Tarea online 01

Título de la tarea: Introducción a la programación.

Unidad: 01

Ciclo formativo y módulo: DAM/DAW, Programación.

Curso académico: 2022/23

¿Qué contenidos o resultados de aprendizaje trabajaremos?

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

- RA1. Reconoce la estructura de un programa informático, identificando y relacionando los elementos propios del lenguaje de programación utilizado.

Esto significa que se trabajará esencialmente con los siguientes contenidos:

- Entornos integrados de desarrollo.
- Estructura y bloques fundamentales.
- Variables. Tipos de datos. Literales. Constantes.
- Operadores y expresiones. Conversiones de tipo.
- Comentarios.

1.- Descripción de la tarea.

Caso práctico

María está todavía empezando a practicar con el lenguaje Java, y realizando los primeros programas, como una práctica, dentro del equipo de desarrollo.

Ahora tiene que resolver una serie de pequeños programas cuyo código formará parte de aplicaciones mayores, de momento son solo pequeñas rutinas sencillas, que una vez que estén probadas y funcionando, se las debe pasar a su compañero **Juan** para que las reutilice como mejor vea. De momento lo que es fundamental es que elija bien el identificador para las variables, el tipo de datos, y el uso de los operadores aritméticos, lógicos, etc.,... y que todo funcione con normalidad.

Juan le ha dicho que de momento no se preocupe siquiera de controlar los errores en la entrada, puesto que de eso se va a ocupar el módulo desde el que se ejecute el código que está haciendo **María**.

¿Qué te pedimos que hagas?

En esta tarea hay varios ejercicios para resolver e implementar como programas en **Java** usando el IDE **NetBeans**, declarando las variables necesarias con cuidado de elegir los tipos más adecuados para cada caso y de nombrarlas siguiendo la nomenclatura que se recoge en los convenios comúnmente aceptados y que se han visto en los contenidos de la unidad.

Los ejercicios se incluirán cada uno en una clase, con su método `main`, en un **único proyecto** NetBeans para facilitar la corrección. La plantilla de ese proyecto base se os proporciona en la sección información de interés.

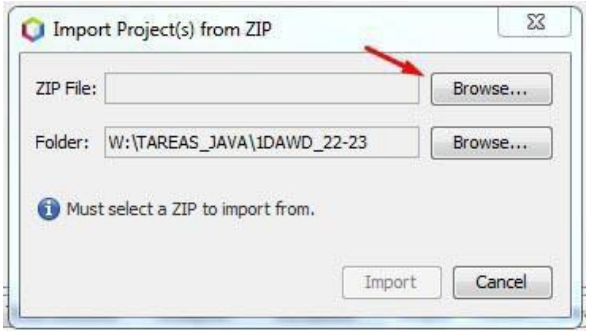
Se valorará en todos los casos la corrección ortográfica y gramatical de los mensajes para comunicarnos con el usuario, así como la presentación clara de cualquier información que se muestre por pantalla.

1.1.- Ejercicio 0: crear los paquetes sobre el proyecto base.

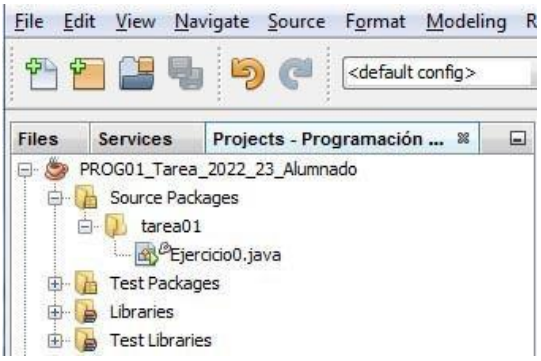
Este apartado no es un ejercicio propiamente dicho sino una preparación y explicación general para que puedas preparar el proyecto y ponerte en marcha. Como podrás ver, en el apartado 2 de la tarea (Información de Interés) os dejamos un **Proyecto Base**, comprimido en formato ZIP, que debéis descargar. Este proyecto base os ayudará a dar vuestros primeros pasos con el entorno de NetBeans.

Podéis cargar este proyecto en NetBeans a través del menú **File > Import Project > From ZIP**

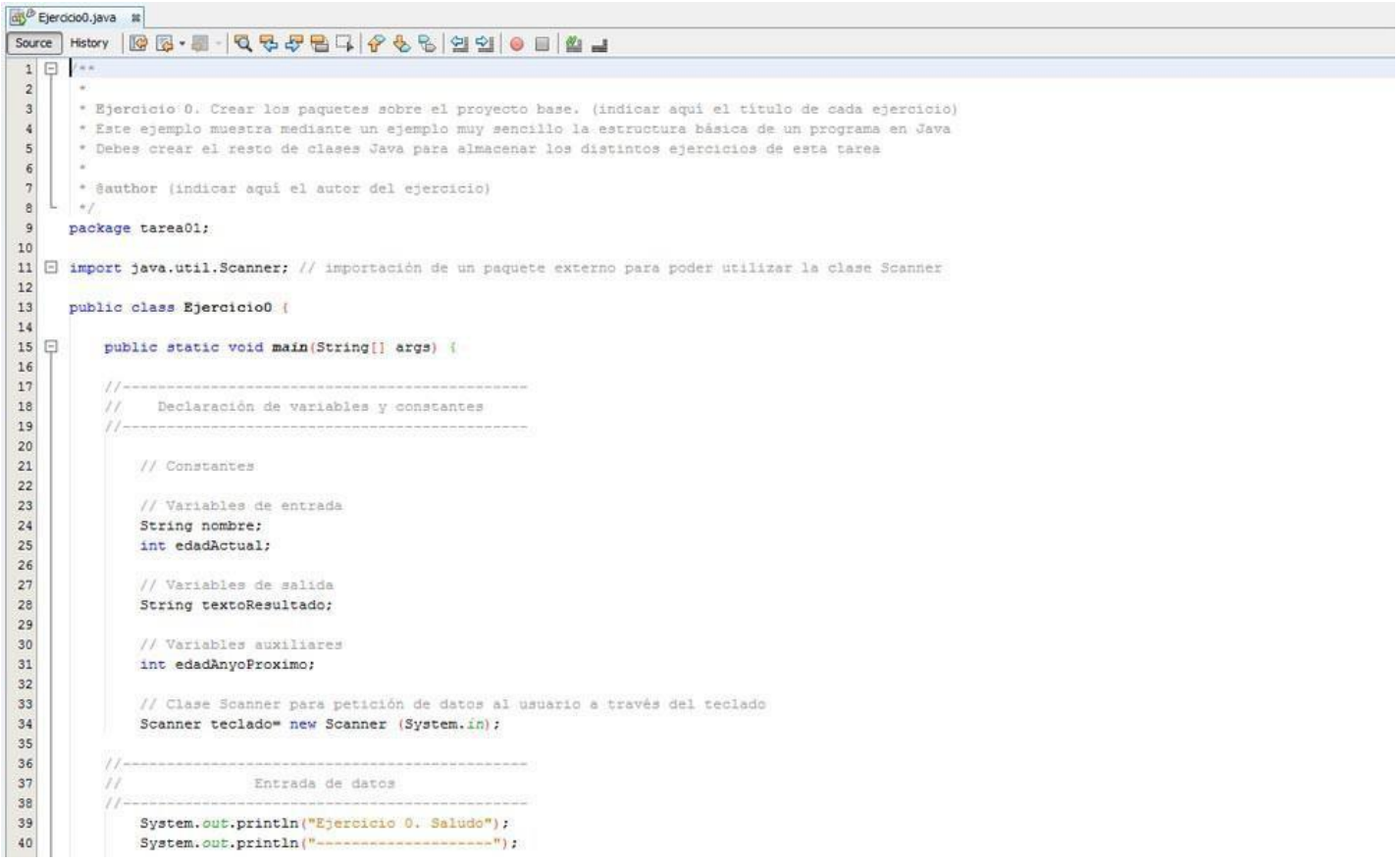
En la siguiente ventana debéis pulsar **Browse...** para localizar el fichero ZIP del proyecto base que habréis descargado (el fichero se llama **PROG01_Tarea_2022_23_Alumnado.zip**). Además, podéis usar el botón **Browse...** de abajo para elegir la carpeta donde colocar vuestros proyectos Java.



Una vez importado el proyecto, podréis ver su contenido en el **Explorador de Proyectos** situado en la zona lateral izquierda de la ventana principal de NetBeans (si no estuviera visible, podréis mostrarla pulsando **CTRL + 1** o a través del menú superior **Window > Projects**).



En la imagen puedes ver el contenido inicial del proyecto base que se os ha proporcionado. Este proyecto se llama **PROG01_Tarea_2022_23_Alumnado**, contiene un *paquete* llamado **tarea01** y una clase Java llamada **Ejercicio0.java**. Esta clase es un programa ejecutable (si te fijas, su icono tiene un triángulo verde) de ejemplo muy sencillo, pero que muestra la estructura general que debes seguir para organizar tu código de forma adecuada (puedes consultar el apartado **7.5.1.- Estandarización del código** de la unidad para obtener más información).



```java
/**
 *
 * Ejercicio 0. Crear los paquetes sobre el proyecto base. (indicar aquí el título de cada ejercicio)
 * Este ejemplo muestra mediante un ejemplo muy sencillo la estructura básica de un programa en Java
 * Debes crear el resto de clases Java para almacenar los distintos ejercicios de esta tarea
 *
 * @author (indicar aquí el autor del ejercicio)
 */
package tarea01;

import java.util.Scanner; // importación de un paquete externo para poder utilizar la clase Scanner

public class Ejercicio0 {

    public static void main(String[] args) {

    //------------------------------------------------
    //    Declaración de variables y constantes
    //------------------------------------------------

        // Constantes

        // Variables de entrada
        String nombre;
        int edadActual;

        // Variables de salida
        String textoResultado;

        // Variables auxiliares
        int edadAnyoProximo;

        // Clase Scanner para petición de datos al usuario a través del teclado
        Scanner teclado= new Scanner (System.in);

    //------------------------------------------------
    //              Entrada de datos
    //------------------------------------------------
        System.out.println("Ejercicio 0. Saludo");
        System.out.println("--------------------");
```

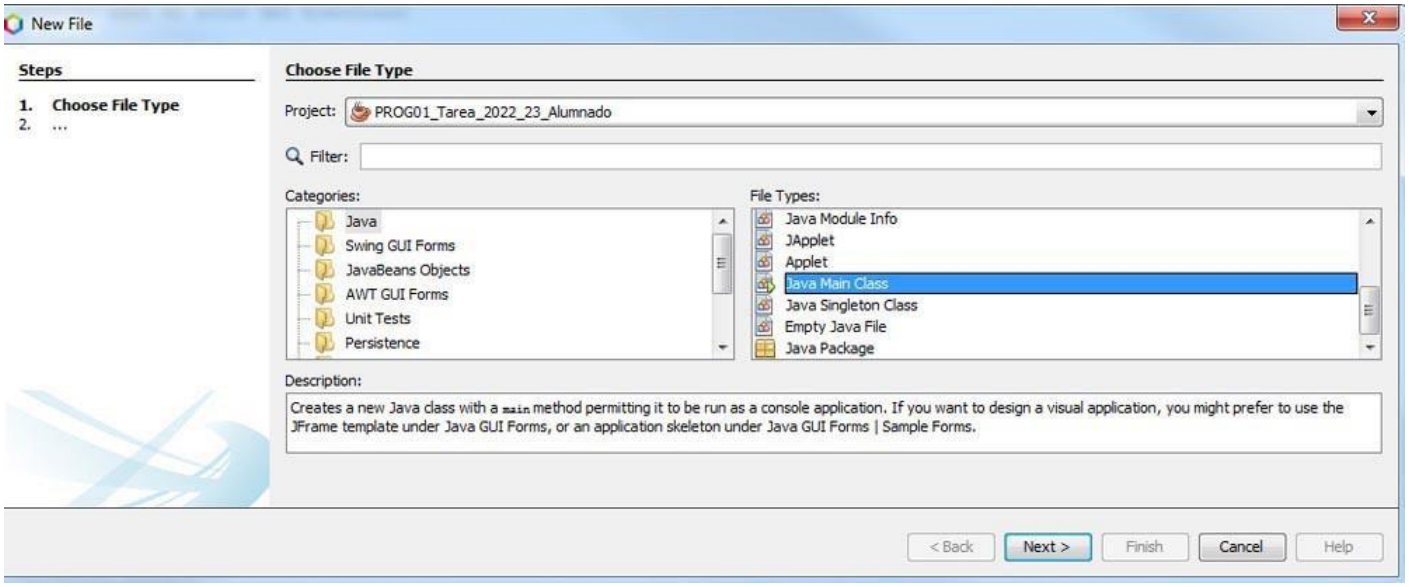
Observa que el ejercicio que se aporta como ejemplo sigue la **misma estructura del código** indicada en los contenidos de la unidad (*Declaración de variables, Entrada de datos, Procesamiento y Salida de resultados*). El resto de ejercicios que realices en esta tarea **también deberán seguir esa estructura**.

Dedica un momento a analizar el código del ejemplo y, posteriormente, prueba su funcionamiento haciendo clic en la opción del menú superior **Run > Run File** o pulsando la combinación de teclas **Mayúsculas + F6**. Puesto que el ejemplo pide datos por teclado a través de la clase Scanner, deberás completar en la zona de la consola los datos que se solicitan:

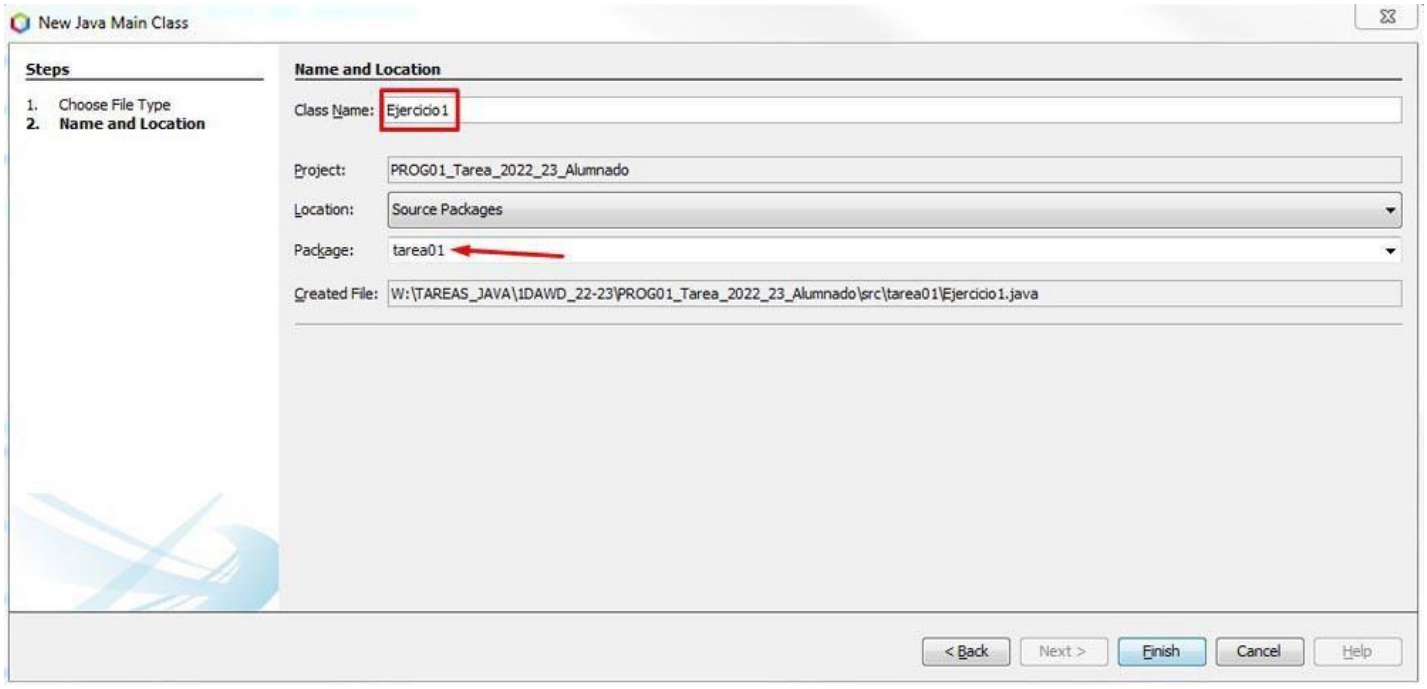
```
Search Results | Test Results | Analyzer | Variables | Output - PROG01_Tarea_2022_23_Alumnado (run-single) | Notifications

ant -f W:\\TAREAS_JAVA\\1DAWD_22-23\\PROG01_Tarea_2022_23_Alumnado -Dnb.internal.action.name=run.single -Djavac.includes=tarea01/Ejercicio00.java
init:
Deleting: W:\TAREAS_JAVA\1DAWD_22-23\PROG01_Tarea_2022_23_Alumnado\build\built-jar.properties
deps-jar:
Updating property file: W:\TAREAS_JAVA\1DAWD_22-23\PROG01_Tarea_2022_23_Alumnado\build\built-jar.properties
Compiling 1 source file to W:\TAREAS_JAVA\1DAWD_22-23\PROG01_Tarea_2022_23_Alumnado\build\classes
compile-single:
run-single:
Ejercicio 0. Saludo
--------------------
Introduce tu nombre:
Maria
Introduce tu edad actual:
7

RESULTADO
---------
Hola Maria, ahora tienes 7 años, el año próximo tendrás 8 años

Fin del programa.
BUILD SUCCESSFUL (total time: 5 seconds)
```

Por último, vamos a crear una nueva clase Java para realizar el **Ejercicio 1** de esta tarea. Para ello, buscaremos en el menú superior la opción **File > New File...** tal y como se muestra en la imagen:

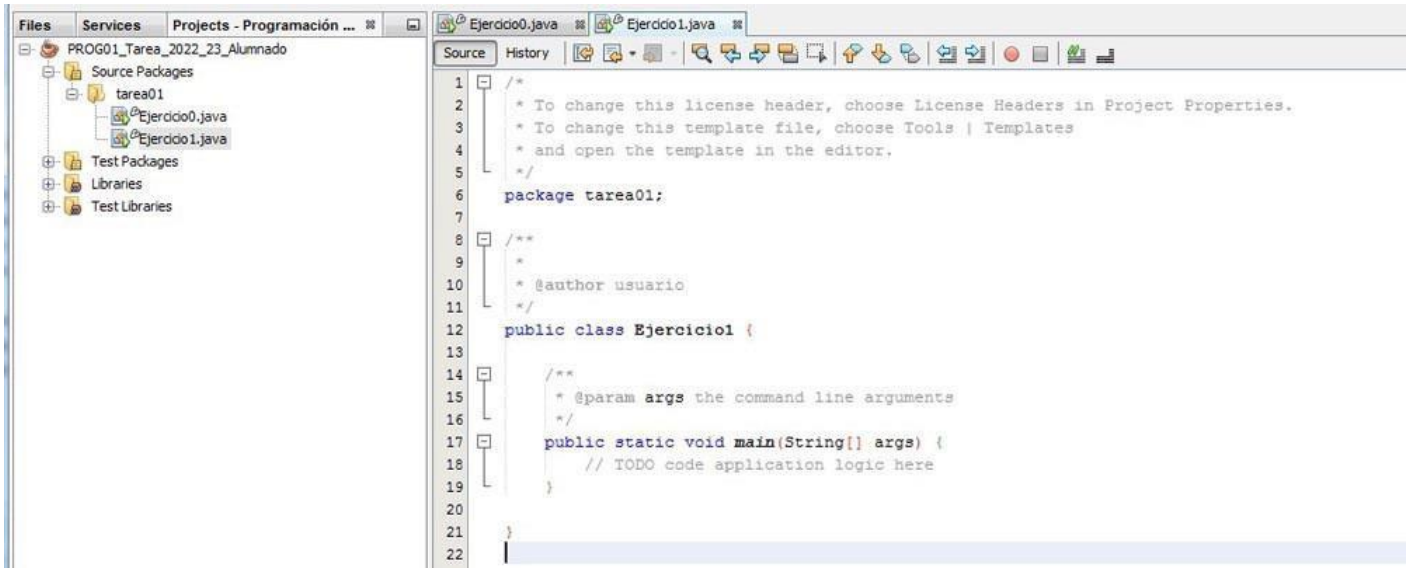
En la siguiente ventana elegiremos el tipo de clase que queremos crear, será una **Java Main Class** (será una nueva clase Java que incluirá el **método main**) ya que queremos que contega un programa sencillo que se pueda ejecutar de forma independiente:



Tras pulsar el botón **Next >**, debemos poner el nombre a la nueva clase que estamos creando. Para el Ejercicio 1, podemos llamar a la nueva clase **Ejercicio1** (no puedes usar espacios en blanco en el nombre de la clase)**.** Observa que, en el apartado **package**, debes seleccionar **tarea01** en el desplegable, ya que es el paquete que contendrá nuestros ejercicios en esta tarea**.**



Una vez pulses el botón **Finish** se creará la nueva clase principal para el ejercicio 1 (para el resto de ejercicios de esta tarea también debes crear sus propias clases principales: *Ejercicio2*, *Ejercicio3*, etc.).



Nota: Recuerda que debes mantener la estructura del código propuesta en el apartado **7.5.1.- Estandarización del código** de la unidad: *Declaración de variables, Entrada de datos, Procesamiento y Salida de resultados.*

Puedes copiar esa estructura desde ejemplo que aparece en el apartado 7.5.1 o desde el Ejercicio0 de esta tarea.

Una vez copiada la estructura propuesta, la clase para el Ejercicio1 nos quedaría más o menos así:

```
1
2   package tarea01;
3
4   import java.util.Scanner;
5
6   /**
7    * Ejercicio 1. Cálculo del área de un trapecio regular
8    * @author [nombre del alumno]
9    */
10  public class Ejercicio1 {
11
12      public static void main(String[] args) {
13      //-----------------------------------------------
14      //    Declaración de variables y constantes
15      //-----------------------------------------------
16
17          // Constantes
18
19          // Variables de entrada
20
21          // Variables de salida
22
23          // Variables auxiliares
24
25          // Clase Scanner para petición de datos al usuario a través del teclado
26          Scanner teclado= new Scanner (System.in);
27
28      //-----------------------------------------------
29      //               Entrada de datos
30      //-----------------------------------------------
31
32      //-----------------------------------------------
33      //                 Procesamiento
34      //-----------------------------------------------
35
36      //-----------------------------------------------
37      //              Salida de resultados
38      //-----------------------------------------------
39          System.out.println();
40          System.out.println("RESULTADO");
```

¡Ya tenemos la clase del primer ejercicio preparada! A partir de aquí te toca trabajar, pensar y **tratar de resolver los 5 ejercicios restantes** que se proponen en la tarea. Recuerda que puedes plantear cualquier duda que te surja utilizando el **Foro 01** de la Unidad.

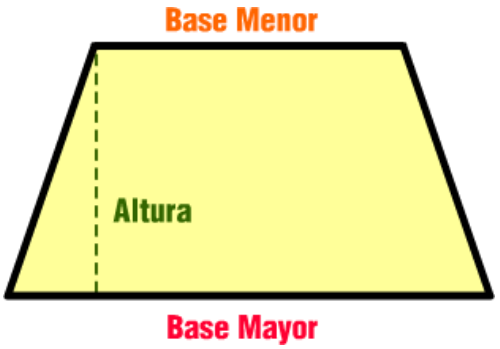
Happy coding!

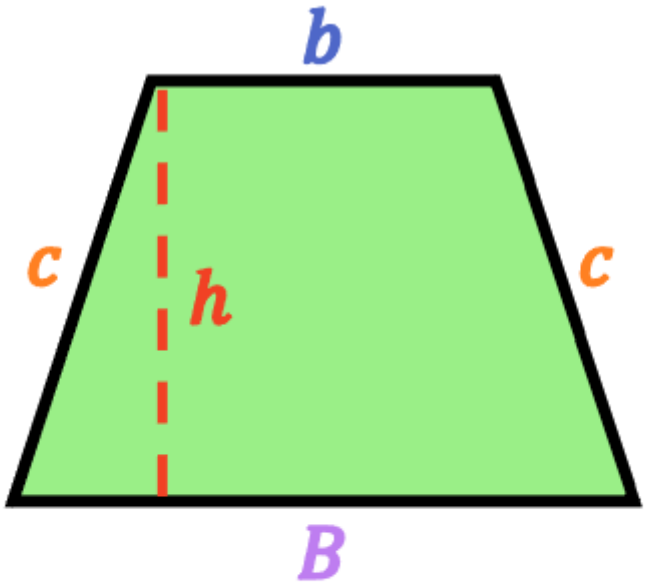
1.2.- Ejercicio 1: cálculo del área de un trapeci regular.

Realiza un programa Java que permita la introducción de tres números reales. Estos números representarán la **base menor**, la **base mayor** y la **altura** de un **trapecio regular**.

Una vez introducidos los números por teclado, el programa calculará el **área del trapecio regular** siguiendo la **fórmula** indicada y escribirá por pantalla el resultado.

La fórmula del área de un trapecio regular es:





$$A = h \cdot \frac{B + b}{2}$$

Licencia: Elaboración propia

Recuerda que, aparte del código que resuelva cada uno de los ejercicios propuestos en esta tarea, **debes incluir también los comentarios mínimos necesarios** para que otra persona que analice el código pueda seguirlo con facilidad. Recuerda que se trata de un **recurso fundamental para mejorar la legibilidad** de tu código.

Nota: Por ahora **no controlaremos que los valores introducidos tengan que ser positivos o no** (los valores negativos en nuestro ejemplo no tendrían sentido). Más adelante veremos cómo poder controlar estas cosas.

Ejemplos de ejecución

Aquí tienes algunos ejemplos de ejecución.

Mostrar retroalimentación

Un ejemplo de ejecución del programa podría ser:

```
Ejercicio 1. Cálculo del área de un trapecio regular
--------------------------------------------------------------------------------
Introduce la base mayor del trapecio: 5
Introduce la base menor del trapecio: 3
Introduce la altura del trapecio: 4

RESULTADO
---------------
El área del trapecio base menor 3.0, base mayor 5.0 y altura 4.0 es 16.0

Fin del programa.
```

Otro ejemplo de ejecución podría ser:

```
Ejercicio 1. Cálculo del área de un trapecio regular
--------------------------------------------------------------------------------
Introduce la base mayor del trapecio:  4,5
Introduce la base menor del trapecio: 3,2
Introduce la altura del trapecio: 6,3

RESULTADO
---------------
El área del trapecio base menor 3.2, base mayor 4.5 y altura 6.3 es 24.255

Fin del programa.
```

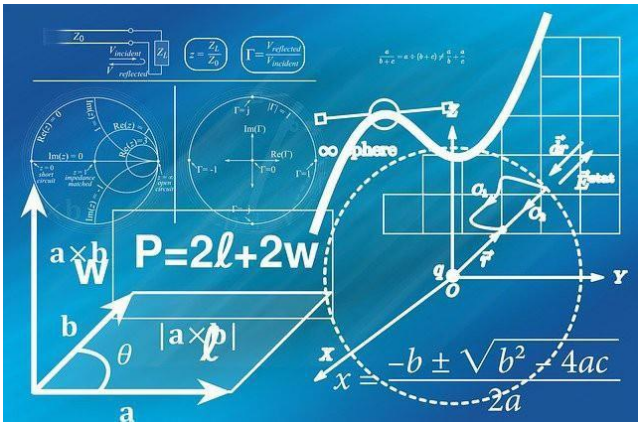
Y aquí tienes otro ejemplo más:

```
Ejercicio 1. Cálculo del área de un trapecio regular
--------------------------------------------------------------------------------
Introduce la base mayor del trapecio:  4,3
Introduce la base menor del trapecio: 2,3
Introduce la altura del trapecio: 0

RESULTADO
---------------
El área del trapecio base menor 2.3, base mayor 4.3 y altura 0.0 es 0.0

Fin del programa.
```

1.3.- Ejercicio 2: operadores aritméticos.

Realiza un programa Java que permita la **introducción de tres números enteros**. El programa realizará los cálculos necesarios para determinar:

- la suma de los dos primeros números divididos por el tercer número.
- si la suma de los tres números es par.
- si el resto del primer número dividido por el segundo número es distinto de 0.
- si la multiplicación del primer número por la mitad del segundo número es mayor que el tercer número.
- si el resultado del cuadrado del primer número menos el triple del segundo número por el tercer número es menor que 0.



Licencia: (Pixabay License)

Para realizar estas comprobaciones dispones de **operadores relacionales** tales como igual (==), menor que (<), mayor que (>), etc. Ten en cuenta que el resultado de la aplicación de operadores relacionales será un valor de tipo **boolean**, es decir un valor que será **true** o **false**.

Recuerda también que puedes obtener el **resto de una división entera entre dos números utilizando el operador módulo (%)**. Este operador también te puede ayudar para determinar si un número es **divisible** entre otro (si el resultado la división a%b es igual a 0, significara que a es divisible entre b).

Nota: Ten en cuenta que para resolver el ejercicio solo podrás utilizar operadores que hayamos visto durante esta unidad. No podrás utilizar métodos de otras clases que aún no hemos visto en los contenidos. En el caso de las divisiones, no es necesario que tengas en cuenta y controles los errores que pueden ocurrir al dividir entre 0, símplemente usa otros números (más adelante nos ocuparemos de eso).

Ejemplos de ejecución

Aquí tienes algunos ejemplos de ejecución.

Mostrar retroalimentación

Un ejemplo de ejecución del programa podría ser:

```
Ejercicio 2. Operadores aritméticos
-----------------------------------------------------
Introduce el primer número: 2
Introduce el segundo número: 2
Introduce el tercer número: 2

RESULTADOS
----------
Comprobamos el valor de la suma de dos primeros números dividida por el tercer número: 2.0
Comprobamos si la suma de los tres números es par: true
Comprobamos si el resto del primer número dividido por el segundo número es distinto de 0: false
Comprobamos si la multiplicación del primer número por la mitad del segundo número es mayor que el tercer nú
Comprobamos si el resultado del cuadrado del primer número menos el triple del segundo número por el tercer

Fin del programa.
```

Otro ejemplo de ejecución podría ser:

```
Ejercicio 2. Operadores aritméticos
-----------------------------------------------------------
Introduce el primer número: 3
Introduce el segundo número: 4
Introduce el tercer número: 1

RESULTADOS
----------
Comprobamos el valor de la suma de dos primeros números dividida por el tercer número: 7.0
Comprobamos si la suma de los tres números es par: true
Comprobamos si el resto del primer número dividido por el segundo número es distinto de 0: true
Comprobamos si la multiplicación del primer número por la mitad del segundo número es mayor que el tercer nú
Comprobamos si el resultado del cuadrado del primer número menos el triple del segundo número por el tercer

Fin del programa.
```

Y otro ejemplo más:

```
Ejercicio 2. Operadores aritméticos
-----------------------------------------------------------
Introduce el primer número: 5
Introduce el segundo número: 2
Introduce el tercer número: 2

RESULTADOS
----------
Comprobamos el valor de la suma de dos primeros números dividida por el tercer número: 3.5
Comprobamos si la suma de los tres números es par: false
Comprobamos si el resto del primer número dividido por el segundo número es distinto de 0: true
Comprobamos si la multiplicación del primer número por la mitad del segundo número es mayor que el tercer nú
Comprobamos si el resultado del cuadrado del primer número menos el triple del segundo número por el tercer

Fin del programa.
```

1.4.- Ejercicio 3: recluido en casa.

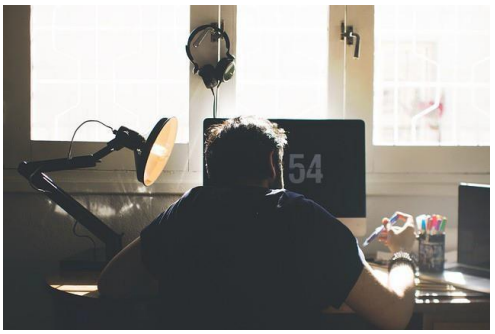
Diseñar un **algoritmo** que nos indique **si podemos salir a la calle o no**. Existen distintos aspectos que influyen en esta decisión: si está **lloviendo** o no, si hemos **terminado la tarea online** de programación o si **debemos ir al supermercado**.

Solo **podremos salir** a la calle si **no está lloviendo** y **ya hemos terminado la tarea** online de programación.

A pesar de lo anterior, existe una opción en la que **podremos salir siempre** a la calle: el hecho de que **tengamos que ir al supermercado** (necesitamos comer).

Realiza un programa en Java que **pregunte al usuario** (mediante valores booleanos) si está **lloviendo**, si ya ha terminado la **tarea** online de programación y si necesita ir al **supermercado**.

El programa decidirá si podemos salir a la calle o no y lo indicará mediante un mensaje (debes conseguir que se muestre el texto **SÍ** o **NO** según el valor booleano de la variable salirCalle sea true o false respectivamente).



Nota: Cuando pidas valores booleanos por teclado al usuario, éste solo podrá introducir **true** o **false** para que se acepte correctamente. En caso contrario dará un error.

Ejemplos de ejecución

Aquí tienes algunos ejemplos de ejecución.

Mostrar retroalimentación

Un ejemplo de ejecución del programa podría ser:

```
Ejercicio 3. Recluido en casa
.....
Indica si está lloviendo (true/false): false
Indica si has terminado la tarea online de Programación (true/false): true
Indica si tienes que ir al supermercado (true/false): false

RESULTADO
.....
Considerando la información anterior, ¿debo salir a la calle?
SI

Fin del programa.
```

Aquí tienes otro ejemplo de ejecución:

```
Ejercicio 3. Recluido en casa
.....
Indica si está lloviendo (true/false): true
Indica si has terminado la tarea online de Programación (true/false): true
Indica si tienes que ir al supermercado (true/false): false

RESULTADO
.....
Considerando la información anterior, ¿debo salir a la calle?
NO

Fin del programa.
```

Y otro más:

```
Ejercicio 3. Recluido en casa
.....................................................
Indica si está lloviendo (true/false): true
Indica si has terminado la tarea online de Programación (true/false): false
Indica si tienes que ir al supermercado (true/false): true

RESULTADO
.................
Considerando la información anterior, ¿podré salir a la calle?
SI

Fin del programa.
```

1.5.- Ejercicio 4: horas, minutos y segundos.

Escribe un programa en Java que permita **introducir por teclado** una cantidad de **segundos** mediante un **valor entero**.

A continuación, se pedirá por pantalla la introducción de **tres valores enteros** correspondientes a un número de **horas**, un número de **minutos** y un número de **segundos**.

La aplicación deberá calcular **cuántas horas, minutos y segundos** corresponden a la suma de ambas cantidades y la mostrará por pantalla.

Licencia: (Pixabay License)

Recomendación

La forma más sencilla para resolver este ejercicio es realizar **la suma de ambas cantidades en segundos** para, posteriormente, calcular el número de horas, de minutos y de segundos utilizando el **cociente** (operador `/`) y el **resto** (operador módulo `%`) de la **división entera**. El primero servirá para calcular, por ejemplo, cuántas horas completas hay en una cantidad de segundos y el segundo para calcular cuántos segundos quedan tras descontar el número de horas completas obtenida.

Recuerda que 1 hora tiene 3600 segundos y 1 minuto contiene 60 segundos

Ejemplos de ejecución

Aquí tienes un par de ejemplos de ejecución:

Mostrar retroalimentación

Un ejemplo de ejecución del programa podría ser:

```
Ejercicio 4. Horas, minutos y segundos
.....
Introduce el total de segundos: 45
Introduce el número de horas: 2
Introduce el número de minutos: 1
Introduce el número de segundos: 20

RESULTADO
.....
Las cantidades introducidas suman un total de 2 horas, 2 minutos y 5 segundos.

Fin del programa.
```

Otro ejemplo de ejecución podría ser:

```
Ejercicio 4. Horas, minutos y segundos
.....
Introduce el total de segundos: 754
Introduce el número de horas: 3
Introduce el número de minutos: 1
Introduce el número de segundos: 45

RESULTADO
.....
Las cantidades introducidas suman un total de 3 horas, 14 minutos y 19 segundos.

Fin del programa.
```

Y otro más:

```
Ejercicio 4. Horas, minutos y segundos
.....................................................................
Introduce el total de segundos: 4198
Introduce el número de horas: 5
Introduce el número de minutos: 24
Introduce el número de segundos: 53
```

1.6.- Ejercicio 5: venta de pienso para animales.

Una tienda se dedica a la **venta de pienso** para animales. A todos sus clientes les aplica un **sistema de descuentos** en función de la cantidad de sacos de pienso comprados. Los descuentos se aplican de la siguiente forma:

- si compran **más de 5 sacos** se les aplica un **5.5%** de descuento.
- si compran **más de 8 sacos** se les aplicará un **10.3%** de descuento.

Antes de calcular el importe final, se deberá sumar el **IVA**, que corresponde a un **10%**.

Licencia: (Pixabay License)

Realiza un programa en Java que permita al usuario introducir el número de sacos que desea comprar un cliente. El **precio unitario** de cada uno de los sacos de pienso es de **9,75€**.

El programa mostrará por pantalla el **número de sacos** que comprará el cliente, el **descuento que se aplicacará**, el importe total aplicando los descuentos pero no el IVA y el **importe total** de la compra tras aplicar los descuentos por cantidad que correspondan y el IVA indicados anteriormente. Por último, el sistema mostrará la **cantidad final que deberá pagar el cliente** la cual será la parte entera del importe total calculado anteriormente (por ejemplo, si el importe total fuera 26,40€ la cantidad final que debería pagar el cliente sería 26€).

Importante: Debes mostrar las cantidades decimales de forma que **muestren únicamente 2 decimales**. Para ello, puedes hacer uso del método **printf** (busca información en internet sobre cómo usarlo).

Recomendación

Puedes obtener la parte entera de un número realizando una operación de casting (los métodos matemáticos de redondeo aún no se han estudiado, por lo que no podrán utilizarse para resolver este problema).

Ejemplo de ejecución

Aquí tienes algunos ejemplos de ejecución.

Mostrar retroalimentación

Un ejemplo de ejecución del programa podríaser:

```
Ejercicio 5. Venta de pienso para animales
.....
Introduce el número de sacos que quiere comprar el cliente: 5

RESULTADO
.....
El cliente va a comprar 5 sacos
Se aplica un descuento del 0.0%
El importe aplicando el descuento (sin IVA) es de 48,75€
El importe total (IVA incl.) asciende a: 53,63€
El importe final a pagar (IVA incl.) asciende a: 53€
```

Y aquí tienes otro ejemplo:

```
Ejercicio 5. Venta de pienso para animales
.....................................................................
Introduce el número de sacos que quiere comprar el cliente: 7

RESULTADO
.................
El cliente va a comprar 7 sacos
Se aplica un descuento del 5.5%
El importe aplicando el descuento (sin IVA) es de 64,50€
El importe total (IVA incl.) asciende a: 70,95€
El importe final a pagar (IVA incl.) asciende a: 70€
```

Y otro más:

```
RESULTADO
.................
El cliente va a comprar 10 sacos
Se aplica un descuento del 10.3%
El importe aplicando el descuento (sin IVA) es de 87,46€
El importe total (IVA incl.) asciende a: 96,20€
El importe final a pagar (IVA incl.) asciende a: 96€
```

2.- Información de interés.

Recursos necesarios y recomendaciones

- Para escribir los programas en Java que resuelven los ejercicios de esta unidad tendrás que utilizar el entorno **Netbeans**.
- En los últimos apartados de la unidad, se te recomendó que utilizaras la **plantilla de programa** propuesta en el apartado **7.5.1** de la unidad. De esta manera dispondrás de una **forma sistemática de estructurar tu código** (declaración de **variables y constantes**, **entrada** de datos, **procesamiento** y **salida** de resultados). Probablemente al principio no entiendas una buena parte de lo que hay en esa plantilla de programa, pero poco a poco irás descubriendo lo que significa realmente cada elemento. Por ahora es suficiente con que sepas que es una manera sencilla de escribir un programa básico en Java.
- Para que te acostumbres a usar esa plantilla, se te proporciona un **proyecto base sobre el que debes realizar tus programas**. Contendrá un archivo Java con un ejercicio de ejemplo el cual lleva ya la plantilla integrada. Puedes descargar el proyecto desde el siguiente enlace:

mohamed Hassan (Licencia Pixabay)

Proyecto base (zip - 18.2 KB) . **Debes utilizar este proyecto base para realizar tu tarea**

- No olvides revisar la gran cantidad de **ejercicios resueltos** que se incluyen en la unidad, tanto dentro de los apartados como en un **anexo específico de ejercicios resueltos**. En algunos de ellos es posible que encuentres la clave para resolver los que están propuestos en esta tarea.
- Te recomendamos que no intentes abordar cada ejercicio de la tarea hasta que hayas repasado los contenidos necesarios para resolver ese ejercicio, y hayas trabajado con algunos de los ejercicios resueltos de forma que hayas tenido ocasión de consultar y resolver en los foros cualquier duda que te haya podido surgir.
- **Uso de los casos de prueba**. ¡Muy importante! Tened muy en cuenta lo que se indica acerca de las pruebas de vuestros programas en los siguientes apartados.
- Los comienzos son duros para casi todo el mundo, pero principalmente para quien se enfrente a la programación por primera vez. Si es tu caso, aunque los problemas iniciales tienen un nivel de dificultad escaso, es quizás lo que más trabajo cuesta, así que ¡¡prohibido desanimarse!!, hay que insistir, insistir, e insistir, y verás cómo según avance el curso, problemas mucho más complejos los resolverás con menos esfuerzo.
- **Debes incluir comentarios útiles en el código**. Os será de utilidad en el futuro si tenéis que revisar ese código para llevar a cabo algún tipo de modificación.
- Recuerda que **todos los ejercicios de esta tarea deben formar parte de un mismo y único proyecto** hecho con el **IDE NetBeans (no con ningún otro)**. En este caso, a partir del ejemplo que se te proporciona en el proyecto base debes crear un nuevo fichero java (con una clase main) para cada uno de los ejercicios solicitados en la tarea.

Indicaciones de entrega

Una vez realizada 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_PROG_Tarea01

2.1.- Estructura de los programas.

En el futuro es probable que necesitemos modificar parte del código de una aplicación por alguna de estas dos razones:

1. porque hay que **corregir algún error** que se ha detectado;
2. porque se quiere **ampliar la funcionalidad** el programa.

Esto es conocido como **mantenimiento de la aplicación**.

Free-Photos (Pixabay License)

Para que un programa sea fácil de mantener es fundamental que el código sea legible, es decir, fácil de entender y de seguir aunque no seamos quiénes lo escribieron inicialmente. Para ello es fundamental que todos sigamos ciertas normas, algunas de las cuales ya hemos visto. Entre ellas se encuentran:

- dotar a los programas de cierta **estructura homogénea**, para que de esa manera todos los programas tengan un aspecto similar y sepamos dónde ir a buscar lo que necesitamos;
- nombrar a las variables con **identificadores de variables descriptivos y representativos** de la información que almacenan (no llamar a las variables `a`, `b`, `c`, o bien `x`, `y`, `z`, o bien `e1`, `e2`, `e3`, etc.;
- usar las **convenciones de nombrado de Java** (y en general del lenguaje que se esté utilizando) según el tipo de identificador (si es variable, si es constante, etc.);
- emplear adecuadamente la **indentación** o sangrado (o tabulación) para que los distintos bloques de código y las estructuras de control queden visualmente reconocibles de un golpe de vista. **Recuerda que NetBeans puede hacer esto por ti** (selecciona el código a "*formatear*" o "*embellecer*" y pulsa `Alt+Mayús.+F`).

Todas estas directrices, más algunas otras que también hemos visto y otras que iremos viendo según vayamos avanzando, están para ser cumplidas y poder dotar a cualquier programa hecho por nosotros de **claridad**, **limpieza** y **uniformidad**. Eso permitirá a cualquier otro programador poder trabajar con ese código sin perderse ni liarse con un código enrevesado, de estructura irregular, con variables con nombres imposibles, tabulaciones que llevan al error, etc. Nuestra misión, además de escribir código que funcione y cumpla con lo que se pide, es redactar **código limpio, claro y fácil de entender**, es decir, un **código legible**.

Para que todos nuestros programas sean fáciles de seguir y de entender vamos a adoptar la siguiente estructura de manera "corporativa", es decir, que vamos a trabajar como si fuéramos una organización. Vosotros por tanto, como miembros de esa organización, deberéis seguir esa estructura. De ese modo cualquier otro miembro que revise vuestro código se sentirá cómodo con él porque podrá seguirlo con facilidad, ya que sigue los mismos estándares de organización, limpieza y claridad que cualquier otro miembro de la organización.

Estructura genérica de un programa en Java

La estructura que debemos seguir obligatoriamente es la siguiente:

Mostrar retroalimentación

```java
/*
 * Plantilla para programas de prueba
 */
import java.util.Scanner;

public class NombreProgramaJava {

    public static void main(String[] args) {

        //------------------------------------------------------------
        //          Declaración de variables
        //------------------------------------------------------------

        // Constantes


        // Variables de entrada


        // Variables de salida
```

```
        // Variables auxiliares


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


        //------------------------------------------------------------
        //                 Entrada de datos
        //------------------------------------------------------------
        System.out.println("PLANTILLA DE PROGRAMA ");
        System.out.println("---------------------- ");
        System.out.println(" ");


        //------------------------------------------------------------
        //                  Procesamiento
        //------------------------------------------------------------




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


        System.out.println ();
        System.out.println ("Fin del programa.");
    }
}
```

Como podéis observar es la que ya vimos en el apartado 7.5.1. ("*Estandarización del código*") de la **unidad 1** y que recomendábamos usar para resolver los ejercicios propuestos. Para las tareas, **no se trata de una recomendación sino de algo obligatorio** para que todos tengamos la misma estructura de programa. Eso significará que:

1. lo primero que debemos hacer es **declarar todas las variables** (y/o constantes) que vayamos a necesitar en nuestro programa (bloque "*declaración de variables*") procurando, en la medida de lo posible, clasificarlas en variables de entrada, auxiliares (o intermedias) y de salida. Procurad usar los **tipos adecuados** y asignar **nombres razonables** que ayuden a entender lo que se guarda en cada variable o constante.
2. a continuación solicitaremos los **datos de entrada** al usuario (bloque "*entrada de datos*") **por teclado** (y en el futuro puede que por otros medios), usando **mensajes claros y bien escritos respecto a lo que le estamos pidiendo**;
3. tras eso llevaremos a cabo todos los **cálculos y procesos que sean necesarios para resolver el problema** que se nos ha propuesto en el ejercicio (bloque "*procesamiento*"). Partiendo de los valores albergados en las **variables de entrada** para obtener los resultados que se nos piden. Si es necesario calcular ciertos **resultados intermedios** podrás almacenarlos en las **variables auxiliares** que declaraste en la primera parte de tu programa (declaración de variables) mientras que los r**esultados finales** podrás guardarlos en las variables que hayas definido como **variables de salida**. En este bloque no deberían declararse variables (eso se hace en el bloque de declaración) salvo que se trate de variables muy específicas como contadores para bucles, pequeños acumuladores y cosas así. El resto de variables intermedias o auxiliares (así como las de entrada y las de salida) se declararán en el bloque inicial de declaración de variables. Así, nada más ver el programa tendremos una idea global de los datos que se van a necesitar, los cálculos que se van a realizar y los resultados que se van a proporcionar, especialmente si hemos asignado **nombres descriptivos y representativos** a todas las variables;
4. por último deberás **proporcionar por pantalla** (y en el futuro puede que por otros medios) **los resultados obtenidos** siguiendo el **formato y la apariencia que se nos haya solicitado** en el enunciado de cada ejercicio. Nuevamente, para mostrar esta información habrá que usar **mensajes claros y bien escritos** (evitando erratas, faltas de ortografía, incoherencias, etc.).

En principio no deberíais tener problema pues en el proyecto base que se os proporciona ya tenéis esa plantilla en el ejercicio 0 de ejemplo (debéis crear ficheros similares para el resto de ejercicios).

No seguir esta estructura en algún ejercicio implicará una penalización en la calificación de ese ejercicio, pues no estaremos cumpliendo las normas de nuestra organización. Y **para poder trabajar bien en equipo lo primero que debemos hacer es seguir las directrices de normalización** de modo que los demás puedan entender con facilidad el código que hemos escrito nosotros y viceversa.

2.2.- Uso de los casos de prueba.

En todos los ejercicios se te proporcionan una serie de **casos de prueba** o **ejemplos de ejecución**, bien como **salidas de pantalla** donde se muestra cómo debería comportarse el programa ante determinadas entradas, bien como **tablas** en las que se resume ese comportamiento para muchas posibles entradas.

¡Esos casos de prueba son para usarlos! No tiene ningún sentido que entreguéis ejercicios que fallen con esos casos de prueba porque eso significa que no os habéis ni molestado en probarlos. Normalmente os proporcionamos un conjunto con **las mínimas pruebas que debe hacer un desarrollador con sus aplicaciones**. Eso no significa que sean todas las pruebas posibles que se deban hacer (eso se estudia específicamente en un módulo llamado *Entornos de Desarrollo*) pero sí son al menos **lo mínimo que debería probarse para asegurarnos de que nuestro programa no va a fallar en lo más básico**. El profesorado, al corregir tu tarea, va a probar como mínimo eso.

No deberíamos dar por finalizado un ejercicio hasta que no nos aseguremos de que como mínimo tiene el comportamiento esperado ante las entradas que se proporcionan como casos de prueba en el enunciado. En consecuencia, no deberíamos entregar una tarea hasta que garanticemos que se cumplen al menos los casos de prueba de todas sus partes o ejercicios.

Solo es comprensible que alguno de nuestros programas falle con esos casos de prueba si nos encontramos ya a final del plazo de entrega y no tenemos más remedio que subir lo que llevamos de tarea, sabiendo que contiene errores, pero que ya no nos da tiempo a terminarla para poder corregirlos. Es posible que en ocasiones incluso haya algunos ejercicios o parte de la tarea que no nos ha dado tiempo a finalizar antes del plazo de entrega. **Pero esa debe ser la excepción y no la norma**.

Mientras tanto, si tenéis tiempo, debéis procurar hacer que vuestros programas funcionen correctamente de acuerdo a las directrices que se han marcado en cada uno de los enunciados. Un programa que hace algo diferente a lo que se ha pedido es un programa que no le servirá al cliente que nos lo ha encargado y por tanto es probable que no cobremos por nuestro trabajo.

Recomendación

Es recomendable que trates siempre de que la salida mostrada por tus programas sea **lo más parecida posible** (en cuando a formato de salida, textos, resultados obtenidos, etc.) a la salida de los **casos de prueba** propuestos.

Para echaros una mano en ese objetivo de lograr que vuestros programas funcionen correctamente disponéis del **foro** donde siempre podéis pedir **ayuda**, así como del **correo** de vuestros profesores y profesoras, que estarán dispuestos a resolver vuestras dudas tanto sobre los contenidos como sobre aspectos específicos de la tarea que no entendéis o que no tenéis claro. Podéis poneros en contacto con ellos a través del **correo** de la plataforma o incluso **telefónicamente** si fuera necesario.

2.3.- Consejos para la realización de los ejercicios.

¿Qué debes revisar en los ejercicios de la tarea?

A continuación, te ofrecemos una orientación de los principales aspectos que debe contemplar tu solución propuesta para cada uno de los ejercicios. Es importante que antes de entregar tu tarea compruebes que tus ejercicios cumplen lo mejor posible esta lista de requisitos:

mohamed Hassan (Licencia Pixabay)

- Cualquier funcionalidad pedida en el enunciado debe poderse probar y ha de funcionar correctamente de acuerdo a las especificaciones del enunciado. **El programa debe compilar y poder ejecutarse**.
- Se tendrá en cuenta que todos los **identificadores** cumplan con el convenio sobre "**asignación de nombres a identificadores**" establecido para el lenguaje Java, para constantes, variables, métodos, clases, etc. Además, los identificadores deben tener **nombres significativos** que representen de alguna manera la información que están almacenando para que el código quede lo más claro, legible y autodocumentado posible.
- Debe seguirse la **estructura** propuesta de **declaración de variables**, **entrada de datos**, **procesamiento** y **salida de resultados**, tal y como se indica en el enunciado y en la plantilla que se proporciona.
- El **código de los programas debe estar apropiadamente comentado** para mejorar su legibilidad y mantenibilidad.
- Se debe mostrar la **información esperada y correcta por pantalla** en un **formato apropiado**.
- Debe observarse la **corrección ortográfica y gramatical**, así como la **coherencia en las expresiones lingüísticas**, tanto en los **comentarios** en el código como en los **textos de los mensajes** que aparezcan en pantalla para pedir información de entrada al usuario o para mostrar resultados de salida. Deben evitarse **mensajes de entrada de datos** y/o **salida de resultados inapropiados, descontextualizados, insuficientes o incorrectos**.
- El **código debe estar apropiadamente indentado**. Es fundamental para poder observar e intuir rápidamente, y de forma visual, la estructura de los programas. Recuerda que NetBeans puede hacer esto por ti (selecciona el código a "*formatear*" o "*embellecer*" y pulsa **Alt+Mayús.+F**).
- Se declaran las **variables** necesarias con el **tipo adecuado** y con **nombres apropiados**.
- Se realizan las **operaciones** correcta y apropiadamente usando los **operadores adecuados**.
- El código debe contener **comentarios** apropiados.

3.- Evaluación de la tarea.

Criterios de evaluación implicados

Del **RA1** (*Reconoce la estructura de un programa informático, identificando y relacionando los elementos propios del lenguaje de programación utilizado*):

a. Se han identificado los bloques que componen la estructura de un programa informático.
b. Se han creado proyectos de desarrollo de aplicaciones.
c. Se han utilizado entornos integrados de desarrollo.
d. Se han identificado los distintos tipos de variables y la utilidad específica de cada uno
e. Se ha modificado el código de un programa para crear y utilizar variables.
f. Se han creado y utilizado constantes y literales.
g. Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje.
h. Se ha comprobado el funcionamiento de las conversiones de tipos explícitas e implícitas.
i. Se han introducido comentarios en el código.

Peggy_Marco (Pixabay License)

¿Cómo valoramos y puntuamos tu tarea?

En esta tabla puedes ver los puntos de control que se van a evaluar en esta tarea, dichos puntos de control se evaluarán en el conjunto de todos los ejercicios, teniendo en cuenta los criterios de evaluación trabajados en cada uno.

Rúbrica de la tarea	
Punto de control 1: Conoce las estructuras de bloques de un programa. Ha creado un proyecto con todos los ejercicios. Utiliza IDE. ✓ CE a) Se han identificado los bloques que componen la estructura de un programa informático. ✓ CE b) Se han creado proyectos de desarrollo de aplicaciones. ✓ CE c) Se han utilizado entornos integrados de desarrollo.	7,14%
Punto de control 2: Comenta los ejercicios con distintos tipos de comentarios. ✓ CE i) Se han introducido comentarios en el código.	7,14%
Punto de Control 3: Creación y uso adecuado de variables (tipos, identificadores, y uso). ✓ CE d) Se han identificado los distintos tipos de variables y la utilidad específica de cada uno. ✓ CE e) Se ha modificado el código de un programa para crear y utilizar variables.	14,29%
Punto de control 4: Define y utiliza correctamente constantes y literales. ✓ CE f) Se han creado y utilizado constantes y literales.	7,14%

Punto de control 5: Utiliza correctamente los operadores. ✓ CE g) Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje.	7,14%
Punto de control 6: Realiza correctamente la conversión de tipos de datos. ✓ CE h) Se ha comprobado el funcionamiento de las conversiones de tipo explícitas e implícitas.	7,14%
Punto de control 7: La aplicación funciona correctamente de acuerdo a las especificaciones requeridas en el enunciado. ✓ CE a) Se han identificado los bloques que componen la estructura de un programa informático. ✓ CE b) Se han creado proyectos de desarrollo de aplicaciones. ✓ CE c) Se han utilizado entornos integrados de desarrollo. ✓ CE d) Se han identificado los distintos tipos de variables y la utilidad específica de cada uno. ✓ CE e) Se ha modificado el código de un programa para crear y utilizar variables. ✓ CE f) Se han creado y utilizado constantes y literales. ✓ CE g) Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje. ✓ CE h) Se ha comprobado el funcionamiento de las conversiones de tipo explícitas e implícitas. ✓ CE i) Se han introducido comentarios en el código.	