Entrega 6 Programación. Juan Carlos Saldaña Herrero

Ejercicio 1

El resultado del ejercicio en consola es el siguiente:

```
*** To String ***
 Alumno: Nif =65482937, Nombre = Raúl, Dirección = C//Bicicleta, Teléfono = 6547
 Alumno: Nif =98765821, Nombre = Carlos, Dirección = C//manteca, Teléfono = 6548
 Administrativo: Nif =98754632, Nombre = Pedrito, Dirección = C//Mora, Teléfono

*** Trabajar ***
 El profesor Raúl va a impartir su clase
 El alumno Carlos va a estudiar para el curso 1º DAW
 El administrativo Pedrito va a realizar las siguientes tareas: Dormitar

*** LLamar ***
 Carlos llamando a Raúl

*** Poner notas ***
 El profesor Raúl va a corregir los exámenes

*** Hacer examen ***
 El alumno Carlos va a hacer su examen

*** Gestionar matrícula ***
 El administrativo Pedrito va a gestionar una matrícula
```

El toString contiene mucha más información, incluyendo atributos, hashcode…

Para llegar a este resultado, he creado 3 Objetos, profe, alu y admin.

Principal.java

```java
package main;

import tiposPersona.Administrativo;

public class Principal {

    public static void main(String[] args) {
        //Creamos objetos
        Profesor profe = new Profesor("65482937", "Raúl", "C//Bicicleta", "654789321", "Tic e Inglés");
        Alumno alu = new Alumno("98765821","Carlos","C//manteca","654839281", "1º DAW");
        Administrativo admin = new Administrativo("98754632", "Pedrito", "C//Mora", "654789123", "Dormitar");

        /*****Sysos con funciones*****/

        //toString
        System.out.println("*** To String ***  " + "\n" + profe.toString() + "\n" + alu.toString() + "\n" + admin.toString() + "\n");

        //Trabajar y llamar
        System.out.println("*** Trabajar ***  " + "\n" + profe.trabajar() + "\n" + alu.trabajar() + "\n" + admin.trabajar() + "\n");
        System.out.println("*** LLamar ***  " + "\n" + alu.llamar(profe) + "\n");

        //Específicos de cada tipo de persona
        System.out.println("*** Poner notas ***  " + "\n" + profe.ponerNotas() + "\n");
        System.out.println("*** Hacer examen ***  " + "\n" + alu.hacerExamen() + "\n");
        System.out.println("*** Gestionar matrícula ***  " + "\n" + admin.gestionarMatrícula());
    }
}
```

A cada uno de estos les he insertado unos valores mediante los constructores que luego enseñaré. Con los valores dados, y mediante sysos, se usan todas las funciones de otras clases en esta, Principal, donde está el main.

Para esto parto de la clase abstracta Persona, desde la que heredan Administrativo, Alumno y Profesor:

Persona.java

```java
package main;

public abstract class Persona {
    //Atributos
    private String nif;
    private String nombre;
    private String dirección;
    private String teléfono;

    //Constructores
    public Persona (String nif, String nombre, String dirección, String teléfono) {
    }
    public Persona () {

    }
    //Funciones
    public String llamar(Persona p){
        return " " + this.nombre + " llamando a " + p.getNombre();
    }
    public abstract String trabajar();
    public String toString() {
        return "Persona [nif=" + nif + ", nombre=" + nombre + ", dirección=" + dirección +
                ", teléfono=" + teléfono + "]";
    }

    //Get y Set
    public String getNif() {
        return nif;
    }
    public String getNombre() {
        return nombre;
    }
    public String getDirección() {
        return dirección;
    }
    public String getTeléfono() {
        return teléfono;
    }
    public void setNif(String nif) {
        this.nif = nif;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public void setDirección(String dirección) {
        this.dirección = dirección;
    }
    public void setTeléfono(String teléfono) {
        this.teléfono = teléfono;
    }
}

```

Hay unos atributos perteneciente a todas las personas, un constructor vacío y otro con todos los valores, las funciones llamar, trabajar y toString, y por último los Get y Set.

A continuación voy a enseñar las 3 clases mencionadas que heredan de Persona. Las 3 son muy parecidas, cambiando sólo alguna propiedad exclusiva a ellas (como tareas) y una función específica, como gestionarMatrícula en caso del administrativo:

Alumno.java

```java
package tiposPersona;
import main.Persona;

public class Alumno extends Persona{

    //Atributos
    private String curso;

    //Get y Set
    public String getCurso() {
        return curso;
    }
    public void setCurso(String curso) {
        this.curso = curso;
    }
    //Constructor
    public Alumno(String nif, String nombre, String dirección, String teléfono, String curso) {
        super(nif, nombre, dirección, teléfono);
        super.setNif(nif);
        super.setNombre(nombre);
        super.setDirección(dirección);
        super.setTeléfono(teléfono);
        this.curso = curso;
    }
    //Funciones
        @Override
        public String trabajar() {
            return " El alumno " + this.getNombre() + " va a estudiar para el curso " + getCurso();
        }
        public String hacerExamen() {
            return " El alumno " + this.getNombre() + " va a hacer su examen";
        }
        @Override
        public String toString() {
            return " Alumno: " + "Nif =" + getNif() + ", Nombre = " + getNombre() + ", Dirección = "
                    + getDirección() + ", Teléfono = " + getTeléfono() + ", Curso = " + getCurso()+
                    ", Class = " + getClass() + ", hashCode = " + hashCode() + ", toString = " + super.toString();
        }
}
```

Profesor.java

```java
package tiposPersona;
import main.Persona;

public class Profesor extends Persona{

    //Atributos
    private String competencias;

    //Funciones
    public String getCompetencias() {
        return competencias;
    }
    public void setCompetencias(String tareas) {
        this.competencias = tareas;
    }

    //Constructor
    public Profesor(String nif, String nombre, String dirección, String teléfono, String competencias) {
        super(nif, nombre, dirección, teléfono);
        super.setNif(nif);
        super.setNombre(nombre);
        super.setDirección(dirección);
        super.setTeléfono(teléfono);
    }

    //Funciones
        @Override
        public String trabajar() {
            return " El profesor " + this.getNombre() + " va a impartir su clase";
        }
        public String ponerNotas() {
            return " El profesor " + this.getNombre() + " va a corregir los exámenes";
        }
        @Override
        public String toString() {
            return " Alumno: " + "Nif =" + getNif() + ", Nombre = " + getNombre() + ", Dirección = "
                    + getDirección() + ", Teléfono = " + getTeléfono() + ", Competencias = " + getCompetencias()+
                    ", Class = " + getClass() + ", hashCode = " + hashCode() + ", toString = " + super.toString();
        }

}
```

Administrativo.java

```java
package tiposPersona;

import main.Persona;

public class Administrativo extends Persona{
    //Atributos
    private String tareas;

    //Get y Set
    public String getTareas() {
        return tareas;
    }
    public void setTareas(String tareas) {
        this.tareas = tareas;
    }

    //Constructor
    public Administrativo(String nif, String nombre, String dirección, String teléfono, String tareas) {
        super(nif, nombre, dirección, teléfono);
        super.setNif(nif);
        super.setNombre(nombre);
        super.setDirección(dirección);
        super.setTeléfono(teléfono);
        this.tareas = tareas;
    }

    //Funciones
        @Override
        public String trabajar() {
            return " El administrativo " + this.getNombre() + " va a realizar las siguientes tareas: " + getTareas();
        }
        public String gestionarMatrícula() {
            return " El administrativo " + this.getNombre() + " va a gestionar una matrícula";
        }
        @Override
        public String toString() {
            return " Administrativo: " + "Nif =" + getNif() + ", Nombre = " + getNombre() + ", Dirección = "
                    + getDirección() + ", Teléfono = " + getTeléfono() + ", Curso = " + getTareas()+
                    ", Class = " + getClass() + ", hashCode = " + hashCode() + ", toString = " + super.toString();
        }
    }
```

Como he explicado, las 3 tienen un funcionamiento muy similar:

Un atributo exclusivo a ellas.

Get y set

Constructor con los atributos de la clase madre, y los suyos propios.

Funciones propias y otras con @Override debido a que están sobrescritas de Persona. En el caso de toString, podemos generarlo de manera automática, pero los he modificado para que sean más sencillos de leer y no tan extensos.

.

.

.

.

.

(Continua con ejercicio 2)

Ejercicio 2

Comienzo con la clase compuesta **Libro**, que cuenta con los atributos título y género, además de autor. Escribo los Get y Set correspondientes para acceder a ellos ya que son privados, y pongo el método toString para ver los atributos. En toString, pongo toString el autor, la clase derivada, para ver sus atributos también. Por último utilizo los constructores necesarios que serán los que vengan por defecto; con todos los atributos y sin ninguno.

Libro.java

```java
package com.itt.librería;

public class Libro{
    private String título;
    private String género;
    private Autor autor;

    //Get y Set

    public String getTítulo() {
        return título;
    }
    public void setTítulo(String título) {
        this.título = título;
    }
    public String getGénero() {
        return género;
    }
    public void setGénero(String género) {
        this.género = género;
    }
    public Autor getAutor() {
        return autor;
    }
    public void setAutor(Autor autor) {
        this.autor = autor;
    }

    //Métodos
    public String toString() {
        return "Título = " + título + "\n" + "Género = " + género + "\n" + "Autor = " + autor.toString();
    }

    //Constructores
    public Libro(String título, String género, Autor autor) {
        this.título = título;
        this.género = género;
        this.autor = autor;
    }
    public Libro() {
    }
}

```

La Clase siguiente es **Autor**, que siendo componente de libro, cuenta además con los atributos nombre y biografía. Tiene los constructores necesarios y además lo usaré para almacenar las biografías para no tener un main demasiado extenso. Con los Get y Set tenemos la clase lista

Autor.java

```java
package com.itt.librería;

public class Autor extends Libro{
    //Atributos
    private String nombre;
    private String biografía;

    //Métodos
    @Override
    public String toString() {
        return "Nombre => " + nombre + "\n" + "      = Biografía => " + getBiografía();
    }

    //Constructores
    public Autor(String nombre, String biografía) {
        this.nombre = nombre;
        this.biografía = biografía;
    }
    public Autor() {
    }

    public Autor(String género, String título, Autor autor, String nombre, String biografía) {
        super(género, título, autor);
        this.nombre = nombre;
        this.biografía = biografía;
    }


    //Biografías
    private String biografiaJosh = "Joshua J. Bloch (born August 28, 1961) is an American software engineer"
            + "\n" + "  and a technology author, formerly employed at Sun Microsystems and Google. He led the design "
            + "\n" + "  and implementation of numerous Java platform features, including the Java Collections Framework,"
            + "\n" + "        the java.math package, and the assert mechanism.[1] He is the author of the programming guide "
            + "\n" + "  Effective Java (2001), which won the 2001 Jolt Award,[2] and is a co-author of two other Java books,"
            + "\n" + "  Java Puzzlers (2005) and Java Concurrency In Practice (2006)." + "\n";

    //Get y Set
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String getBiografíaJosh() {
        return biografiaJosh;
    }
    public void setBiografíaJosh(String biografiaJosh) {
        this.biografía = biografiaJosh;
    }
    public String getBiografía() {
        return biografía;
    }
    public void setBiografía(String biografía) {
        this.biografía = biografía;
    }
}

```

Por último tenemos la clase **Principal** en el paquete por defecto, con el main, en la que creamos un objeto libro y otro autor. Creamos libro **con** autor, como se verá en el constructor. Hago un toString de la clase Libro, y luego modifico la biografía.

"En el ejercicio se pide modificar biografía desde el objeto libro, sin embargo usar modificar un atributo de la clase derivada desde la clase madre entiendo que va en contra de la POO. Al crear libro, ya modifico la biografía del autor, pero no una vez creado. Si en la corrección se me aclarase lo agradecería"

Principal.java

```java
import com.itt.librería.Autor;

public class Principal {

    public static void main(String[] args) {

        //Creamos objeto autor y libro
        Autor josh = new Autor("Joshua Bloch", "biografiaJosh.pdf");
        Libro java = new Libro("Effective Java","Informática", josh);

        //String del libro
        System.out.println("\n" + " ***** toString del libro *****" + "\n" + "\n" + java.toString());

        //Modificar biografía mediante Libro (no se puede modificar atributo de clase componente mediante la compuesta)
        josh.setBiografíaJosh(josh.getBiografíaJosh());

        //String del autor mediante referencia al objecto Libro y Autor
        System.out.println("\n" + " ***** toString del autor mediante referencia Libro *****" + "\n" + "\n" + java.toString());
        System.out.println(" ***** toString del autor mediante referencia Autor *****" + "\n" + "\n" + josh.toString());

        //Cambiar propiedad género
        java.setGénero("Sigue siendo informática, no va a cambiar");
        System.out.println("\n" + " ***** toString del libro modificado *****" + "\n" + "\n" + java.toString());
    }
}
```

Modificada la biografía, vuelvo a ejecutar toString desde el objeto libro, y desde el objeto autor, cuyo toString contiene el de libro.

Para terminar, cambio el género del objeto libro llamado java, y vuelvo a imprimir para demostrar resultado. La consola queda como sigue:

.

.

.

.

.

.

.

.

.

.

```
 ***** toString del libro *****

Título = Effective Java
Género = Informática
Autor = Nombre => Joshua Bloch
      = Biografía => biografiaJosh.pdf

 ***** toString del autor mediante referencia Libro *****

Título = Effective Java
Género = Informática
Autor = Nombre => Joshua Bloch
      = Biografía => Joshua J. Bloch (born August 28, 1961) is an American software engineer
        and a technology author, formerly employed at Sun Microsystems and Google. He led the design
        and implementation of numerous Java platform features, including the Java Collections Framework,
        the java.math package, and the assert mechanism.[1] He is the author of the programming guide
        Effective Java (2001), which won the 2001 Jolt Award,[2] and is a co-author of two other Java books,
        Java Puzzlers (2005) and Java Concurrency In Practice (2006).

 ***** toString del autor mediante referencia Autor *****

Nombre => Joshua Bloch
      = Biografía => Joshua J. Bloch (born August 28, 1961) is an American software engineer
        and a technology author, formerly employed at Sun Microsystems and Google. He led the design
        and implementation of numerous Java platform features, including the Java Collections Framework,
        the java.math package, and the assert mechanism.[1] He is the author of the programming guide
        Effective Java (2001), which won the 2001 Jolt Award,[2] and is a co-author of two other Java books,
        Java Puzzlers (2005) and Java Concurrency In Practice (2006).


 ***** toString del libro modificado *****

Título = Effective Java
Género = Sigue siendo informática, no va a cambiar
Autor = Nombre => Joshua Bloch
      = Biografía => Joshua J. Bloch (born August 28, 1961) is an American software engineer
        and a technology author, formerly employed at Sun Microsystems and Google. He led the design
        and implementation of numerous Java platform features, including the Java Collections Framework,
        the java.math package, and the assert mechanism.[1] He is the author of the programming guide
        Effective Java (2001), which won the 2001 Jolt Award,[2] and is a co-author of two other Java books,
        Java Puzzlers (2005) and Java Concurrency In Practice (2006).
```