public class ***N****ombre* {

public static void main (String [] args) {} }

System.out.Print();

PrintIn (salto de línea)

Print (sin salto de línea)

Datos primitivos:

int / long / double / float (F) al final

char = 'a' / String = "xxx"

boolean = true/false

Secuencias:

\r\n → cambio de línea

\t → Tabulador

Casting:

double num = (int) num; Quita decimales

int num2 = (int) num; trata como un entero al doublé.

Operadores:

= asignación

== comparación

!= distinto de

&& → "Y" se cumplen ambas condiciones

|| → "O" con solo cumplir una condición

Operaciones simplificadas:

a = a + b → a+=b // a= a-b → a-=b

a*=b // a/=b

Scanner:

import java.util.Scanner;

- Scanner *teclado* = new Scanner (System.in);
- *variable* = *teclado*.nextInt();

Bucles:

```
for (int i = x; i<= y; i++){
        Instrucciones;
}
```

```
do{
        instrucciones;
}while (condición);
```

```
while (condición) {
        Instrucciones;
}
```

```
if (condición){
        Instrucciones;
}else{
        instrucciones;
 }
```

```
switch (variable que se introduzca) {
        case 0: instrucción;
                break;
        case 1:  instrucción;
                break;
        :       :       :
        default: instrucción;
}
```

Métodos Estáticos

```
static int/char/string/boolean nombre (argumentos){
        Instrucciones;
        return  valor;
}
Static void nombre (argumentos){
        Instrucciones,
        System.out.print;
 }
```

Clase OBJETO:

LOS CONSTRUCTORES

```
public  NombredeClase (String x, int y){
        this.x = x;
        this.y = y;
 }
```

GET AND SET

- ❑ SET:

```
Public void setNombre(String Nombre){
        this.nombre = nombre;
}
```

- ❑ GET:

```
public String getNombre(){
        return nombre;
}
```

METODOS:

```
public void Nombremetodo (parametros){
        instrucciones;
}
```

```
public int/string Nombremetodo(parametros){
        Instrucciones;
        return valor;
}
```

Arrays - Undimensional

Empieza en el 0

- Declarar + crear:

Tipo [] nombre_array = new tipo [num elementos]

-Declarar + crear + iniciar array:

Tipo dato [] nombre_array = {v1,v2…}

ACCEDER a un elemento de la matriz:

nombre_array [num elemento que accede]

Al primero: nombre_array [0]

Al último: nombre_array [n – 1]

MÉTODOS DE ARRAYS: (Import java.util.Arrays)

Nom_array.**lenght** → longitud del array

Arrays.**sort**(nom_array) → ordena el array teniendo en cuenta las mayúsculas en los strings.

EXCEPCIONES

Clases:

NumberFormatException → convertir una cadena a un tipo numérico pero la cadena no tiene el formato adecuado.

ArithmeticException → condición aritmética excepcional (como dividir por cero)

IndeoxOutOfBoundException → un índice de algun tipo esta fuera de rango (array, cadena, vector…)

InputMisMatchException → Lanzada por Scanner para indicar que el valor recuperado no coincide con el patrón para el tipo esperado.

IOException → Error en la entrada/salida

USO EXCEPCIONES

```
Try{
        El código que puede provocar errores;
} catch (Exception nombre) {
}
```

FORMATO DECIMALES

```
import java.text.DecimalFormat;
DecimalFormat formateador =
        new DecimalFormat("#,##0.00");
formateador.format(sueldo);
```

WRAPPERS

parseXxxx() permiten convertir un wrapper en un dato de tipo primitivo y le pasamos como parámetro el String.

```
double d4 = Double.parseDouble("3.14"); //
```

toString() permite retornar un String con el valor primitivo que se encuentra en el objeto contenedor.

```
Double d1 = new Double("3.14");
System.out.println(d1.toString() );
```

FECHAS

fecha como String con el formato que deseemos.

Convertir a fecha con la sentencia parse.

```
DateTimeFormatter formato =
        DateTimeFormatter.ofPattern("d/MM/yyyy");
String fechaCadena = "16/08/2016";
LocalDate mifecha =
        LocalDate.parse(fechaCadena, formato);
System.out.println(formato.format(mifecha));
```

Símbolo	Descripción
y	Año
D	Día del Año
M	Mes del Año
d	Día del Mes
Símbolo	Descripción
w	Semana del Año
EEE	Día de la Semana
F	Semana del Mes
a	AM/PM
K	Hora AM/PM (0-11)
H	Hora del día (0-23)
m	Minutos de la hora
s	Segundos del minuto
n	Nanosegundos del Segundo

STREAMS

Flujo de datos de lectura de caracteres.

```
FileReader fichero = new FileReader(nombreFichero);
fichero.read(); // castear con char() y -1 fin de fichero
fichero.close();
```

Flujo de datos de escritura de caracteres.

```
FileWriter fichero = new FileWriter(nombreFichero);
fichero.write(cadena.charAt(i));
fichero.close();
```

Flujo de datos de lectura con Buffer.

```
FileReader fichero = new FileReader(nombreFichero);
BufferedReader miBuffer =
        new BufferedReader(fichero);
cadena=miBuffer.readLine(); // nulll fin fichero
fichero.close();
```

Flujo de datos de escritura con Buffer.

```
FileWriter fichero =
   new FileWriter(nombreFichero,true); // true añade
BufferedWriter miBuffer =
        new BufferedWriter(fichero);
      fichero.write(cadena);
      fichero.close();
```

INTERFACE GRÁFICA

Métodos JFrame
setTitle(texto)
setSize(ancho,alto)
setLocation(x, y)
setBounds(x, y, ancho, alto)
setResizable(boolean)
setLocationRelativeTo(null)
SetDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
setVisible(boolean)

IMPLEMENTAR LA LÓGICA DEL BOTÓN.

a) Importar event.

```
import java.awt.event.*;
import javax.swing.*;
```

b) Implementar a interface ActionListener

c) Poner botón a la escucha.

```
miBoton.addActionListener(this);
```

d) Sobreescibir ActionPerformed

```
@Override
public void actionPerformed(ActionEvent e) {
        :       :       :       :
}
```

ARRAY Y LISTAS

Arrays o Vectores.

```
tipo[] nombreArray = new tipo[numelementos];
```

Arrays multidimensionales o matrices.

```
tipoDato[][] nombreArray = new
tipoDato[filas][columnas];
```

```
tipoDato[][] nombreArray = { {valor1, valor2, valor3} ,
                             {valor4, valor5,valor 6}
                           };
```

Listas.

```
ArrayList<tipoElementos> nombreLista = new
ArrayList<>();
```

ArrayList dispone de los siguientes métodos:

- add: añade un elemento. add(elemento) o add(posicion, elemento)
- get: retorna un elemento. get(posicion)
- remove: elimina un elemento. remove(elemento) o remove(posicion)
- set: cambia un elemento. set(posicion, valor)
- size: longitud de la lista. size()

Recorrer un ArrayList

```
for(int i=0; i<miListaNombres.size();i++){
   System.out.println(miListaNombres.get(i));
}
```

CONEXIÓN A BASE DE DATOS

Crear Conexion a la base de datos

```
Connection miConexion =
DriverManager.getConnection("jdbc:mysql://localhos
t:3306/basedatos","root","");
```

Crear y ejecutar sentencia de consulta SQL

```
Statement sentencia =
miConexion.createStatement();
ResultSet resultado =
sentencia.executeQuery("Sentencia SELECT");
```

Recorrer ResultSet

```
while (resultado.next()) {
        :       :       :
}
```

Cerrar Conexión

```
miConexion.close();
```

Insertar, Eliminar o Actualizar registros.

```
Statement sentencia =
miConexion.createStatement();
sentencia.executeUpdate("Sentencia INSERT,
UPDATE o DELETE");
```

Consultas Preparadas.

```
PreparedStatement sentencia =
miConexion.prepareStatement("SELECT * FROM
PELICULAS WHERE TEMA=? AND PRECIO > ?");
```

método setString(númeroParametro, valor).

Crear base de datos Orientada a Objetos.

```
ObjectContainer base;
base = Db4oEmbedded.openFile("nomfichero.db4o");
```

Cerrar la base de datos Objetos

```
base.close();
```

Almacenar objetos en la base de datos Objetos.

```
base.store(objeto);
```

Crear y ejecutar consulta QBE.

Objeto instanciaObjeto = new Objeto(0 o null todos los campos menos los que establecemos condición);

```
ObjectSet resultado =
base.queryByExample(instanciaObjeto);
```

Bucle listar objetos de la base de datos.

```
while(resultado.hasNext()) {
        :       :       :       :
}
```

Actualizar objetos en la base de datos .

- Recuperar objeto casteado
- settear atributos
- volver a almacenar con store().

Borrar Objetos de la Base de Datos.

```
base.delete(objeto);
```

Consultas Nativas (NativeQuery NQ).

```
Predicate<Objeto> consulta=new
Predicate<Objeto>(){
        @Override
        public boolean match(Objeto o){
                return (condicion con getters);
        }
};
```

Ejecutar consulta NQ

```
ObjectSet resultado = base.query(consulta);
```

Recorrer resultado de ObjectSet

```
while(resultado.hasNext()) {
        :       :       :       :       :
}
```