Tarea 02. Sintaxis. Tipos de datos, operadores y estructuras de control.

INTRODUCCIÓN

LECTURA POR TECLADO

Para leer datos de entrada, por ejemplo un entero, se puede hacer con un objeto de tipo ***Scanner****. Además de esta clase, existe una clase java.io.**Console** (JDK 6+), muy útil para leer datos desde consola (escribir contraseñas sin mostrar datos, etc…) pero desde el IDE como Netbeans no se tiene acceso a ella, no directamente(la ventana "Output" no es propiamente un terminal).* ***En esta tarea lo haremos con la clase Console*** *(que si lo admiten IDEs como Visual Studio Code).*

SCANNER

https://www.w3schools.com/java/java_user_input.asp
https://docs.oracle.com/en/java/javase/19/docs/api/java.base/java/util/Scanner.html

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt(); // Existe método diferente para cada tipo.
sc.nextLine(); /* En el caso de querer leer un línea después de
leer valores numéricos */
```

El fichero fuente debe contener la importación del paquete ***java.util.Scanner****;*. Se pone antes de la declaración de la clase (y después de la sentencia *package*):

```
import java.util.*; // o import java.util.Scanner
```

Si lo deseamos, para evitar tener que escribir la lectura de la línea y controlar que se introduce un dato numérico, podemos leer la línea entera y convertirla a entero:

```
try {
   entero = Integer.parseInt(sc.nextLine());
} catch (Exception e) {
```

```
}
```

Él método ***nextLine()*** avanza hasta la línea siguiente.

Existen otros métodos interesantes, como:

next(), que obtiene el siguiente elemento de entrada procedente del flujo. Para ello emplea cualquier separador implícito (salto de línea, espacio tabulador, etc…), pero puede cambiarse el separador de elementos por defecto:

```
Scanner sc = new Scanner(System.in).useDelimiter("\\s*filos\\s*");
```

Que, delimitaría por la palabra "filos", con cualquier espacio antes o después.

Para otros tipos de datos: ***nextLong()***, ***nextDouble()***, etc.
O leer mientras haya elementos:

```
while (sc.hasNext()) {
    ...
}
```

CONSOLE

https://docs.oracle.com/en/java/javase/19/docs/api/java.base/java/io/Console.html

Desde la versión 6 de Java existe una clase para leer datos desde consola, *java.io.**Console***. Para obtener el objeto *Console* se invoca al método estático System.console();

```
Console consola = System.console();
```

La lectura de la línea se puede hacer con el método "readLine(…)" de Console.

```
String nome = console.readLine();
```

Si queremos, podemos añadir el mensaje en el propio método de lectura:

```
String nome = console.readLine("introduce tu nombre: ");
```

Además, existe un método "readPassword(…)", que permite leer una clave sin mostrar los caracteres que se escriben ni guardarlo como cadena (hacerlo así implicaría un problema de seguridad). Por ejemplo:

```
Console cons;
char[] contrasinal;
if ((cons = System.console()) != null &&
    (contrasinal = cons.readPassword("[%s]", "Password:")) != null) {
    ... comprobaciones
    java.util.Arrays.fill(passwd, ' '); // limpeza
}
```

Podemos hacer una lectura condicional con Scanner si no hay acceso a la consola o, una vez tenemos acceso a ella, pasarle el lector al Scanner:

```
Console cons = System.console();
 if (cons != null) {
     Scanner sc = new Scanner(con.reader());
     ...
 }
```

BUFFEREDREADER

https://docs.oracle.com/en/java/javase/19/docs/api/java.base/java/io/BufferedReader.html

La lectura por teclado de BufferedReader la veremos en detalle en la unidad de flujos de E/S.

Ejemplos de ***BufferedReader*** (que permite leer de mucha fuente de datos, como Scanner):

```
BufferedReader br = new BufferedReader(
      new StringReader("Cadena que voy a leer"));

BufferedReader br = new BufferedReader(
      new FileReader("archivo.txt"));

BufferedReader br = new BufferedReader(
      new InputStreamReader(System.in))

Socket socket = new Socket(hostName, portNumber);
BufferedReader br = new BufferedReader(
      new InputStreamReader(socket.getInputStream()));
```

MÉTODOS ESTÁTICOS

En java, como en otros lenguajes, para poder reutilizar código, **se emplean "funciones" o "métodos" que realizan operaciones comunes** (entre muchas otras cosas). Además, permite estructurar de una manera ordenada el programa y hacer depuraciones de manera sencilla.

En el entorno Java, los métodos estáticos, **"static", son funciones globales** que se **invocan** precedidas del nombre de la clase (*parseInt*, de Integer, es un ejemplo, o funciones matemáticas como Math.sqrt(…), Math.pow(…),…) o, si estamos en el programa principal, directamente:

https://www.geeksforgeeks.org/static-method-in-java-with-examples/
https://www.geeksforgeeks.org/static-methods-vs-instance-methods-java/
https://www.w3schools.com/java/java_class_methods.asp
https://docs.oracle.com/javase/tutorial/java/javaOO/classvars.html

Formato de declaración de un método estático (para la presente tarea):

```
public static [tipo devuelto] [nombre método](tipo1 arg1, …){
        //..
}
```

En donde el **tipo devuelto** puede ser un **tipo de dato básico** (***byte, int,…***), nada/vacío (***void***) o un **objeto** (String, Integer,…).

Ejemplos:

```
public static boolean isLeapYear(short year) {
        // …
 }

public static byte getDiasMes(short year, byte month) {
        // …
}
```

La invocación sería desde otro método estático de la propia clase, como el main (en nuestro caso):

```
byte días = getDiasMes(2022, 10);
```

También puede invocarse desde otra clase precediendo el nombre del método estático del nombre de la clase.

En la presente tarea crearemos métodos estáticos para cada apartado y los invocaremos desde el método ***main*** (que es estático y el inicio del programa).

La sentencia switch

La sentencia switch ha tenido grandes cambios a partir de la versión 7. **Hasta la versión 7 sólo se podían usar valores enteros en la expresión switch**:

```
int nota = 5;
switch (nota) {
    case 1: case 2: case 3: case 4:
        System.out.println("Suspenso");
        break;
    default:
        System.out.println("Aprobado");
}
```

En la **versión 8** se permitió emplear cadenas y enumeraciones en la sentencia switch:

```
String diaS = "luns";
switch (diaS) {
    case "luns":
        System.out.println("Monday en inglés");
        break;
    // ...
    default:
        System.out.println("Día desconocido");
}
```

O con enumeraciones:

```
enum Disciplina {
    FILOSOFÍA, MATEMÁTICAS, LINGUA, LITERATURA, PINTURA, MÚSICA, OUTRAS
}

Disciplina interes = Disciplina.FILOSOFÍA;
switch (interes) {
    case FILOSOFÍA:
        System.out.println("Gústache o pensamento");
        break;
    case MÚSICA:
        System.out.println("Gústache as sensacións e o ritmo.");
        break;
    //…
    default:
        System.out.println("Tes outros intereses");
}
```

En Java 12 se introdujeron muchas novedades:

- **Devolver valores** de un bloque switch y realizar expresiones switch.
- Poder **poner múltiples valores en un "case"** de switch.
- Devolver valores de un switch por medio de un **operador flecha** o mediante

la palabra "**break**" (***yield*** en Java 13+).

Ejemplos:

a) Con ***break***: la sentencia "break" puede emplearse para devolver un valor.

```
return switch (diaS) {
    case "luns":
        break "Monday en inglés";
    // ...
    default:
        break "Día desconocido";
}
```

En Java 13 el break fue sustituido por la palabra ***yield***:

```
return switch (diaS) {
    case "luns":
        yield "Monday en inglés";
    // ...
    default:
        yield "Día desconocido";
}
```

b) Con el operador **flecha** (->): Java 12 introdujo operadores -> como alternativa sencilla para devolver valores:

```
return switch (diaS) {
    case "luns" -> "Monday en inglés";
    // ...
    default -> "Día desconocido";
}
```

c) Múltiples etiquetas en case:

```
return switch (diaS) {
    case "luns", "martes", "mércores", "xoves", "venres" -> "Día
semana";
    default -> "Guay!";
}
```

En la versión 14 se hizo que todos estos comportamientos fueran permanentes (antes eran *preview* y había que habilitar la marca -enable-preview como verdadera).

La **versión 17 LTS** de Java (15 septiembre del 2021) añade patrones (expresiones regulares) y nulos (https://medium.com/nipafx-news/pattern-matching-switching-null-guarded-patterns-26-hours-of-java-and-accento-78fb32d198d5)

a) ***Pattern matching***: puede ponerse coincidencia de patrones en una etiqueta.

Permite pasar objetos a una expresión switch y comprobar el tipo de objeto en las etiquetas del "case":

```
return switch (objeto) {
        case Integer i -> "Es un entero";
        case String s -> "Es una cadena";
        case Compositor comp -> "Es un compositor";
        default -> "No reconozco ese tipo de objeto";
};
```

b) ***Guarded Patterns***: permite hacer comprobaciones sobre el objeto en el propio "case".

```
return switch (objeto) {
        case Integer i -> "Es un entero";
        case String s -> "Es una cadena";
        case  Compositor  comp  &&  comp.getPeriodo().equals("Barroco")->
        "Colega de Bach";
        default -> "No reconozco ese tipo de objeto";
};
```

c) ***Casos nulos***: permite manejar nulos en case.

```
        case null -> "Ni compositor ni nada";
```

https://docs.oracle.com/en/java/javase/13/language/switch-expressions.html

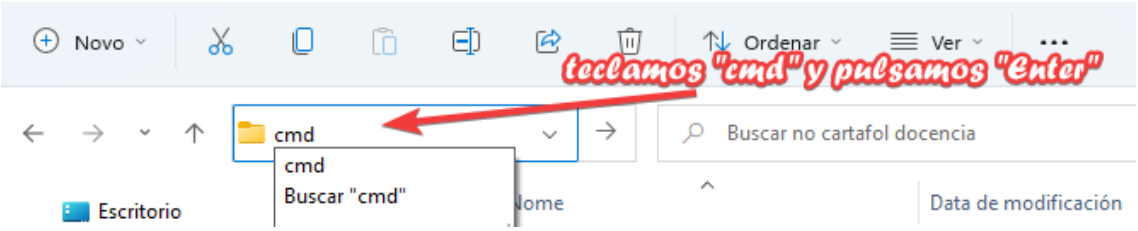
Calendario proporcionando mes y año

Se trata de realizar un programa en Java, ***JCalendario***, que imprima el calendario solicitando el mes y el año.

El programa debe cumplir los siguientes requisitos:

- La **lectura de teclado debe realizarse con la clase *Console*.**
- El año y el mes deben **guardarse en variables del tipo mínimo necesario** para almacenar los valores (byte, short,..)
- Debe estar en el **paquete *local.<tunombre>.programacion.tarea02***.
- Todos los métodos estáticos de la práctica deben estar en el mismo archivo ***JCalendario.java***.
- La **compilación y ejecución debe realizarse desde la línea de órdenes/terminal**., mostrando capturas de la ejecución y compilación de cada apartado.

Truco: en Windows, si queremos abrir una consola de órdenes del sistema en una carpeta concreta, simplemente tecleamos cmd na barra direcciones del explorador de archivos y pulsamos ·Enter:



Nos facilitará el acceso a la carpeta con el programa fuente desde la línea de órdenes.´

```
E:\programas\docencia>java com.pepinho.programacion.tarea02.AppCalendario
Introduce el mes: 10
Introduce el año: 2022
OUTUBRO del 2022
-------------------------------------------------
| Luns | Mar. | Mér. | Xov. | Ven. | Sáb. | Dom. |
|-------------------------------------------------
|      |      |      |      |      |  01  |  02  |
-------------------------------------------------
|  03  |  04  |  05  |  06  |  07  |  08  |  09  |
-------------------------------------------------
|  10  |  11  |  12  |  13  |  14  |  15  |  16  |
-------------------------------------------------
|  17  |  18  |  19  |  20  |  21  |  22  |  23  |
-------------------------------------------------
|  24  |  25  |  26  |  27  |  28  |  29  |  30  |
-------------------------------------------------
|  31  |
```

Nota: cada apartado será evaluado como APTO o NO APTO.

APARTADO 1. ENUMERACIONES (1 PUNTO)

Crea un enumeración llamada ***Mes*** con los meses en *galego* y mayúscula. La enumeración debe contener, además, los días del mes, de modo que se puedan construir/declarar en la enumeración del siguiente modo:

XANEIRO(31), FEBREIRO(28), MARZO(31), ABRIL(30), …, DECEMBRO(31);

A mayores, crea un método ***getDias()*** que devuelva el número de días del mes *(consulta el ejemplo de la presentación de los apuntes y los planetas).*

APARTADO 2. MÚLTIPLOS SIN OPERADOR MÓDULO (%) (1 PUNTO)

Realícese un método estático, ***isMultiple***, que recoja un número y un divisor, devolviendo **verdadero cuando el número es divisible** por el divisor (el resto es cero), falso en caso contrario.

*NOTA: **no puede emplearse el operador módulo (%)**. Usa operaciones aritméticas básicas (división, multiplicación y resta) y operadores de comparación. Si te parece interesante, puedes hacerlo con un bucle y el operador "resta" (o "suma"). Pienso en un algoritmo sencillo de cómo hacerlo.*

APARTADO 3. AÑO BISIESTOS (1 PUNTO)

Realícense dos métodos estáticos que recojan un año de tamaño ***short*** y devuelva si es o no bisiesto. Previamente debe comprobar que el año sea superior al 1583 (fecha en la que se instauró el calendario gregoriano).

Para comprobar **si el año es múltiplo de 4 no debe emplearse el operador módulo**, use cada uno de los siguientes operadores para comprobar que sea múltiplo de 4:

a) Operador **& a nivel de bit**. El método debe llamarse ***isLeapYearAND***.
b) Operadores desplazamiento y comparación. El método debe llamarse ***isLeepYearShift***

*Ayuda: recuerda que **los múltiplos de 4 son aquellos que en binario terminan en "00"**.*

APARTADO 4. NOMBRE DEL MES (1 PUNTO)

Crea un método ***getNomeMes*** que recoja el mes como byte y devuelva el nombre del mes de la enumeración. *Ayuda: Mes.values()[numeroMes-1] devuelve el valor en la enumeración del mes con número numeroMes.*

APARTADO 5. DÍAS DEL MES (1,5 PUNTOS)

Realiza dos métodos estáticos que recojan el **año** como short y el **mes** como byte, devolviendo los días del mes. Recuerda que si es un año bisiesto y el mes es febrero debe sumarle 1 al día del mes (haz uso del método anteriormente creado). Debes hacer dos versiones:

a) ***getDiasMes***, que emplea una expresión *switch* con operador flecha (->) y múltiples opciones en el case (añadido en Java 12), al menos, o para versiones siguientes.
b) ***getDiasMesEnum***, que obtiene el valor del mes del mismo modo que en el método anterior, por medio del método "values()".

APARTADO 6. DÍA DE LA SEMANA. (2 PUNTOS)

Se trata de crear un método ***getDayOfWeek*** para obtener el día de la semana como número, recogiendo como parámetros el **día**, **mes** y **año**. En la que el número de la semana va de 0 (domingo) a sábado (6).

Uno de los algoritmos más populares es el **algoritmo de *Tomohiko Sakamoto*** *Algorithm, Wang and Schwerdtfeger* (adaptación de la versión de Brian Kernighan/Dennis Ritchie de C)

El algoritmo puede representarse así:

```
díaDeLaSemana(año, mes, día)
Inicio
      Si el mes es menor que 3 entonces
            Descontamos al año una unidad.
      Fin si
      Numero <-   (año + año / 4 - año / 100 + año / 400 +
valorEnTabla(mes) + día
      dia = Número módulo 7
Fin
```

En el que `valorEnTabla` es una "función/sentencia" que devuelve un número ("e" en la imagen) dependiendo del valor del mes:

m	1	2	3	4	5	6	7	8	9	10	11	12
e	0	3	2	5	0	3	5	1	4	6	2	4

Haga el método estático ***getDayOfWeek*** para devuelva el día de la semana como número, **sin emplear arrays** (pueden usarse sentencias de selección que consideres), pues no forma parte del estudio de esta unidad.

APARTADO 7. IMPRIMIR EL MES DEL CALENDARIO A PARTIR DE UN AÑO Y MES.

Haga un programa método estático ***printCalendario*** y un programa/***main*** dentro de la misma clase, *JCalendario*, que pida por consola el año y el mes, mostrando el calendario por pantalla. Para ello, cree el método estático que recoja el año y mes e imprima el mes en una tabla (tal y como aparece en la imagen previa)

El programa debe poder **recoger desde línea de órdenes el mes y el año**. Si no se proporcionan, sólo en ese caso, debe pedirlos con un mensaje. Ejemplo de ejecución:

java -jar jcalendario.jar 2022 10

Ayuda:

Téngase en cuenta que debe imprimir tantas celdas vacías iniciales hasta que toque el primer día del mes.

Además, debe imprimir un salto de línea y una barra horizontal cada 7 días del mes (precisas de un contador).

ENTREGA

Genérese un único documento **PDF con el código del programa y las capturas de compilación y ejecución.** Además, debe proporcionar el **código fuente, *JCalendar.java*, y archivo *jcalendario.jar*** para poder ser ejecutado, tal y como se muestra en el ejemplo anterior.

Entréguese un archivo comprimido de extensión ZIP o 7Z con:

- PDF con formato *Apellido1Apellido2NombreTarea02PROG.pdf*
- Código fuente: ***JCalendario.java***
- Archivo comprimido para ejecutar: ***jcalendario.jar***.

El formato de documento debe ser:

*Apellido1Apellido2NombreTarea02PROG.zip (o *.7z)*

Por ejemplo:

WittgensteinLudwigJosefTarea02PROG.zip (sólo tiene un apellido)