

Introducción a Programación orientada a objetos

¿Qué es la Programación orientada a objetos?

JavaScript La programación Orientada a Objetos (abreviatura POO en español, y OOP en inglés) es una metodología que basa la estructura de los programas en torno a los objetos. Los lenguajes que implementan POO ofrecen medios y herramientas para describir los objetos manipulados por un programa. Más que describir cada objeto individualmente, estos lenguajes proveen una construcción (Clase) que describe a un conjunto de objetos. POO no es único de un lenguaje, pero hay lenguajes en los cuales resulta es más fácil trabajar con esta metodología.

Conceptos básicos en la POO

Clase

Una clase es una agrupación de objetos que comparten una estructura común y un comportamiento común. Todo objeto pertenece a alguna clase.

Todos los objetos que pertenezcan a la misma clase tienen igual formato y comportamiento la diferencia está en que adquieren valores únicos.

Las clases se relacionan entre sí por medio de un sistema de jerarquía, esto es posible gracias a un sistema de herencia donde las clases padre (superclase) heredan métodos y atributos a la clases hija (subclase).

Ejemplo:

Clase Padre: Clase bicicleta.

Clase hija: clase bicicleta de tándem, clase bicicleta de montaña.

Ejemplo código:

```
public class Bicicleta{
    String marca;
    String tipo;
    String color;

    void comenzar_desplazamiento() {
    }

    void aumentar_velocidad() {
    }

    void detener_desplazamiento() {
    }
}
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

Objeto

Es un conjunto de variables (o datos) y métodos (o funciones) las cuales están relacionadas entre sí. Un objeto por lo tanto es la representación en un programa de un concepto y contiene la información necesaria para abstraerlo (Un auto, una casa, una persona, serian algunos ejemplos).

Cabe recalcar que cada objeto tiene sus atributos propios y cuyo comportamiento está determinado por las acciones o funciones que pueden modificarlo.

Ejemplo:

Tomemos como ejemplo la clase león estos serían algunos de sus datos y algunos de sus métodos.

Datos	Métodos
Color	Desplazarse
Tamaño	Masticar
Peso	Digerir
Colmillos	Secretar hormonas
Cuadrúpedo	Parpadear

Una vez ya entendidos los conceptos de objetos y clases aprenderemos como crearlos. En java está reservada la palabra **new** la cual se utiliza para la creación de nuevos objetos.

Existen tres pasos al crear un objeto de una clase:

Declarar: Debemos declarar una variable con su nombre y con el tipo de objeto que va a contener.

Instanciar: La palabra clave new se utiliza para crear el objeto.

Inicialización: La palabra clave new va seguida de una llamada a un constructor. Esta llamada inicializa el nuevo objeto.

Ejemplo 2:

```
Bicicleta mibicibleta = new Bicicleta( "Bulls" );
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

Ejemplo código:

En el ejemplo que se presenta a continuación el archivo llamado **Bicicleta.java** va dentro de un único package llamado **herencia**.

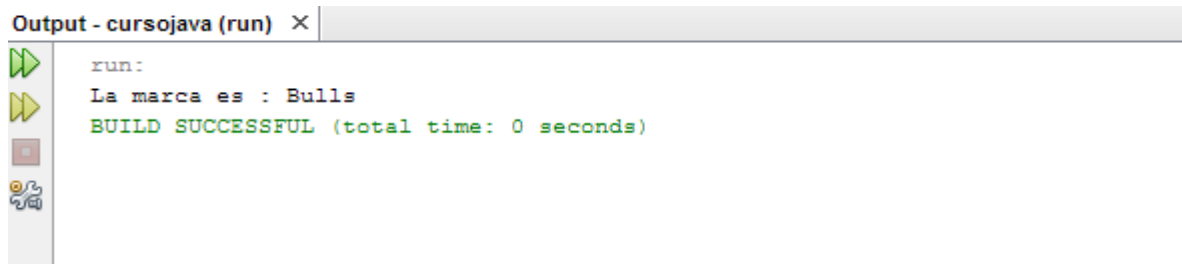
Bicicleta.java

```
//Al crear un nuevo archivo se incluye el package en el cual está incluido
package encapsulamiento;

//creamos una clase pública llamada Bicicleta
public class Bicicleta{

    public Bicicleta (String marca){
        //Cuando ejecutamos el método principal(main()), se ejecuta
        //esto por el objeto creado dentro de main, tal que
        //marca=Bulls
        System.out.println("La marca es: " + marca );
    }
    /*Se define un método público y estático llamado main()
    *El tipo de dato void indica al método que no devuelva parámetros
    *String[] args, recibe cualquier argumento de tipo string que sea
    * enviado al ejecutar el programa.
    */
    public static void main(String []args){
        // Creamos la variable mibicicleta a partir de Bicicleta y le
        //daremos el valor "Bulls"
        Bicicleta mibicicleta = new Bicicleta( "Bulls" );
    }
}
```

Si compilamos el código anterior obtendremos por resultado:



```
Output - cursojava (run) ×
run:
La marca es : Bulls
BUILD SUCCESSFUL (total time: 0 seconds)
```

Encapsulamiento

Se denomina encapsulamiento al ocultamiento de los "datos" (estado) de un objeto, de tal manera que sólo se puedan cambiar dichos estados mediante las operaciones (métodos) definidas en dicho objeto. La idea principal detrás de este concepto es de esconder los detalles y mostrar solo lo relevante. Permite el ocultamiento de la información separando el aspecto correspondiente a la especificación de la implementación; de esta forma, distingue el "*qué hacer*" del "*cómo hacer*". La especificación es visible por el usuario, mientras la implementación se oculta.

Ejemplo:

"Un teclado del pc es la interfaz pública y el mecanismo interno para que al pulsar una tecla determinada imprima un caracter determinado, es lo que llamaremos interfaz privada."

El encapsulamiento en POO se representa en cada clase u objeto, definiendo sus atributos y métodos según los siguientes modos de acceso:

public	Atributos o Métodos que son accesibles fuera de la clase. Pueden ser llamados por cualquier clase, aun si no está relacionada con ella.
private	Atributos o Métodos que solo son accesibles dentro de la implementación de la clase.
protected	Atributos o Métodos que son accesibles para la propia clase y sus clases hijas (subclases).

Ejemplo código:

En el ejemplo que se presenta a continuacion el archivo llamdo **Bicicleta.java** va dentro de un unico package llamado **herencia**.

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

```
package encapsulamiento;

public class Bicicleta{

    private int tipo;

    public void settipo(int t) {
        tipo=t;
    }
    public int gettipo() {
        return tipo;
    }
}

class accesoindirecto{

    public static void main(String [] args){
        Bicicleta mc = new Bicicleta ();
        mc.settipo(20);
        System.out.println("El aro de la bicicleta es: "+mc.gettipo());
    }
}
```

Para explicar de mejor manera el ejemplo anterior diremos que en el método `setTipo()` no existen validaciones para prevenir que un valor no válido sea asignado a la variable, sin embargo, el proveer de un método de este tipo desde el diseño inicial de la aplicación nos permite posteriormente modificar el comportamiento de la misma sin afectar los métodos utilizados, tal vez en un futuro se desee que dicha variable solamente pueda tener uno entre un rango de valores y se podrán aplicar posteriormente los cambios sin que haya repercusiones negativas. En cuanto al método `gettipo()` se utiliza para obtener un dato (generalmente no recibe parametro alguno).

Para explicar mejor los métodos `Setters(set)` & `Getters(get)`, expondremos otro código:

En el ejemplo que se presenta a continuación los archivos de [Estudiante.java](#) y [Principal.java](#) van dentro de un único package llamado `setandget`.

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

Estudiante.java

```
package setandget;
public class Estudiante{
    private String nombre;
    private String apellido;
    private String correo;

    public String getNombre(){
        return nombre;
    }
    public void setNombre(String nombre){
        this.nombre = nombre;
    }

    public String getApellido(){
        return apellido;
    }
    public void setApellido(String apellido){
        this.apellido = apellido;
    }
    public String getCorreo(){
        return correo;
    }
    public void setCorreo(String correo){
        this.correo = correo;
    }
}
```

Principal.java

```
package setandget;
public Class Principal
{
    public static void main(String args[])
    {
        Estudiante estudiante1 = new Estudiante();

        estudiante1.setNombre( "Patricio" );
        estudiante1.setApellido( "Gomez" );
        estudiante1.setCorreo( "p.gomez999@ufromail.cl" );

        System.out.println(estudiante1.getNombre());
        System.out.println(estudiante1.getApellido());
        System.out.println(estudiante1.getCorreo());
    }
}
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

Modularidad

Esto permite que se tengan independencia entre las diferentes partes de un sistema. La modularidad consiste en dividir un programa en distintas partes o módulos, que pueden ser compilados separadamente, pero tienen conexiones con otros módulos. En un mismo modulo se suele colocar clases y objetos que guarden una estrecha relación. El sentido de modularidad está muy relacionado con el ocultamiento de información.

Herencia

La herencia es uno de los mecanismos fundamentales de la POO, puesto que se puede construir una clase a partir de otra clase. Una de sus funciones más importantes es proveer el polimorfismo. La forma en la que se ordenan es en base a una jerarquía de clases que tiene estructura de árbol.

Como y cuando se deriva la herencia de una clase a otra es puramente decisión del programador.

```
public class SeleccionFutbol{...}  
public class Futbolista extends SeleccionFutbol{...}
```

Clase this: Esto lo que hace es invocar a otro constructor que este en la misma clase y que soporte el conjunto de parámetros que le pasamos.

Constructor: Su función es inicializar el objeto y sirve para asegurarnos que los objetos siempre contengan valores válidos.

En el ejemplo que se presenta a continuacion los archivos de [moto.java](#), [vendedor.java](#) y [EjemploHerencia.java](#) van dentro de un unico package llamado [herencia](#).

moto.java

```
package herencia;  
public class moto {  
  
    private String marca;  
    private String tipo;  
    private int patente;
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

```
public moto (String marca, String tipo, int patente) {
    this.marca = marca;
    this.tipo = tipo;
    this.patente = patente;
}

public String getMarca() {
    return marca;
}

public String getTipo() {
    return tipo;
}

public int getPatente () {
    return patente;
}
}
```

vendedor.java

```
package herencia;
public class vendedor extends moto {

    private String Idvendedor;

    public vendedor (String marca, String tipo, int patente) {
        super(marca, tipo, patente);
        Idvendedor = "Unknown";
    }

    public void setIdvendedor (String Idvendedor) {
        this.Idvendedor = Idvendedor;
    }

    public String getIdvendedor () {
        return Idvendedor;
    }

    public void mostrar_Marca_tipo_patente() {
        System.out.println ("Marca de la moto: " + getMarca() + "
\nTipo de Moto: " + getTipo() + "\nVendedor de la moto: " +
getIdvendedor());
    }
}
```


UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

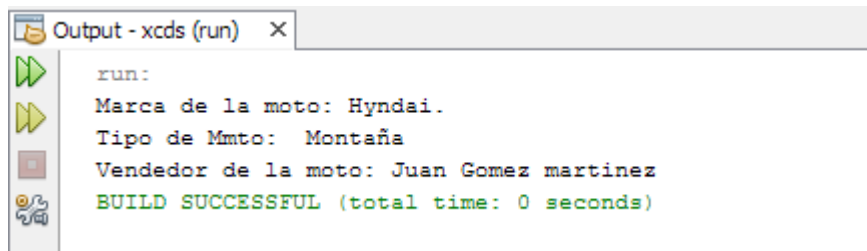
Depto. Ciencias de Computación e Informática

EjemploHerencia.java

```
package herencia;
public class EjemploHerencia {

    public static void main (String [ ] Args) {
        vendedor vendedor1 = new vendedor ("Hyndai","Ciudad",5785);
        vendedor1.setIdvendedor("Juan Zigela Rodriguez");
        vendedor1.mostrar_Marca_tipo_patente();
    }
}
```

Y al ejecutarlo nos arroja:



Polimorfismo

Por polimorfismo se entiende aquella cualidad que poseen los objetos para responder de distinto modo ante el mismo mensaje. Pongamos el ejemplo que tenemos las clases *auto*, *bicicleta* y *monopatín*, si le enviamos el mensaje *desplázate*, cada uno de ellos sabrá cómo hacerlo y realizara su comportamiento a su modo.

Para ser un poco más precisos en el mensaje no se envía a las clases en sí, sino a los objetos instanciados de las clases.

El polimorfismo nos ayuda a simplificar el trabajo ya que gracias a este, los nombres de objetos a recordar disminuyen considerablemente.

Clases de otros paquetes

Para obtener Clases de otro paquete se debe utilizar "import nombrePaquete.*;" esto hará que se importe todas las clases dentro del paquete definido, pero si se desea importar una clase en específico se utiliza "import nombrePaquete.nombreClase;" .

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

Código Ejemplo:

En el ejemplo que se presenta a continuación los archivos de **Frances.java**, **Estadounidense.java** y **Persona.java** van dentro de un único package llamado **humanos**.

Por otro lado **Polimorfismo.java** va en un package diferente llamado **polimorfismo**.

Polimorfismo.java

//Como tenemos diferentes archivos en diferentes package, se importan;
para esto se utiliza **import nombre_package.nombre_archivo**;

```
package polimorfismo;
import humanos.Persona;
import humanos.Estadounidense;
import humanos.Frances;

public class Polimorfismo {
    public static void main(String[] args) {
        Persona p;
        a = new Estadounidense();
        test(p);
        a = new Frances();
        test(p);

        Persona humanos[] = new Persona[2];
        humanos[0] = new Estadounidense();
        humanos[1] = new Frances();
    }
    public static void test(Persona persona) {
        persona.idioma();
    }
}
```

Para explicar el código anterior, se invoca un objeto(p), el cual es una instancia de la clase persona, luego le digo que p, hace referencia a **Estadounidense**, luego se llama a un método llamado test recibe “una” Persona, pero no sabe cual es, puede ser un **Estadounidense** o un **Frances**, se llama al método idioma().

El siguiente trozo de código, **Persona humanos[] = new Persona[2];**
inicializa el array y luego **humanos[0] = new Estadounidense();**
agregar en cada posición el **humanos[1] = new Frances();**
objeto:

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

Persona.java

```
package humanos;

public class Persona {

    String tipo;
    public void idioma() {
        System.out.println("Que idioma hablan?");
    }

}
```

Estadounidense.java

```
package humanos;

public class Estadounidense extends Persona {

    @Override
    public void idioma() {
        System.out.println("ingles");
    }

}
```

Frances.java

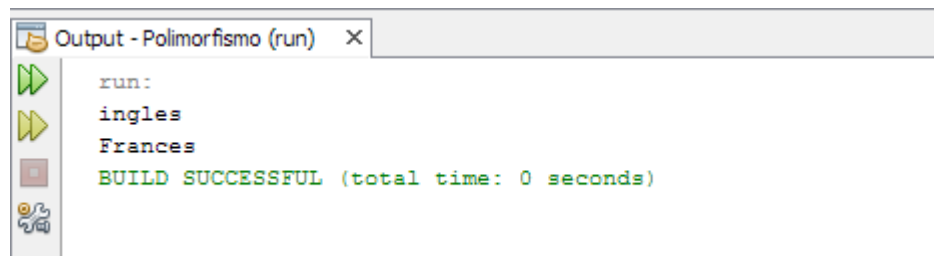
```
package humanos;

public class Frances extends Persona{

    @Override
    public void idioma() {
        System.out.println("Frances");
    }

}
```

Y al ejecutarlo nos arroja:



```
Output - Polimorfismo (run) X
run:
ingles
Frances
BUILD SUCCESSFUL (total time: 0 seconds)
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

Codigo ejemplo:

En el ejemplo que se presenta a continuacion los archivos de [Persona.java](#) y [Pruebapersona.java](#) van dentro de un unico package llamado [persona](#).

Persona.java

```
package persona;

public class Persona {

    private String nombre;
    private String apellidos;
    private String dni;
    private Integer edad;
    private String sexo;

    public String codigoIdentificador;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellidos() {
        return apellidos;
    }

    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }

    public String getDni() {
        return dni;
    }

    public void setDni(String dni) {
        this.dni = dni;
    }

    public Integer getEdad() {
        return edad;
    }
}
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

```
public void setEdad(Integer edad) {
    this.edad = edad;
}

public String getSexo() {
    return sexo;
}

public void setSexo(String sexo) {
    this.sexo = sexo;
}

public String toString() {
    String resultado = "DATOS DE LA PERSONA:" +
        "\nCodigo Identificador: " + this.codigoIdentificador+
        "\nNombres: " + this.getNombre()+
        "\nApellidos: " + this.getApellidos()+
        "\nRUT: " + this.getDni() +
        "\nSexo: " + this.getSexo() +
        "\nEdad: " + this.getEdad();

    return resultado;
}
}
```

PruebaPersona.java

```
package persona;

public class PruebaPersona {

    public static void main(String [] args) {
        Persona humano = new Persona();

        humano.codigoIdentificador = "112";

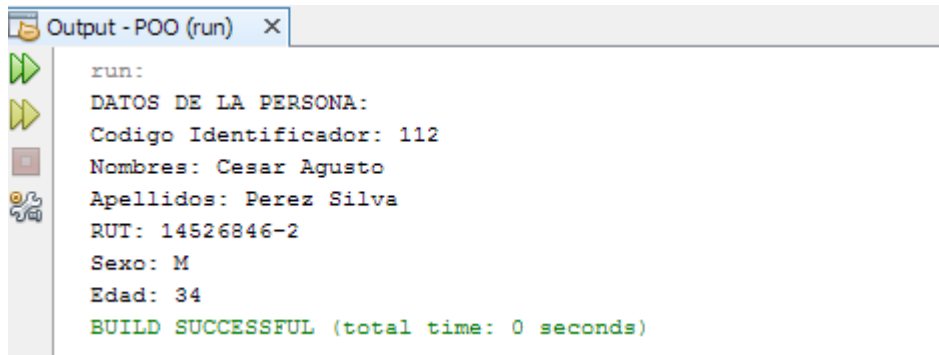
        humano.setNombre("Cesar Augusto");
        humano.setApellidos("Perez Silva" );
        humano.setEdad(34 );
        humano.setSexo("M" );
        humano.setDni("14526846-2" );
        System.out.println(humano.toString());
    }
}
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

Y al ejecutarlo nos arroja:



```
Output - POO (run) X
run:
DATOS DE LA PERSONA:
Codigo Identificador: 112
Nombres: Cesar Augusto
Apellidos: Perez Silva
RUT: 14526846-2
Sexo: M
Edad: 34
BUILD SUCCESSFUL (total time: 0 seconds)
```

Codigo ejemplo Cambiando propiedades de un objeto:

En el ejemplo que se presenta a continuacion los archivos de **Principal.java** y **Objeto.java** van dentro de un unico package llamado **Principal**.

Principal.java

```
package Principal;
public class Principal {

    public static void main(String[] args) {

        Objeto n_objeto = new Objeto();
        n_objeto.CambiarTamanio(50);
        n_objeto.pantalla();
    }

}
```

Objeto.java

```
package Principal;
public class Objeto {

    int ancho;
    int alto;
    int area;
    public Objeto(){
        this.ancho=50;
        this.alto=50;
        this.area=50;
    }

}
```

UNIVERSIDAD DE LA FRONTERA

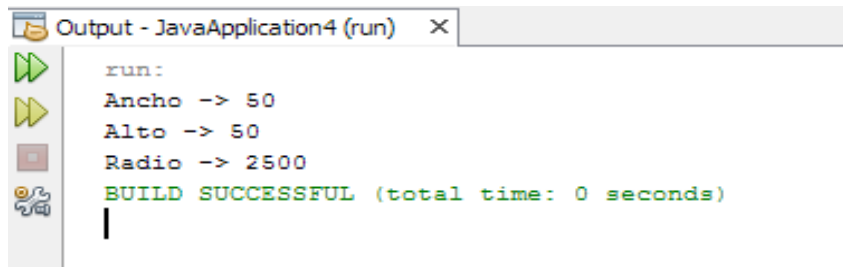
Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

```
public void CambiarTamano(int nuevo_tamano){
    this.ancho=nuevo_tamano;
    this.alto=nuevo_tamano;
    this.area=(int) Math.pow(nuevo_tamano,2);
}
public void pantalla(){
    System.out.println( "Ancho -> " +this.ancho);
    System.out.println( "Alto -> " +this.alto);
    System.out.println( "Radio -> " +this.area);
}
}
```

Este es un ejemplo de código simple en el cual se cambian los parámetros (propiedades) de un objeto (el cual es hijo de otro), en este caso en específico para calcular el área se utiliza “(int) Math.pow(nuevo_tamano,2)”, donde nuevo_tamano es la base y 2 el exponente, en cuanto a “n_objeto.CambiarTamano(50);” esto lo que hace es que n_objeto que tiene unos valores establecidos, se ejecute el código dentro de CambiarTamano, asignándole el valor “(50)” a nuevo_tamano, luego se imprime por pantalla.

Y al ejecutarlo nos arroja:



```
Output - JavaApplication4 (run) X
run:
Ancho -> 50
Alto -> 50
Radio -> 2500
BUILD SUCCESSFUL (total time: 0 seconds)
```

Código de ejemplo para Ejercicio de cine

Un cliente estadounidense propietario del cine más grande del planeta pide a un equipo de estudiantes de la UFRO que le realice un programa que maneje una simulación sobre una sala de su cine para ver cómo funcionaría. Para que esta simulación funcione correctamente el equipo de estudiantes comprende que tiene que generar datos, como la película la cual será proyectada (con una cierta duración, además de con un director, y una clasificación de edad). Además de generar la película se tienen que generar espectadores para esta (con un nombre, una edad, y con una cantidad de dinero)

Los únicos datos que se tendrán que introducir son la cantidad de filas y columnas de asientos, el precio de la película, y una cantidad de espectadores que se crearán para la simulación.

Código:

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

Pelicula.java

```
package simulacion;
```

```
public class Pelicula {
```

```
    private String titulo;
```

```
    private int duracion;
```

```
    private int edadMinima;
```

```
    private String director;
```

```
    public Pelicula(String titulo, int duracion, int edadMinima,  
String director) {
```

```
        this.titulo = titulo;
```

```
        this.duracion = duracion;
```

```
        this.edadMinima = edadMinima;
```

```
        this.director = director;
```

```
    }
```

```
    public String getTitulo() {
```

```
        return titulo;
```

```
    }
```

```
    public void setTitulo(String titulo) {
```

```
        this.titulo = titulo;
```

```
    }
```

```
    public int getDuracion() {
```

```
        return duracion;
```

```
    }
```

```
    public void setDuracion(int duracion) {
```

```
        this.duracion = duracion;
```

```
    }
```

```
    public int getEdadMinima() {
```

```
        return edadMinima;
```

```
    }
```

```
    public void setEdadMinima(int edadMinima) {
```

```
        this.edadMinima = edadMinima;
```

```
    }
```

```
    public String getDirector() {
```

```
        return director;
```

```
    }
```


UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

```
    public void setDirector(String director) {
        this.director = director;
    }

    @Override
    public String toString() {
        return "'" + titulo + "' del director " + director + ", con
        una duracion de " + duracion + " minutos y la edad minima es
        de " + edadMinima + " años";
    }
}
```

Cine.java

```
package simulacion;
```

```
public class Cine {
    private Asiento asientos[][];
    private double precio;
    private Pelicula pelicula;

    public Cine(int filas, int columnas, double precio, Pelicula
    pelicula) {

        asientos = new Asiento[filas][columnas];
        this.precio = precio;
        this.pelicula = pelicula;
        rellenaButacas();
    }

    public Asiento[][] getAsientos() {
        return asientos;
    }
    public void setAsientos(Asiento[][] asientos) {
        this.asientos = asientos;
    }
    public double getPrecio() {
        return precio;
    }
    public void setPrecio(double precio) {
        this.precio = precio;
    }
    public Pelicula getPelicula() {
        return pelicula;
    }
    public void setPelicula(Pelicula pelicula) {
        this.pelicula = pelicula;
    }
}
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

```
//Rellena nuestros asientos, dandoles una fila y una letra
private void rellenaButacas() {
    int fila = asientos.length;
    for (int i = 0; i < asientos.length; i++) {
        for (int j = 0; j < asientos[0].length; j++) {
            asientos[i][j] = new Asiento((char) ('A' + j),
            fila);
        }
        fila--; //Se decrementa la fila para actualizar la fila
    }
}

//Indicamos si hay sitio en el cine, cuando vemos una vacia salimos
de la función
public boolean haySitio() {
    for (int i = 0; i < asientos.length; i++) {
        for (int j = 0; j < asientos[0].length; j++) {
            if (!asientos[i][j].ocupado()) {
                return true;
            }
        }
    }
    return false;
}

//Indico si en una posicion concreta esta ocupada(fila,letra).
public boolean haySitioButaca(int fila, char letra) {
    return getAsiento(fila, letra).ocupado();
}

//Indicamos si el espectador cumple lo necesario para entrar:
dinero y edad.El tema de si hay sitio, se controla en el main
public boolean sePuedeSentar(Espectador e) {
    return e.tieneDinero(precio) &&
    e.tieneEdad(pelicula.getEdadMinima());
}

//Siento al espectador en un asiento
public void sentar(int fila, char letra, Espectador e) {
    getAsiento(fila, letra).setEspectador(e);
}

//Devuelvo un asiento concreto por su fila y letra
public Asiento getAsiento(int fila, char letra) {
    return asientos[asientos.length - fila - 1][letra - 'A'];
}

//Numero de filas de nuestro cine
public int getFilas() {
    return asientos.length;
}
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

```
//Numero de columnas de nuestro cine
public int getColumnas() {
    return asientos[0].length;
}
//Mostramos la información de nuestro cine (Tambien se puede hacer
en un toString pero hay que devolver un String)
public void mostrar() {

    System.out.println("Información cine");
    System.out.println("Película reproducida: " + pelicula);
    System.out.println("Precio entrada: " + precio);
    System.out.println("");
    for (int i = 0; i < asientos.length; i++) {
        for (int j = 0; j < asientos[0].length; j++) {
            System.out.println(asientos[i][j]);
        }
        System.out.println("");
    }
}
```

Asiento.java

```
package simulacion;

public class Asiento {

    private char letra;
    private int fila;
    private Espectador espectador; // informacion del espectador que
esta sentado, null si es vacio

    public Asiento(char letra, int fila) {
        this.letra = letra;
        this.fila = fila;
        this.espectador = null; //al iniciar el asiento, no habrá
nadie sentado
    }

    public char getLetra() {
        return letra;
    }
    public void setLetra(char letra) {
        this.letra = letra;
    }
    public int getFila() {
        return fila;
    }
}
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

```
    public void setFila(int fila) {
        this.fila = fila;
    }
    public Espectador getEspectador() {
        return espectador;
    }
    public void setEspectador(Espectador espectador) {
        this.espectador = espectador;
    }
    //Indica si el asiento esta ocupado
    public boolean ocupado() {
        return espectador != null;
    }
    @Override
    public String toString() {
        if (ocupado()) {
            return "Asiento: " + fila + letra + " y " + espectador;
        }
        return "Asiento: " + fila + letra + " y este asiento está
        vacío ";
    }
}
```

Espectador.java

```
package simulacion;
```

```
public class Espectador {

    private String nombre;
    private int edad;
    private double dinero;

    public Espectador(String nombre, int edad, double dinero) {
        this.nombre = nombre;
        this.edad = edad;
        this.dinero = dinero;
    }
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public int getEdad() {
        return edad;
    }
    public void setEdad(int edad) {
        this.edad = edad;
    }
}
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

```
    public double getDinero() {
        return dinero;
    }
    public void setDinero(double dinero) {
        this.dinero = dinero;
    }

    //Pagamos la entrada del cine
    public void pagar(double precio) {
        dinero -= precio;
    }
    //Indicamos si el espectador tiene edad para ver la pelicula (en el
    video estaba en la clase pelicula tiene mas sentido que sea un
    metodo del espectador)
    public boolean tieneEdad(int edadMinima) {
        return edad >= edadMinima;
    }
    //Indicamos si el espectador tiene dinero (en el video estaba en la
    clase cine tiene mas sentido que sea un metodo del espectador)
    public boolean tieneDinero(double precioEntrada) {
        return dinero >= precioEntrada;
    }

    @Override
    public String toString() {
        return "el nombre del espectador es " + nombre + " de " +
            edad + " años y con " + dinero + " dolares en su bolsillo";
    }
}
```

Metodos.java

```
package simulacion;

//Clase Metodos, contiene funciones útiles para nuestro programa
public class Metodos {

    public static final String nombres[] = {"Jose", "Maria", "Diego",
        "Eufasia"};

    public static int generaNumeroEnteroAleatorio(int minimo, int
        maximo) {
        int num = (int) (Math.random() * (minimo - (maximo + 1)) +
            (maximo + 1));
        return num;
    }
}
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

Principal.java

```
package simulacion;

import java.util.Scanner;

public class Principal{
    public static void main(String[] args) {

        //Creo la pelicula
        Pelicula pelicula = new Pelicula("Mi vida", 90, 16, "DDR");

        // Pido datos (esto no se mostro en el video por falta de
        tiempo)
        // No valida nada al respecto de tamaños (siguiente version)
        Scanner sn = new Scanner(System.in);

        System.out.println("Introduce el numero de filas");
        int filas=sn.nextInt();

        System.out.println("Introduce el numero de columnas");
        int columnas=sn.nextInt();

        System.out.println("Introduce el precio de la entrada de
        cine");
        double precio=sn.nextDouble();

        //Creo el cine, necesito la pelicula para ello
        Cine cine = new Cine(filas, columnas, precio, pelicula);

        //Numero de espectadores que seran creados
        System.out.println("Introduce el numero de espectadores a
        crear");
        int numEspectadores = sn.nextInt();

        //Variables y objetos usados
        Espectador e;
        int fila;
        char letra;

        System.out.println("Espectadores generados: ");
        //Termino cuando no queden espectadores o no haya mas sitio
        en el cine
        for (int i = 0; i< numEspectadores && cine.haySitio(); i++) {

            //Generamos un espectador
            e = new Espectador(
                Metodos.nombres[Metodos.generaNumeroEnteroAleatorio(0,
                Metodos.nombres.length - 1)], //Nombre al azar
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

```
        Metodos.generaNumeroEnteroAleatorio(10, 30),
        //Generamos una edad entre 10 y 30
        Metodos.generaNumeroEnteroAleatorio(1, 10));
        //Generamos el dinero entre 1 y 10 euros

    //Mostramos la informacion del espectador
    System.out.println(e);

    //Generamos una fila y letra
    //Si esta libre continua sino busca de nuevo
    do {

        fila = Metodos.generaNumeroEnteroAleatorio(0,
            cine.getFilas() - 1);

        letra = (char) Metodos.generaNumeroEnteroAleatorio('A',
            'A' + (cine.getColumnas()-1));

    } while (cine.haySitioButaca(fila, letra));

    //Si el espectador cumple con las condiciones
    if (cine.sePuedeSentar(e)) {
        e.pagar(cine.getPrecio()); //El espectador paga
        el precio de la entrada
        cine.sentar(fila, letra, e); //El espectador se
        sienta
    }

    }
    System.out.println("");
    cine.mostrar(); //Mostramos la información del cine
    System.out.println("Fin");
}
}
```

Y al ejecutarlo:

```
Introduce el numero de filas
2
Introduce el numero de columnas
2
Introduce el precio de la entrada de cine
5
Introduce el numero de espectadores a crear
5
```

UNIVERSIDAD DE LA FRONTERA

Facultad de Ingeniería y Ciencias

Depto. Ciencias de Computación e Informática

Espectadores generados:

el nombre del espectador es Eufasia de 24 años y con 1.0 dolares en su bolsillo
el nombre del espectador es Diego de 26 años y con 6.0 dolares en su bolsillo
el nombre del espectador es Diego de 12 años y con 10.0 dolares en su bolsillo
el nombre del espectador es Eufasia de 24 años y con 4.0 dolares en su bolsillo
el nombre del espectador es Eufasia de 19 años y con 4.0 dolares en su bolsillo

Información cine

Pelicula reproducida: 'Mi vida' del director DDR, con una duracion de 90 minutos y la edad minima es de 16 años
Precio entrada: 5.0

Asiento: 2A y este asiento está vacio

Asiento: 2B y este asiento está vacio

Asiento: 1A y el nombre del espectador es Diego de 26 años y con 1.0 dolares en su bolsillo

Asiento: 1B y este asiento está vacio

Fin

Desafios:

- Modificar el codigo anterior agregando la funcion para que hayan distintas peliculas(con distintos nombres, directores, clasificacion de edad)
- modificar para que sean x salas(introducidas por el usuario).

Ejercicos Java:

El restaurante australiano de José cuya especialidad es la sopa sorpresa, pide diseñar un método con el que se pueda saber cuántos clientes pueden atender con la materia prima que tienen en el almacén. El método recibe la cantidad de papas y chocolate en kilos y devuelve el número de clientes que puede atender el restaurante teniendo en cuenta que por cada tres personas, José utiliza un kilo de papas y medio de chocolate.