }
```

- **Encapsulamiento básico.**

```java
/**
 * Método que devuelve el nombre de la película
 * @return nombre de la película
 */
public String getNombre() {
    return nombre;
}

/**
 * Método para establecer un nombre a la película
 * @param nombre Nombre para la película
 */
public void setNombre(String nombre) {
    this.nombre = nombre;
}

/**
 * Método que devuelve el año de la película
 * @return año de la película
 */
public int getAño() {
    return año;
}

/**
 * Método para establecer el año de la película
 * @param año Año de la película
 */
public void setAño(int año) {
    this.año = año;
}

/**
 * Método para obtener la duración de la película
 * @return Duración de la película
 */
public int getDuracion() {
    return duracion;
}

/**
 * Método para establecer la duración de la película
 * @param duracion Duración de la película en minutos
 */
public void setDuracion(int duracion) {
    this.duracion = duracion;
}
```

```java
/**
 * Método que devuelve los actores de la película
 * @return Devuelve un array de los actores de la película
 */
public Persona[] getActores() {
    return actores;
}

/**
 * Método que establece los actores de la película
 * @param actores Array de actores de tipo Actor
 */
public void setActores(Actor[] actores) {
    this.actores = actores;
}

/**
 * Método que devuelve el director de la película
 * @return Devuelve el director de la película de tipo Director
 */
public Persona getDirector() {
    return director;
}

/**
 * Método que establece el director de la película
 * @param director Director de tipo Director
 */
public void setDirector(Director director) {
    this.director = director;
}
```

- **Desarrolla los métodos toString y equals. Una película es igual si tiene el mismo nombre y año.**

```java
/**
 * Método toString() de la clase
 * @return Devuelve los datos de la película en un String
 */
@Override
public String toString() {

    return "Nombre película: " + this.getNombre() + ", año: " + this.getAño() + ", duración: " + this.getDuracion()
            + ", actores " + Arrays.toString(a: this.getActores()) + ", director: " + this.getDirector();

}

/**
 * Método equals de la clase. Dos películas son iguales si coinciden en año y nombre
 * @param obj Recibe un objeto con el que queremos comparar
 * @return True si son iguales o false en el caso contrario
 */
@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    Pelicula other = (Pelicula) obj;
    if (this.año != other.año) {
        return false;
    }
    return Objects.equals(a: this.nombre, b: other.nombre);
}
```

- **Desarrolla un método que se llame mostrarInfo() de la película que muestre toda la información de sus participantes: Director y Actor o actores.**

```java
/**
 * Método mostrarInfo() de la clase. Muestra por pantalla la información de la película,
 * así como los datos de su director y actores que participan en la misma
 */
public void mostrarInfo() {

    System.out.println( x: "*** Información de la película ***");
    System.out.println( x: "Director:");

    System.out.println( x: director.toString());

    System.out.println( x: "Actores:");
    for (int i = 0; i < this.actores.length; i++) {

        if (actores[i] != null) {

            System.out.println( x: actores[i].toString());

        }

    }

}
```

- **Desarrolla un método que, recibiendo una Persona, indique si participa en una película. Devolverá una A si participa como Actor, D si participa como Director y N si no participa (Devuelve un char).**

```java
/**
 * Método que devuelve si una persona participa en una película o no. En el caso de participar,
 * nos muestra cuál es su rol dentro de la película
 * @param p Objeto de tipo persona que queremos comprobar si es partícipe de la película
 * @return Devuelve 'D' si la persona introducida es el director de la película, 'A' si es actor ó 'N' si no participa
 * en la misma
 */
public char participa(Persona p) {

    if (p.getClass().getSimpleName().equals( anObject: "Director")) {

        if (this.director.equals( obj: p)) {

            return 'D';

        }

    } else {

        for (int i = 0; i < this.actores.length; i++) {

            if (actores[i] != null) {

                if (actores[i].equals( obj: p)) {

                    return 'A';

                }

            }

        }

    }

    return 'N';

}
```

- **Desarrolla un método que recibe un array de Persona. El método calculará un número aleatorio y seleccionará una de estas Personas. En función de si esta Persona es actor o director la añadirá como director de la película o como uno de los actores de la misma (si ya están ocupados todos los actores de la película, la añadirá como primer actor de la misma).**

  Para este apartado nos hemos ayudado del método estadoActores(), que nos muestra los huecos libres que hay en el array de actores de la película.

```java
/**
 * Método que devuelve el número de huecos que hay en el array de actores de la película
 * @return número de huecos libres en el array de actores de la película
 */
public int estadoActores() {

    int huecosArray = 0;

    for (int i = 0; i < this.actores.length; i++) {

        if (this.actores[i] == null) {

            huecosArray++;

        }

    }

    return huecosArray;

}
```

```java
/**
 * Método que añade una persona de forma aleatoria a la película, contenida en un array que se le pasa como
 * parámetro. Si esta persona es director, se asigna directamente. Si es actor, se añade al array de actores en el primer
 * hueco libre del mismo. Si el array está completo, lo añade en la primera posición, sobreescribiendo el que ya hay.
 * @param p Array de personas
 */
public void addPersona(Persona[] p) {

    int numAleatorio = (int) (Math.random() * (((p.length - 1) - 0) + 1)) + 0;
    Persona personaSeleccionada = p[numAleatorio];

    if (personaSeleccionada.getRol().equals( anObject: "Director")) {

        this.director = (Director) personaSeleccionada;

    } else if (personaSeleccionada.getRol().equals( anObject: "Actor")) {

        if (estadoActores() == 0) {

            this.actores[0] = (Actor) personaSeleccionada;

        } else {

            for (int i = 0; i < this.actores.length; i++) {

                if (this.actores[i] == null) {

                    this.actores[i] = (Actor) personaSeleccionada;
                    break;

                }

            }

        }

    }

}
```

```java
public class Garcia_Cutillas_Francisco_Jose_Act08_Herencia_Ej03 {

    public static void main(String[] args) {

        Actor a1 = new Actor(_nombre: "Antonio", _apellidos: "Banderas", _fechaNacimiento: LocalDate.of(year: 1965, month: 1, dayOfMonth: 1),
                             _altura: 175, _telefonoRepresentante: "666666666", _codigoActor: "1");
        Actor a2 = new Actor(_nombre: "Jason", _apellidos: "Statan", _fechaNacimiento: LocalDate.of(year: 1969, month: 5, dayOfMonth: 7),
                             _altura: 181, _idiomaPrincipal: "Inglés", _telefonoRepresentante: "666666888", _codigoActor: "2");
        Actor a3 = new Actor(_nombre: "Vin", _apellidos: "Diesel", _fechaNacimiento: LocalDate.of(year: 1970, month: 6, dayOfMonth: 10),
                             _altura: 179, _telefonoRepresentante: "666669999", _codigoActor: "3");
        Actor a4 = new Actor(_nombre: "Juan", _apellidos: "Córcoles", _fechaNacimiento: LocalDate.of(year: 1975, month: 8, dayOfMonth: 10),
                             _altura: 177, _telefonoRepresentante: "666669979", _codigoActor: "4");

        Director d1 = new Director(_nombre: "Álex", _apellidos: "De la iglesia", _fechaNacimiento: LocalDate.of(year: 1956, month: 6, dayOfMonth: 10),
                             _dni: "11111111A", _generoFavorito: "Terror");
        Director d2 = new Director(_nombre: "Pepe", _apellidos: "De los muebles", _fechaNacimiento: LocalDate.of(year: 1963, month: 9, dayOfMonth: 20),
                             _dni: "22222222B", _generoFavorito: "Comedia");
        Director d3 = new Director(_nombre: "Juanico", _apellidos: "Colorao", _fechaNacimiento: LocalDate.of(year: 1959, month: 12, dayOfMonth: 30),
                             _dni: "33333333C", _generoFavorito: "Acción");

        Actor arrayActor[] = {a1,a3};
        Actor arrayActor1[] = {a1, a3, a2};
        Pelicula p = new Pelicula(_nombre: "Muerte", _año: 2023, _duracion: 80, _actores: arrayActor, _director: d1);
        Pelicula p1 = new Pelicula(_nombre: "Muerte2", _año: 2023, _duracion: 120, _actores: arrayActor1, _director: d2);

        System.out.println(x: p.participa(p: a1));
        System.out.println(x: p.participa(p: a2));
        System.out.println(x: p.participa(p: d1));
        System.out.println(x: "--------------------------------------");
        p.mostrarInfo();
        p1.mostrarInfo();
        System.out.println(x: "--------------------------------------");
        System.out.println(x: p.equals(obj: p1));

        Persona arrayPersonas[] = {a1, a2, a3, a4, d1, d2, d3};
        System.out.println(x: "--------------------------------------");
        p.addPersona(p: arrayPersonas);
        p.mostrarInfo();
        System.out.println(x: "--------------------------------------");
        p1.addPersona(p: arrayPersonas);
        p1.mostrarInfo();


    }
}
```

```
Building Garcia_Cutillas_Francisco_Jose_Act08_Herencia_Ej03 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Garcia_Cutillas_Francisco_Jose_Act08_Herencia_Ej03 ---
A
N
D
--------------------------------------
*** Información de la película ***
Director:
nombre: Álex, apellidos: De la iglesia, fecha de nacimiento: 1956-06-10, Dni: 11111111A, Género Favorito: Terror
Actores:
nombre: Antonio, apellidos: Banderas, fecha de nacimiento: 1965-01-01, Altura: 175, Idioma principal: Castellano, Teléfono representante: 666666666, Código actor: 1
nombre: Vin, apellidos: Diesel, fecha de nacimiento: 1970-06-10, Altura: 179, Idioma principal: Castellano, Teléfono representante: 666669999, Código actor: 3
*** Información de la película ***
Director:
nombre: Pepe, apellidos: De los muebles, fecha de nacimiento: 1963-09-20, Dni: 22222222B, Género Favorito: Comedia
Actores:
nombre: Antonio, apellidos: Banderas, fecha de nacimiento: 1965-01-01, Altura: 175, Idioma principal: Castellano, Teléfono representante: 666666666, Código actor: 1
nombre: Vin, apellidos: Diesel, fecha de nacimiento: 1970-06-10, Altura: 179, Idioma principal: Castellano, Teléfono representante: 666669999, Código actor: 3
nombre: Jason, apellidos: Statan, fecha de nacimiento: 1969-05-07, Altura: 181, Idioma principal: Inglés, Teléfono representante: 666666888, Código actor: 2
--------------------------------------
false
--------------------------------------
*** Información de la película ***
Director:
nombre: Juanico, apellidos: Colorao, fecha de nacimiento: 1959-12-30, Dni: 33333333C, Género Favorito: Acción
Actores:
nombre: Antonio, apellidos: Banderas, fecha de nacimiento: 1965-01-01, Altura: 175, Idioma principal: Castellano, Teléfono representante: 666666666, Código actor: 1
nombre: Vin, apellidos: Diesel, fecha de nacimiento: 1970-06-10, Altura: 179, Idioma principal: Castellano, Teléfono representante: 666669999, Código actor: 3
--------------------------------------
*** Información de la película ***
Director:
nombre: Álex, apellidos: De la iglesia, fecha de nacimiento: 1956-06-10, Dni: 11111111A, Género Favorito: Terror
Actores:
nombre: Antonio, apellidos: Banderas, fecha de nacimiento: 1965-01-01, Altura: 175, Idioma principal: Castellano, Teléfono representante: 666666666, Código actor: 1
nombre: Vin, apellidos: Diesel, fecha de nacimiento: 1970-06-10, Altura: 179, Idioma principal: Castellano, Teléfono representante: 666669999, Código actor: 3
nombre: Jason, apellidos: Statan, fecha de nacimiento: 1969-05-07, Altura: 181, Idioma principal: Inglés, Teléfono representante: 666666888, Código actor: 2
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.313 s
Finished at: 2023-03-22T20:32:02+01:00
------------------------------------------------------------------------
```