Tarea online 2

Pulse el enlace para ver la tarea a pantalla completa

UNIÓN EUROPEA
Fondo Social Europeo

Tarea online

Título de la tarea: PROG02. Uso de estructuras de control de flujo.

Unidad: 02

Ciclo formativo y módulo: DAM/DAW, Programación.

Curso académico: 2021/2022

¿Qué contenidos o resultados de aprendizaje trabajaremos?

El principal resultado de aprendizaje que se va a trabajar con esta tarea será:

- RA3. Escribe y depura código, analizando y utilizando las estructuras de control del lenguaje.

Esto significa que se trabajará esencialmente con los siguientes contenidos:

- Uso de estructuras de selección.
- Uso de estructuras de repetición.
- Depuración de programas.

Estado de la entrega

Esta tarea aceptará entregas desde el **miércoles, 10 de noviembre de 2021, 00:00**

Número del intento	Este es el intento 1.
Estado de la entrega	No entregado
Estado de la calificación	Sin calificar
Fecha de entrega	viernes, 12 de noviembre de 2021, 23:55
Fecha límite	domingo, 14 de noviembre de 2021, 23:55
Última modificación	-
Comentarios de la entrega	Comentarios (0)

◄ Examen 2

Ir a...

1.- Descripción de la tarea.

Caso práctico

María continúa con su aprendizaje y su entrenamiento.

Le siguen asignando pequeños problemas a resolver, que luego serán integrados en aplicaciones más complejas. María se los toma como pequeños retos.

Poco a poco va aumentando la dificultad de las tareas que le proponen, y como se ve insegura todavía, ha decidido que va a repasar un poco las estructuras de control de flujo resolviendo algunos problemas típicos, cogiendo soltura en la depuración de los mismos, etc.

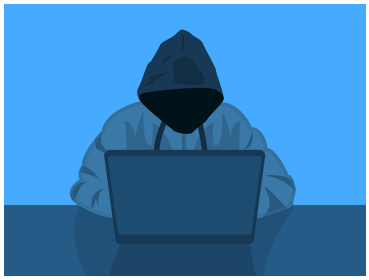
¿Qué te pedimos que hagas?

Esta tarea consiste en resolver una pequeña relación de ejercicios en los que se usan las estructuras de control de flujo. Antes de empezar a trabajar con los ejercicios, **lee bien los enunciados,** así como los apartados **"Información de interés"** y **"Evaluación de la tarea"**. Es importante que revises todo esto antes de empezar a resolver los ejercicios, y también antes de llevar a cabo la entrega para asegurarte de que **cumples con todo lo que ha pedido**. Revisa bien el apartado de evaluación y asegúrate de que estás entregando lo que se te pide y de la forma que se te pide (tipos y formato de los documentos, uno o varios proyectos, clases o archivos Java, etc.).

No debe haber más que un único archivo fuente `.java` por cada ejercicio. Dada la cantidad de líneas que se necesita para construir cada programa, de momento no necesitamos más. Todo lo que no sea eso es una complicación gratuita. Y las complicaciones gratuitas, en programación, penalizan.

Como siempre, **se valorará en todos los casos la corrección ortográfica y gramatical de los mensajes para comunicarnos con el usuario, así como la presentación clara de cualquier información que se muestre por pantalla.**

1.1.- Ejercicio 1: uso del depurador.



Shafin_Protic (Licencia Pixabay)

En este ejercicio **no se pide que elabores código en Java**, pues ya se te proporciona en el archivo **Ejercicio01.java** que se encuentra en el proyecto base que se os proporciona con la tarea. Tu misión consistirá en **seguir una serie de pasos** que se te indican en un documento que tendrás que usar como **ficha de la tarea**. **Para documentar cada paso tendrás que realizar una captura de pantalla de tu entorno Netbeans** que servirá como evidencia de que has realizado apropiadamente ese paso.

Para llevar a cabo cada uno de esos pasos deberás utilizar algunas de las funcionalidades que ofrece **la herramienta Netbeans como entorno de depuración** para programas escritos en Java.

Aquí tienes el documento con el **modelo de ficha**, en versión .odt (odt - 25.77 KB) o versión .docx (docx - 11.49 KB), que debes rellenar. Además, se te proporciona un **documento de ejemplo** (pdf - 587.4 KB) en el que podrás observar cómo debería quedar tu ficha una vez cumplimentada con todas las capturas. Debes usarlo como modelo para generar tus propias capturas.

No debes modificar el código del programa que se te proporciona (**Ejercicio01.java**) salvo la **línea 40**, donde en lugar de dejar el texto: "*PRUEBA DEL ALUMNO NOMBRE APELLIDO1 APELLIDO2*" **deberás indicar tu nombre y apellidos**.

En la primera captura debes incluir también, como fondo, tu *login* en la plataforma, donde se pueda observar tu nombre y tu foto, para así comprobar que el trabajo lo has realizado tú. En el resto de capturas no es necesario, pero sí debes capturar siempre la **línea 40** (siempre que se trate de una captura de código) para que se pueda observar tu nombre y comprobar que todas las capturas las has ido realizando tú.

Procura ceñirte a la **estructura y las páginas del documento modelo** y a incluir las capturas en cada página que se indica tal y como se muestra en el PDF proporcionado. Tu documento debe intentar parecerse al máximo al PDF salvo por la aparición de tu nombre en el código (línea 40).

Una vez hayas terminado de cumplimentar el documento, genera un PDF a partir de él para evitar problemas de formato a la hora de corregirlo. ¡Y no olvides incluir ese documento en el paquete junto con el resto de tu proyecto!

Código que debes depurar

Aunque el código que debes depurar se te proporciona ya en el archivo **Ejercicio01.java** que se encuentra en el proyecto base, aquí lo tienes nuevamente por si quieres echarle un vistazo.

Mostrar retroalimentación

```java
package tarea2;

import java.util.Scanner;

/**
 * Ejercicio 1: uso del depurador.
 *
 * Recuerda que lo único que debes modificar es la línea en la que se escri
 * tu nombre por pantalla.
 *
 * @author Profe
 */
public class Ejercicio01 {

    public static void main(String[] args) {

        //------------------------------------------------
        //          Declaración de variables
        //------------------------------------------------
        // Constantes
        final byte MIN_FILAS = 1;
        final byte MAX_FILAS = 10;

        // Variables de entrada
        int numFilas;

        // Variables de salida
        String cadenaResultado = "";

        // Variables auxiliares
        boolean entradaValida = false;
        byte numero = 1;

        // Clase Scanner para petición de datos
        Scanner teclado = new Scanner(System.in);

        //------------------------------------------------
        //                Entrada de datos
        //------------------------------------------------
        System.out.println("ESCALERA INCREMENTAL: PRUEBA DEL ALUMNO APELLI
        System.out.println("------------------------------------------------
        do {
            System.out.print("Introduzca número de filas (" +
                    MIN_FILAS + "-" + MAX_FILAS + "): ");
            numFilas = teclado.nextByte();
            if (numFilas >= MIN_FILAS && numFilas <= MAX_FILAS) {
                entradaValida = true;
            }
        } while (!entradaValida);

        //------------------------------------------------
        //                 Procesamiento
        //------------------------------------------------
        for (int fila = 1; fila <= numFilas; fila++) {
            cadenaResultado += fila + ": ";
            for (int columna = 1; columna <= fila; columna++) {
                cadenaResultado += numero + " ";
```

```
                    numero++;
                }
                cadenaResultado += "\n";
            }

            //-----------------------------------------------
            //              Salida de resultados
            //-----------------------------------------------
            System.out.println();
            System.out.println("RESULTADO");
            System.out.println("---------");
            System.out.println(cadenaResultado);
        }

    }
```

1.2.- Ejercicio 2: años y siglos.

Cristian Ferronato (Licencia Pixabay)

Escribe un programa en Java que solicite al usuario un **año** que debe encontrarse entre 1801 y 2100, ambos valores incluidos. Si el año introducido no es válido (no está en ese rango), se indicará que el año no es válido y finalizará el programa.

Si el año es válido, el programa procederá a **calcular la siguiente información**:

- **siglo** en el que se encuentra ese año (solo puede ser XIX, XX o XXI);
- **si el año es anterior, igual o posterior al año actual (2021)**. Si es anterior se calculará cuántos años han pasado desde entonces y si es posterior cuántos años faltan para llegar hasta él.

Una vez calculada toda esa información, se mostrará por pantalla con el siguiente formato:

- **si el año no era válido**, no se realizará ningún cálculo ni ninguna comprobación más y se indicará por pantalla que no es un año válido;
- **si el año es válido** se mostrará por pantalla:
  - **si el año coincide con el actual (2021)**, o bien si es **anterior o posterior**. En tales casos se indicará también **cuántos años han pasado o cuántos años faltan**;
  - **el siglo** al que pertenece el año.

Debes tener en cuenta que:

- se valorará que se **minimice el número de líneas y condiciones**. Intenta utilizar los **operadores lógicos** siempre que te sea posible;
- antes de comenzar a analizar el año, deberías primero comprobar que está dentro del rango válido. **Si el año no es válido, no tiene sentido realizar más comprobaciones**, no debería entrarse en el bloque de procesamiento y debería saltarse directamente a la salida de resultados donde se mostraría por pantalla un mensaje de error;
- **una vez que se establezca el siglo al que pertenece el año, no deberías seguir realizando comprobaciones**. No es necesario y sería ineficiente. Para ello puedes aprovechar el uso de la estructura **"if/elseif"**;
- **si has hecho la comprobación de dos posibles siglos y el año no corresponde con ellos, no es necesario que hagas una tercera comprobación**. Sería redundante pues sabrás seguro que estás en el tercer siglo posible, dado que no hay otra posibilidad. Para eso tienes el **else**;
- **las únicas clases y métodos que puedes usar en el programa** son los de **System** y **Scanner**, que utilizamos para la E/S por consola. Aparte de estas, no puedes usar ninguna otra clase, método o herramienta, ni de la API de Java ni propia, para resolver el ejercicio;
- se deberán utilizar estructuras **"if/else"** o **"if/elseif/else"** obligatoriamente;
- **el primer año del siglo XIX es el 1801 y el último el 1900**; en el caso del **siglo XX**, sus años están **entre 1901 y 2000**, y el **siglo XXI** está compuesto por los años que van **desde el 2001 hasta el 2100**. Consideramos que no existen ni el año 0 ni el siglo 0, por tanto el siglo I iría desde el año 1 hasta el 100 y así sucesivamente.

Ejemplos de ejecución (I)

Aquí tienes algunos ejemplos de ejecución del programa con años válidos:

Para el año **1801**:

Mostrar retroalimentación

```
ANÁLISIS DEL AÑO
----------------
Introduzca un año (entre 1801-2100): 1801

RESULTADO
---------
El año introducido es anterior al actual. Han pasado 220 años.
El año pertenece al siglo XIX.
```

Para el año **1900**:

Mostrar retroalimentación

```
ANÁLISIS DEL AÑO
----------------
Introduzca un año (entre 1801-2100): 1900

RESULTADO
---------
El año introducido es anterior al actual. Han pasado 121 años.
El año pertenece al siglo XIX.
```

Para el año **2000**:

Mostrar retroalimentación

```
ANÁLISIS DEL AÑO
----------------
Introduzca un año (entre 1801-2100): 2000

RESULTADO
---------
```

```
El año introducido es anterior al actual. Han pasado 21 años.
El año pertenece al siglo XX.
```

Para el año **2021**:

Mostrar retroalimentación

```
ANÁLISIS DEL AÑO
----------------
Introduzca un año (entre 1801-2100): 2021

RESULTADO
---------
El año introducido coincide con el actual.
El año pertenece al siglo XXI.
```

Para el año **2100**:

Mostrar retroalimentación

```
ANÁLISIS DEL AÑO
----------------
Introduzca un año (entre 1801-2100): 2100

RESULTADO
---------
El año introducido es posterior al actual. Faltan 79 años.
El año pertenece al siglo XXI.
```

Ejemplos de ejecución (II)

Y aquí algunos otros con años no válidos:

Para el año **1800**:

Mostrar retroalimentación

```
ANÁLISIS DEL AÑO
----------------
Introduzca un año (entre 1801-2100): 1800

RESULTADO
---------
El año introducido no es válido.
```

Para el año **2101**:

Mostrar retroalimentación

```
ANÁLISIS DEL AÑO
----------------
Introduzca un año (entre 1801-2100): 2101

RESULTADO
---------
El año introducido no es válido.
```

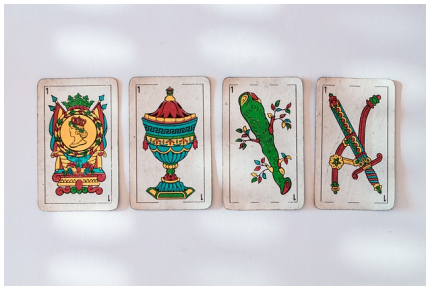
Para el año **-1**:

Mostrar retroalimentación

```
ANÁLISIS DEL AÑO
----------------
Introduzca un año (entre 1801-2100): -1

RESULTADO
---------
El año introducido no es válido.
```

1.3.- Ejercicio 3: valor de un naipe.

Estamos pensando en desarrollar una aplicación para simular el juego de naipes conocido como "*Tute*". El objetivo del juego consiste en sumar tantos y se emplea la baraja española de cuarenta cartas.



kevberon (Pixabay License)

En este juego cada naipe de la baraja tiene un valor, que es el siguiente:

Valor de los naipes en el juego del tute

Naipe	Puntuación
As (1)	11 puntos
Tres (3)	10 puntos
Sota (10)	2 puntos
Caballo (11)	3 puntos
Rey (12)	4 puntos
Resto de cartas (2, 4, 5, 6, 7)	0 puntos

Escribe un programa en Java que solicite un número de carta (del 1 al 7, o del 10 al 12) y que muestre por pantalla el valor de esa carta. En caso de que se introduzca un número de carta no válido, el resultado devuelto debe ser -1.

Ten en cuenta que en este programa:

- se deberá utilizar únicamente la **estructura** `"switch"` para el cálculo del valor del naipe. No pueden utilizarse estructuras `"if/else"` o `"if/elseif/else"` ni el operador condicional (`?`). El objetivo de este ejercicio es que practiques con el uso de la estructura `"switch"`;
- se valorará que se **minimice el número de líneas** en el interior la sentencia `switch`. Procura que cuando haya varias circunstancias (cláusulas `case`) en las que se deba realizar la misma acción, estén todas agrupadas y se lleve a cabo esa acción una sola vez;
- las únicas clases y objetos que puedes usar en el programa son `System` y `Scanner`, junto con sus métodos, que utilizamos para la E/S por consola. Aparte de estas, no puedes usar ninguna otra clase, objeto o método, ni de la API de Java ni propio, para resolver el ejercicio;

Ejemplos de ejecución

Aquí tienes algunos posibles ejemplos de ejecución del programa.

Para **números de naipe no válidos**, como por ejemplo el 0, el 8 o el 13.

Mostrar retroalimentación

```
VALOR DE UN NAIPE EN EL TUTE
----------------------------
Introduzca número (1-7, 10-12): 0

RESULTADO
---------
El valor del naipe con ese número es: -1



VALOR DE UN NAIPE EN EL TUTE
----------------------------
Introduzca número (1-7, 10-12): 8

RESULTADO
---------
El valor del naipe con ese número es: -1



VALOR DE UN NAIPE EN EL TUTE
----------------------------
Introduzca número (1-7, 10-12): 13

RESULTADO
---------
El valor del naipe con ese número es: -1
```

Para **números de naipe válidos**.

Mostrar retroalimentación

```
VALOR DE UN NAIPE EN EL TUTE
----------------------------
Introduzca número (1-7, 10-12): 1

RESULTADO
---------
El valor del naipe con ese número es: 11
```

```
VALOR DE UN NAIPE EN EL TUTE
----------------------------
Introduzca número (1-7, 10-12): 2

RESULTADO
---------
El valor del naipe con ese número es: 0



VALOR DE UN NAIPE EN EL TUTE
----------------------------
Introduzca número (1-7, 10-12): 3

RESULTADO
---------
El valor del naipe con ese número es: 10



VALOR DE UN NAIPE EN EL TUTE
----------------------------
Introduzca número (1-7, 10-12): 4

RESULTADO
---------
El valor del naipe con ese número es: 0



VALOR DE UN NAIPE EN EL TUTE
----------------------------
Introduzca número (1-7, 10-12): 7

RESULTADO
---------
El valor del naipe con ese número es: 0



VALOR DE UN NAIPE EN EL TUTE
----------------------------
Introduzca número (1-7, 10-12): 10

RESULTADO
---------
El valor del naipe con ese número es: 2



VALOR DE UN NAIPE EN EL TUTE
----------------------------
Introduzca número (1-7, 10-12): 11

RESULTADO
```

```
---------
El valor del naipe con ese número es: 3



VALOR DE UN NAIPE EN EL TUTE
----------------------------
Introduzca número (1-7, 10-12): 12

RESULTADO
---------
El valor del naipe con ese número es: 4
```

1.4.- Ejercicio 4: puntuación de un texto.

Free-Photos (Pixabay License)

Estamos planteando el desarrollo de una aplicación para simular juegos de tipo "*Scrabble*", donde cada palabra tiene una puntuación en función de las letras que la componen.

Escribe un programa en Java que solicite al usuario un texto de al menos cinco caracteres de longitud. Si el texto no contiene al menos cinco caracteres debe volver a pedirse hasta que finalmente se introduzca un texto con esa longitud como mínimo.

Tras la introducción de un texto que cumpla los requisitos exigidos, se solicitará un número entero que servirá como "multiplicador", y que debe estar obligatoriamente entre 1 y 3. Nuevamente, si se introduce cualquier otro valor que no se encuentre en ese rango, habrá que volver a solicitarlo hasta que se proporcione una cantidad válida.

Una vez que se disponga del texto y el multiplicador, se procederá al cálculo de la puntuación inicial del texto sumando la puntuación obtenida por cada carácter individual, teniendo en cuenta las siguientes reglas:

- cada **vocal**, tanto si es minúscula como mayúscula, valdrá **un punto**;
- las **letras** `x`, `y`, `z` **(minúsculas)**, valdrán **dos puntos**;
- la **letra X (mayúscula)**, valdrá **cinco puntos**;
- **cualquier otro carácter** (tanto si es letra como si es otra cosa), **restará un punto**.

Una vez obtenida esa puntuación inicial, deberás multiplicarla por el valor "multiplicador" para calcular la **puntuación final**.

En este programa **no pueden utilizarse estructuras** `"if/else"` **o** `"if/elseif/else"` **ni el operador condicional** (`?`). Parte del objetivo de este ejercicio es que continúes familiarizándote con el uso de la estructura "`switch`";

Como siempre, se valorará que se **minimice el número de líneas** en el interior la sentencia `switch`.

> Recuerda que para obtener la **longitud de una cadena de caracteres** se utiliza la herramienta `cadena.length()`. Para obtener **cada carácter de una cadena** dispones de la herramienta `cadena.charAt(pos)`, donde `pos` será un número entero entre 0 y la longitud de la cadena menos uno.

Ejemplos de ejecución

Aquí tienes algunos posibles ejemplos de ejecución del programa.

Mostrar retroalimentación

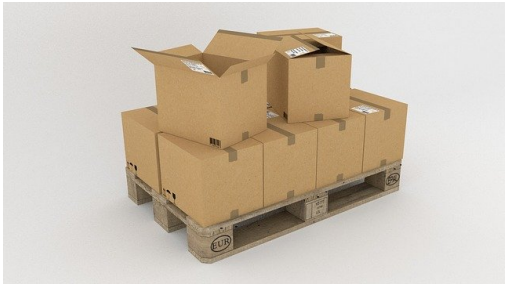
```
VALOR DE UN TEXTO
-----------------
Introduzca un texto con al menos 5 caracteres: a
Introduzca un texto con al menos 5 caracteres: aa
Introduzca un texto con al menos 5 caracteres: aaa
```

```
Introduzca un texto con al menos 5 caracteres: aaaa
Introduzca un texto con al menos 5 caracteres: aaaaa
Introduzca el valor del multiplicador (entre 1-3): 0
Introduzca el valor del multiplicador (entre 1-3): 4
Introduzca el valor del multiplicador (entre 1-3): 1

RESULTADO
---------
El valor del texto es: 5



VALOR DE UN TEXTO
------------------
Introduzca un texto con al menos 5 caracteres: 123456
Introduzca el valor del multiplicador (entre 1-3): 3

RESULTADO
---------
El valor del texto es: -18
```

Y aquí una batería de casos de prueba algo más amplia:

Texto	ABCDE	AEIOUaeiou	Ejemplo	Extra	EXTRA	%1234	XYZBC	xyzbc	Extra
Multiplicador	1	1	1	1	1	1	1	1	3
Puntuación	-1	10	-1	2	5	-5	1	4	6

1.5.- Ejercicio 5: construcción de cajas.



Francesco Romeo (Licencia Pixabay)

Se desea disponer de una aplicación que permita mostrar por pantalla rectángulos creados mediante caracteres. A esos rectángulos los llamaremos "*cajas*".

Escribe un programa en Java que solicite por teclado un **número de filas** y un **número de columnas** (en los dos casos entre 2 y 10, ambos valores incluidos). Además, **se preguntará también al usuario si desea que se use relleno o no**.

Si el número de filas o de columnas está fuera del rango considerado como válido (2-10) deberá volver a solicitarse hasta que sean valores correctos. Respecto a la pregunta de si se desea el uso de relleno o no, se contestará introduciendo **un 0 si no se desea relleno** (caja vacía) o cualquier otro número si desea relleno.

Una vez se disponga de toda la información de entrada y esta sea válida se procederá a la **construcción de una cadena de caracteres** (acumulador) que represente la caja **concatenando** los siguientes caracteres:

- para cada una de las cuatro **esquinas**, se utilizará el **carácter** '+';
- para los **bordes superior e inferior**, el **carácter** '-';
- para los **bordes izquierdo y derecho**, el **carácter** '|';
- para el interior:
  - si la caja va sin rellenar (**vacía**), el **carácter espacio** (' ');
  - si la caja va **rellena**, **caracteres numéricos que indiquen el número de fila (sin tener en cuenta el borde superior)**. Es decir, caracteres '1' para toda la primera fila de relleno, caracteres '2' para la segunda fila de relleno, y así sucesivamente.

Por ejemplo, para el caso de una caja de 5 filas y 7 columnas vacía (sin relleno), se generaría la siguiente cadena:

```
+-----+
|     |
|     |
|     |
+-----+
```

Mientras que si fuera con relleno tendría el siguiente aspecto:

```
+-----+
|11111|
|22222|
|33333|
+-----+
```

Observa la colocación de los caracteres que forman las esquinas, los bordes y el relleno en caso de que lo haya.

Como ya se ha indicado, **si se introduce una cantidad de filas o de columnas que no se encuentra en el rango permitido** (2-10, ambos inclusive), **el programa volverá a solicitar ese valor hasta que sea correcta tantas veces que sea necesario.** Para llevar a cabo este control te recomendamos utilizar bucles `do-while`.

Una vez que se haya construido la cadena con el rectángulo debes mostrarla por pantalla como resultado final del programa. **Para construir la caja debes utilizar bucles** `for`.

Recuerda que para que un `String` incluya un "*salto de línea*" basta con concatenar (operación `+`) a esa cadena un carácter de salto de línea (`"\n"`): `cadena += "\n";`

Ejemplos de ejecución

Aquí tienes algunos posibles ejemplos de ejecución del programa

Ejemplo 1: caja de 2x2 vacía y rellena.

Mostrar retroalimentación

Rellena:

```
CONSTRUCCIÓN DE CAJAS
---------------------
Introduzca número de filas (2-10): 1
Introduzca número de filas (2-10): 11
Introduzca número de filas (2-10): 2
Introduzca número de columnas (2-10): 1
Introduzca número de columnas (2-10): 11
Introduzca número de columnas (2-10): 2
¿Caja rellena? (0: vacía, cualquier otro número rellena) 0

RESULTADO
---------
Caja de tamaño 2x2 vacía:

++
++
```

Sin rellenar:

```
CONSTRUCCIÓN DE CAJAS
---------------------
Introduzca número de filas (2-10): 2
Introduzca número de columnas (2-10): 2
¿Caja rellena? (0: vacía, cualquier otro número rellena) 1

RESULTADO
---------
Caja de tamaño 2x2 rellena:

++
++
```

Como puedes observar son iguales pues todo son bordes (de hecho solo esquinas) y no ha quedado espacio interior para poder rellenar.

Ejemplo 2: cajas de tamaño 6x2 y 2x6.

Mostrar retroalimentación

Caja de tamaño 6x2:

```
CONSTRUCCIÓN DE CAJAS
---------------------
Introduzca número de filas (2-10): 6
Introduzca número de columnas (2-10): 2
¿Caja rellena? (0: vacía, cualquier otro número rellena) 0

RESULTADO
---------
Caja de tamaño 6x2 vacía:

++
||
||
||
||
++
```

Caja de tamaño 2x6:

```
CONSTRUCCIÓN DE CAJAS
---------------------
Introduzca número de filas (2-10): 2
Introduzca número de columnas (2-10): 6
¿Caja rellena? (0: rellena, cualquier otro número vacía) 1

RESULTADO
```

```
---------
Caja de tamaño 2x6 rellena:

+----+
+----+
```

Nuevamente se trata de cajas en las que todo son bordes y esquinas, de manera que serían exactamente iguales con o sin relleno.

Ejemplo 3: cajas de tamaño 5x10 vacías y rellenas.

Mostrar retroalimentación

Caja de tamaño 5x10 vacía:

```
CONSTRUCCIÓN DE CAJAS
---------------------
Introduzca número de filas (2-10): 5
Introduzca número de columnas (2-10): 10
¿Caja rellena? (0: vacía, cualquier otro número rellena) 0

RESULTADO
---------
Caja de tamaño 5x10 vacía:

+--------+
|        |
|        |
|        |
+--------+
```

Caja de tamaño 5x10 rellena:

```
CONSTRUCCIÓN DE CAJAS
---------------------
Introduzca número de filas (2-10): 5
Introduzca número de columnas (2-10): 10
¿Caja rellena? (0: vacía, cualquier otro número rellena) 1

RESULTADO
---------
Caja de tamaño 5x10 rellena:

+--------+
|11111111|
|22222222|
|33333333|
+--------+
```

Ejemplo 4: cajas de tamaño 10x5 vacías y rellenas.

Mostrar retroalimentación

Caja de tamaño 10x5 vacía:

```
CONSTRUCCIÓN DE CAJAS
---------------------
Introduzca número de filas (2-10): 10
Introduzca número de columnas (2-10): 5
¿Caja rellena? (0: vacía, cualquier otro número rellena) 0

RESULTADO
---------
Caja de tamaño 10x5 vacía:

+---+
|   |
|   |
|   |
|   |
|   |
|   |
|   |
|   |
+---+
```

Caja de tamaño 10x5 rellena:

```
CONSTRUCCIÓN DE CAJAS
---------------------
Introduzca número de filas (2-10): 10
Introduzca número de columnas (2-10): 5
¿Caja rellena? (0: vacía, cualquier otro número rellena) 2

RESULTADO
---------
Caja de tamaño 10x5 rellena:

+---+
|111|
|222|
|333|
|444|
|555|
|666|
|777|
```

```
|888|
+---+
```

2.- Información de interés.

Recursos necesarios y recomendaciones

A continuación se os proporcionan algunas recomendaciones y consejos importantes sobre la realización de la tarea para que podáis llevarla a cabo con éxito:

mohamed Hassan (Licencia Pixabay)

- El código de vuestros programas debe seguir la **estructura de programa** que ya se vio en la unidad anterior y su tarea. Salvo que se indique lo contrario, siempre **será obligatorio**escribir vuestros programas usando esa estructura. Recordad todas las indicaciones al respecto que se os dieron en la tarea anterior y no olvidéis descargar el proyecto que debéis usar como guía. En él dispondréis de esa estructura preparada para cada ejercicio: Proyecto base para la tarea 2 (zip - 20.99 KB) .
- Uso de los **casos de prueba**. ¡Muy importante! Ya se ha hablado sobre esta cuestión en la tarea anterior. Si no lo recuerdas, vuelve a leer lo que se dice al respecto en esa tarea. Y ten siempre en cuenta que **los ejemplos y casos de prueba son para usarlos y probar nuestro programa**. No son un "adorno" para "rellenar" el texto del ejercicio.
- Para llevar a cabo las **capturas** (*screenshots*) que es necesario realizar en el **ejercicio 1**, os recomendamos la aplicación *Greenshot* o cualquier otra similar en **Sistemas Operativos Windows**. Dispones de unas instrucciones detalladas de las capturas que debes realizar en el documento de ficha en el que debes incluir esas capturas.
- En caso de que estés trabajando en una plataforma **Mac OSX**, puedes utilizar las siquientes combinaciones de teclas (puedes encontrar más información aquí):
  - Para relizar una captura de toda la pantalla en Mac OSX pulsa simultáneamente las teclas **Shift + Comand + 3**
  - Para relizar una captura de parte de la pantalla en Mac OSX pulsa simultáneamente las teclas **Shift + Comand + 4**
  - Para realizar la captura de una ventana o menú en Mac OSX pulsa simultáneamente las teclas **Shift + Comand + 4 + espacio**
- Para realizar capturas de pantalla en **Linux** puedes utilzar las siguientes combinaciones de teclas más usuales, aunque dependiendo de la distribución de linux o actualizaciones pueden no funcionar (puedes encontrar más información aquí):
  - **ImprPant**: Guarda una captura de todo el escritorio en la carpeta «Imágenes».
  - **Shift + ImprPant**: nos permite seleccionar un trozo de la pantalla y guarda la captura en «Imágenes».
  - **Alt + ImprPant:** Guarda una captura de la ventana activa en «Imágenes».
  - **Ctrl + ImprPant**: Copia la captura de toda la pantalla en el portapapeles.
  - **Shift + Ctrl + ImprPant:** Copia la captura de un trozo de la pantalla en el portapapeles.
  - **Ctrl + Alt + ImprPant:** Copia la captura de la ventana activa en el portapapeles.
- **Utilizad el depurador** para descubrir los errores que se producen en tus programas. De hecho, tendréis que usarlo explícitamente para resolver el primer ejercicio, pero que os recomendamos que utilicéis normalmente en vuestro trabajo como programadores para ayudaros a localizar y corregir errores en vuestro código. Podríamos parafrasear la famosa expresión "*El perro es el mejor amigo del hombre*" como "*El depurador es el mejor amigo del*

programador". Es fundamental que aprendáis a utilizarlo pues vuestro rendimiento como desarrolladores de aplicaciones se verá incrementado exponencialmente.

- Siempre que lo consideréis oportuno, **incluid comentarios útiles en el código**. Os resultará de utilidad en el futuro si tenéis que revisar ese código para llevar a cabo algún tipo de modificación.
- Los enunciados de cada ejercicio suelen contener toda la información necesaria para su resolución. **Cualquier cosa que no tengáis clara, se la preguntáis "al cliente"** que en este caso es el profesorado, aunque se valorará también mostrar cierta soltura interpretando los enunciados de una manera natural. Siempre tendréis a vuestra disposición los **foros** y el **correo** de la plataforma, así como la asistencia telefónica si fuera necesario. En cualquier caso, toda suposición que se haga al resolver el problema debe estar correctamente documentada en el código mediante comentarios y debe ser razonable y razonada.
- Recuerda que **las estructuras de salto incondicional deben ser evitadas** (a no ser que sea totalmente imprescindible, y no es el caso en estos ejercicios). Se trata de elementos no deseables en un programa estructurado y normalmente pueden ser sustituidas por las otras estructuras de control que sí cumplen las reglas básicas de la programación estructurada. En estos ejercicios no vas a tener que usarlas. Su uso podría llevarte a la anulación completa del ejercicio en el que se utilicen.
- **Indenta** tu código de forma adecuada. Es algo que se tendrá en cuenta a la hora de evaluar la tarea. Recuerda que NetBeans puede hacer esto por ti (selecciona el código a "*formatear*" o "*embellecer*" y pulsa **Alt+Mayús.+F**).

Indicaciones de entrega

Una vez completados todos los ejercicios de la tarea, el envío se realizará a través de la plataforma. Comprime la carpeta del proyecto NetBeans en un fichero .zip y nómbralo siguiendo las siguientes pautas:

Apellido1_Apellido2_Nombre_PROG02_Tarea

¡No olvides incluir en ese paquete de entrega el documento PDF que se te pide elaborar en el **primer ejercicio** (*uso del depurador*)!

3.- Evaluación de la tarea.

Criterios de evaluación implicados

En esta unidad se va a evaluar el resultao de aprendizaje RA3. "Escribe y depura código, analizando y utilizando las estructuras de control del lenguaje." Ello se hará mediante los siguientes Criterios de Evaluación correspondientes al citado resultado de aprendizaje

- a. Se ha escrito y probado código que haga uso de estructuras de selección.
- b. Se han utilizado estructuras de repetición.
- c. Se han reconocido las posibilidades de las sentencias de salto.
- e. Se han creado programas ejecutables utilizando diferentes estructuras de control.
- f. Se han probado y depurado los programas.
- g. Se ha comentado y documentado el código.

Peggy_Marco (Pixabay License)

Nota: El criterio de evalaución (d. Se ha escrito código utilizando control de excepciones) correspondiente al RA3, se trabajará en la unidad 5.

¿Cómo valoramos y puntuamos tu tarea?

La rúbrica de evaluación se publicará en las próximas semanas, en cuanto tengamos las directrices necesarias desde la dirección general de FP