

# 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.

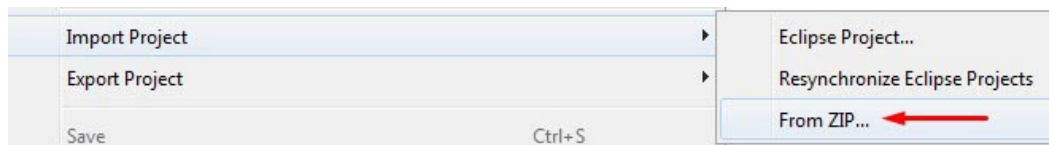
**Los ejercicios** se incluirán cada uno en una clase, con su método `main`, en un **único proyecto** NetBeans para facilitar la corrección. La plantilla de ese proyecto base se os proporciona en la sección información de interés.

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.

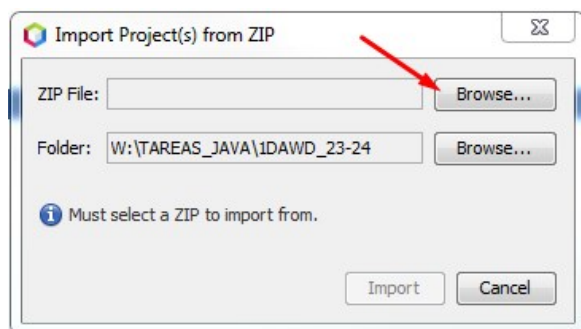
# 1.1.- Ejercicio 0: crear los paquetes sobre el proyecto base.

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.

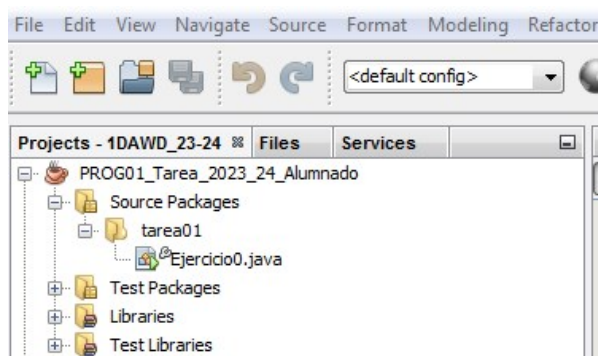
Podéis cargar este proyecto en NetBeans a través del menú **File > Import Project > From ZIP**



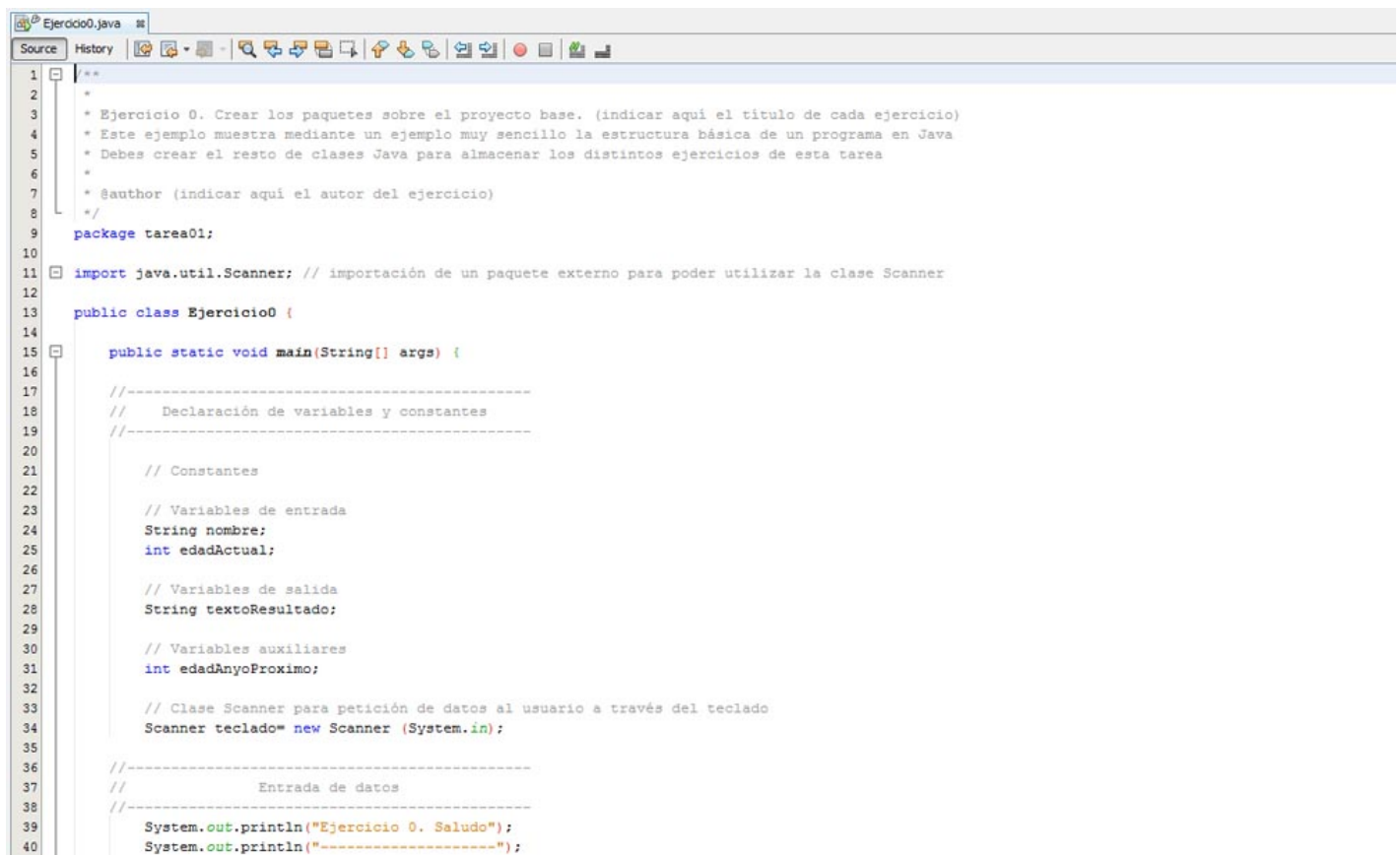
En la siguiente ventana debéis pulsar **Browse...** para localizar el fichero ZIP del proyecto base que habréis descargado (el fichero se llama **PROG01\_Tarea\_2023\_24\_Alumnado.zip**). Además, podéis usar el botón **Browse...** de abajo para elegir la carpeta donde colocar vuestros proyectos Java.



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**).



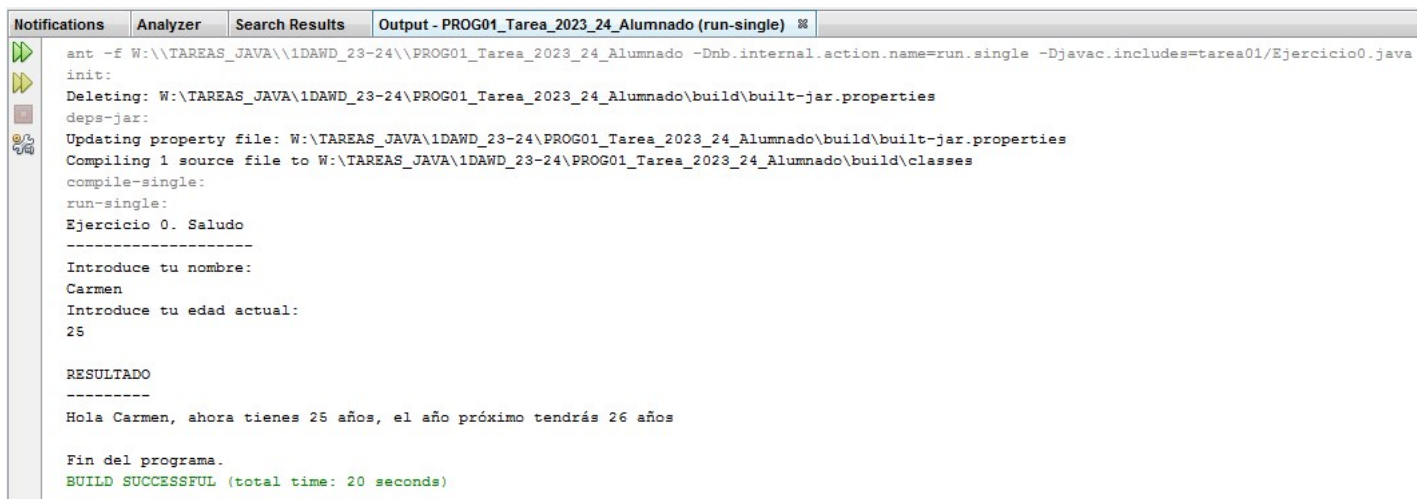
En la imagen puedes ver el contenido inicial del proyecto base que se os ha proporcionado. Este proyecto se llama **PROG01\_Tarea\_2023\_24\_Alumnado**, contiene un *paquete* llamado **tarea01** y una clase Java llamada **Ejercicio0.java**. Esta clase es un programa ejecutable (si te fijas, su icono 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).



```
1  /**
2   *
3   * Ejercicio 0. Crear los paquetes sobre el proyecto base. (indicar aqui el titulo de cada ejercicio)
4   * Este ejemplo muestra mediante un ejemplo muy sencillo la estructura básica de un programa en Java
5   * Debes crear el resto de clases Java para almacenar los distintos ejercicios de esta tarea
6   *
7   * @author (indicar aqui el autor del ejercicio)
8   */
9  package tarea01;
10
11  import java.util.Scanner; // importación de un paquete externo para poder utilizar la clase Scanner
12
13  public class Ejercicio0 {
14
15      public static void main(String[] args) {
16
17          //-----
18          //  Declaración de variables y constantes
19          //-----
20
21          // Constantes
22
23          // Variables de entrada
24          String nombre;
25          int edadActual;
26
27          // Variables de salida
28          String textoResultado;
29
30          // Variables auxiliares
31          int edadAñoProximo;
32
33          // Clase Scanner para petición de datos al usuario a través del teclado
34          Scanner teclado= new Scanner (System.in);
35
36          //-----
37          //  Entrada de datos
38          //-----
39          System.out.println("Ejercicio 0. Saludo");
40          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:

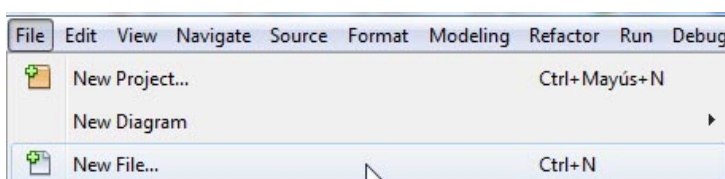


```
ant -f W:\TAREAS_JAVA\1DAWD_23-24\PROG01_Tarea_2023_24_Alumnado -Dnb.internal.action.name=run.single -Djavac.includes=tarea01/Ejercicio0.java
init:
Deleting: W:\TAREAS_JAVA\1DAWD_23-24\PROG01_Tarea_2023_24_Alumnado\build\build-jar.properties
deps-jar:
Updating property file: W:\TAREAS_JAVA\1DAWD_23-24\PROG01_Tarea_2023_24_Alumnado\build\build-jar.properties
Compiling 1 source file to W:\TAREAS_JAVA\1DAWD_23-24\PROG01_Tarea_2023_24_Alumnado\build\classes
compile-single:
run-single:
Ejercicio 0. Saludo
-----
Introduce tu nombre:
Carmen
Introduce tu edad actual:
25

RESULTADO
-----
Hola Carmen, ahora tienes 25 años, el año próximo tendrás 26 años

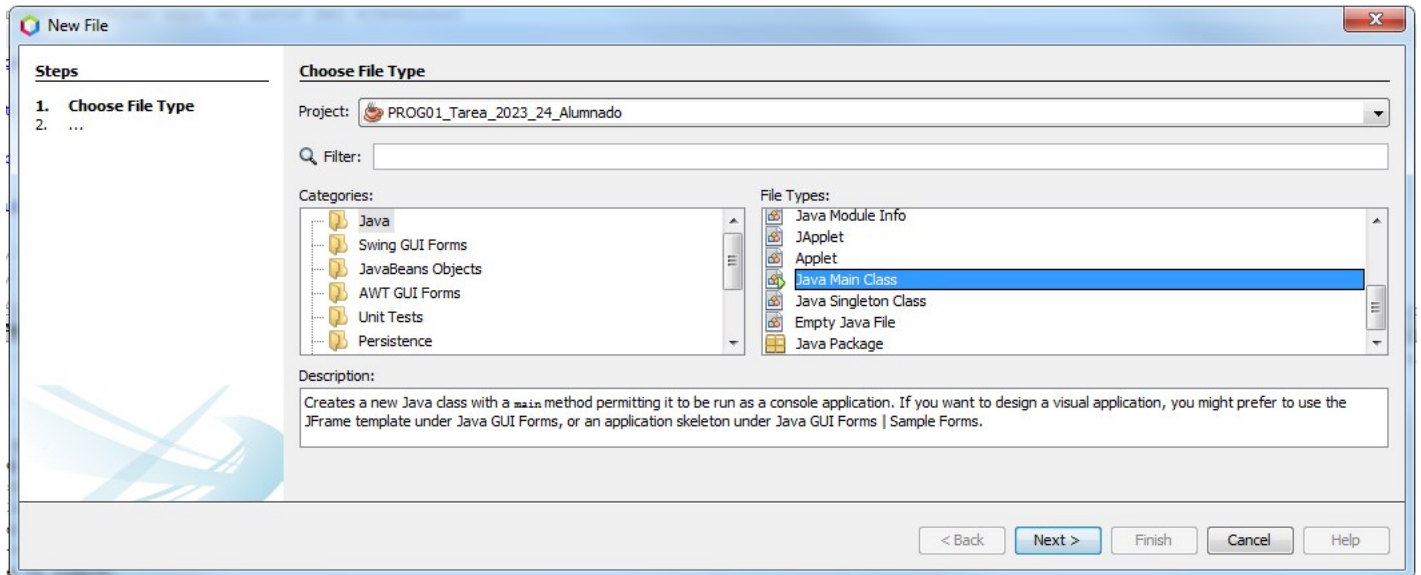
Fin del programa.
BUILD SUCCESSFUL (total time: 20 seconds)
```

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:

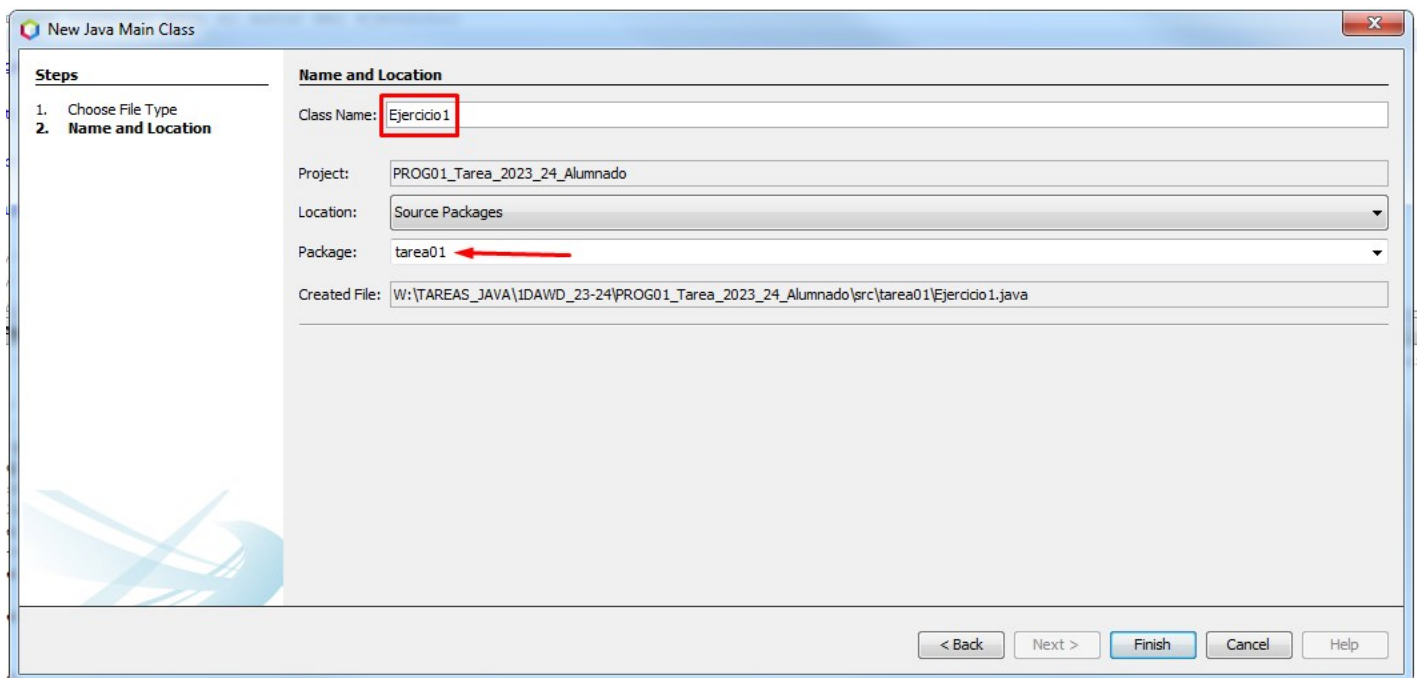


En la siguiente ventana elegiremos el tipo de clase que queremos crear, será una **Java Main Class** (será una nueva clase Java que incluirá el **método main**) ya que queremos que contenga un programa sencillo que se pueda ejecutar de forma

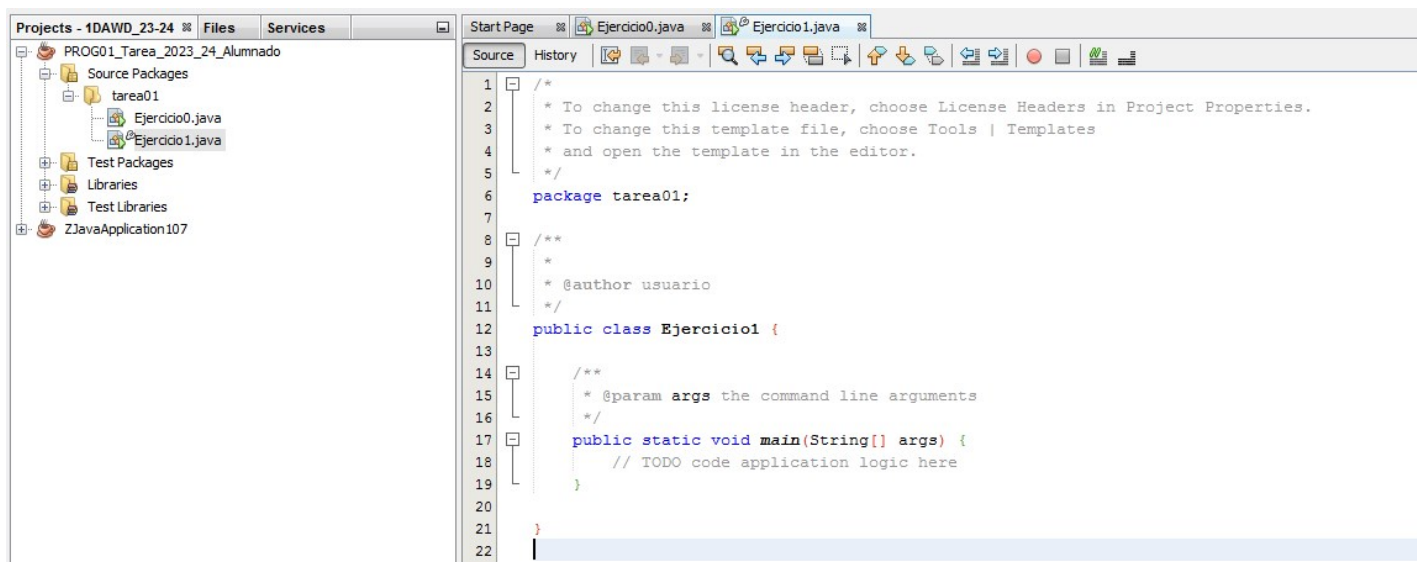
independiente:



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  package tarea01;
2
3  import java.util.Scanner;
4
5  /**
6   * Ejercicio 1. Cálculo del volumen de un cilindro
7   * @author ESCRIBE_AQUÍ_TU_NOMBRE
8   */
9  public class Ejercicio1 {
10
11     public static void main(String[] args) {
12         //-----
13         //  Declaración de variables y constantes
14         //-----
15
16         // Constantes
17
18         // Variables de entrada
19
20         // Variables de salida
21
22         // Variables auxiliares
23
24         // Clase Scanner para petición de datos al usuario a través del teclado
25         Scanner teclado= new Scanner (System.in);
26
27         //-----
28         //          Entrada de datos
29         //-----
30
31         //-----
32         //          Procesamiento
33         //-----
34
35         //-----
36         //          Salida de resultados
37         //-----
38         System.out.println();
39         System.out.println("RESULTADO");
40
41     }
42 }

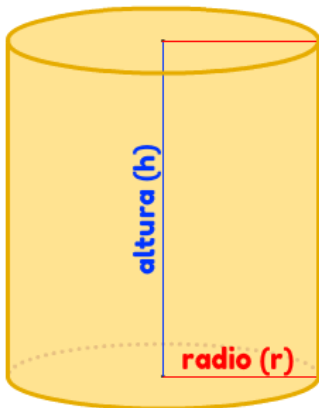
```

¡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 volumen de un cilindro.

Realiza un programa Java que permita la introducción de **dos números reales**. Estos números representarán el **radio de la circunferencia** y la **altura del cilindro**. Una vez introducidos los números por teclado, el programa calculará el **volumen de dicho cilindro** siguiendo la **fórmula** indicada y escribirá por pantalla el resultado (debes mostrar el resultado presentando únicamente dos decimales).



### Volumen del cilindro:

$$PI \times r^2 \times h$$

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.

**IMPORTANTE:** No debes utilizar métodos, estructuras o constantes predefinidas **que NO se hayan visto aún** en los contenidos de la Unidad 1 (más adelante sí podremos utilizarlos, pero en esta unidad no está permitido). Puedes definir **PI** como una **constante** con el valor **3,1415927**.

**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.

Ocultar retroalimentación

Un ejemplo de ejecución del programa podría ser:

```
Ejercicio 1. Cálculo del volumen de un cilindro
-----
Introduce el radio del cilindro: 3
Introduce la altura del cilindro: 4

RESULTADO
-----
El volumen de un cilindro de radio 3,00 y altura 4,00 es 113,10
```

Fin del programa.

Otro ejemplo de ejecución podría ser:

```
Ejercicio 1. Cálculo del volumen de un cilindro
-----
Introduce el radio del cilindro: 7
Introduce la altura del cilindro: 3,5

RESULTADO
-----
El volumen de un cilindro de radio 7,00 y altura 3,50 es 538,78

Fin del programa.
```

Y aquí tienes otro ejemplo más:

```
Ejercicio 1. Cálculo del volumen de un cilindro
-----
Introduce el radio del cilindro: 2,2
Introduce la altura del cilindro: 3,7

RESULTADO
-----
El volumen de un cilindro de radio 2,20 y altura 3,70 es 56,26

Fin del programa.
```



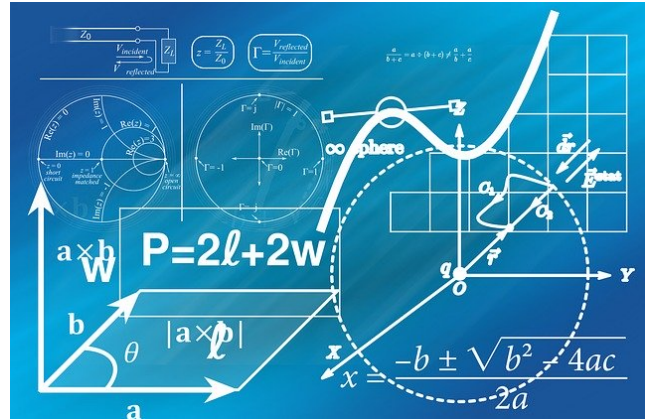
## 1.3.- Ejercicio 2: operadores aritméticos.

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

- ✓ un tercio del primer número más la mitad del tercer número.
- ✓ el cuadrado de la mitad de la suma del segundo número más el tercero.
- ✓ si el triple de la suma del segundo más el tercer número menos el primero es par.
- ✓ la suma del primero más el segundo, multiplicado por la diferencia del tercero menos el primero y todo ello partido por el segundo número.

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).



Licencia: [Pixabay License](#)



### Recomendación

Recuerda que, en Java, si en una división el numerador y el denominador son valores enteros el resultado de la operación será otro valor entero (sin decimales). Así, por ejemplo, la división del valor 7 entre el valor 2 dará un resultado 3 que corresponde al cociente entero de dicha división. Si queremos obtener el resultado exacto de la división (con todos sus decimales) será necesario aplicar **operaciones de conversión de tipos**. Para obtener más información consulta el **apartado 12** en los contenidos de la Unidad.

**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, simplemente usa otros números (más adelante nos ocuparemos de eso).



### Ejemplos de ejecución

Aquí tienes algunos ejemplos de ejecución.

Ocultar retroalimentación

Un ejemplo de ejecución del programa podría ser:

Ejercicio 2. Operadores aritméticos

```
-----  
Introduce el valor del primer número: 2  
Introduce el valor del segundo número: 5  
Introduce el valor del tercer número: 3
```

#### RESULTADO

-----

Valor del tercio del primer número más la mitad del tercer número: 2.1666666666666665

Valor del cuadrado de la mitad de la suma del segundo número más el tercero: 16.0

Comprobamos si el triple de la suma del segundo más el tercer número menos el primero es par: true

Valor de la suma del primero más el segundo, multiplicado por la diferencia del tercero menos el primero y t

Fin del programa.

Otro ejemplo de ejecución podría ser:

#### Ejercicio 2. Operadores aritméticos

-----

Introduce el valor del primer número: 5

Introduce el valor del segundo número: 2

Introduce el valor del tercer número: 4

#### RESULTADO

-----

Valor del tercio del primer número más la mitad del tercer número: 3.6666666666666667

Valor del cuadrado de la mitad de la suma del segundo número más el tercero: 9.0

Comprobamos si el triple de la suma del segundo más el tercer número menos el primero es par: false

Valor de la suma del primero más el segundo, multiplicado por la diferencia del tercero menos el primero y t

Fin del programa.

Y otro ejemplo más:

#### Ejercicio 2. Operadores aritméticos

-----

Introduce el valor del primer número: 2

Introduce el valor del segundo número: 7

Introduce el valor del tercer número: 5

#### RESULTADO

-----

Valor del tercio del primer número más la mitad del tercer número: 3.1666666666666665

Valor del cuadrado de la mitad de la suma del segundo número más el tercero: 36.0

Comprobamos si el triple de la suma del segundo más el tercer número menos el primero es par: true

Valor de la suma del primero más el segundo, multiplicado por la diferencia del tercero menos el primero y t

Fin del programa.

## 1.4.- Ejercicio 3: palabras circulares.

Realiza un programa en Java que permita introducir por teclado **tres palabras**. Una vez introducidas las palabras, se realizarán las comprobaciones necesarias para determinar:

1. Si las dos primeras palabras introducidas tienen menos de 6 caracteres o la tercera palabra tiene más de 8 caracteres.
2. Si la segunda palabra es la de mayor longitud de las tres.
3. Si las tres palabras están encadenadas (la última letra de una palabra es igual a la primera letra de la palabra siguiente) o no.
4. Si se trata de palabras circulares (palabras encadenadas en la que la última letra de la última palabra también coincide con la primera letra de la primera palabra) o no.

Observa que las **respuestas** a cada una de las 4 cuestiones anteriores **debe ser SI o NO** (según se cumpla o no la situación descrita en cada apartado). Puedes conseguir este comportamiento apoyándote en uno de los operadores vistos en la unidad.

**IMPORTANTE:** No debes utilizar estructuras de selección (if-else) puesto que aún no se han tratado en esta Unidad (más adelante sí podremos utilizarlos, pero en esta unidad no está permitido).



Licencia: Elaboración propia

**Nota:** para la determinación de palabras encadenadas o circulares no tengas en cuenta si los caracteres están escritos en mayúsculas o minúsculas. Así, por ejemplo, las palabras Ordenador -> Ratón se considerarían palabras encadenadas aunque en la primera palabra la 'r' sea minúscula y en la segunda palabra la 'R' sea mayúscula.



### Ejemplos de ejecución

Aquí tienes algunos ejemplos de ejecución.

Ocultar retroalimentación

Un ejemplo de ejecución del programa podría ser: Algoritmo - Objetos - Secuencia

```
Ejercicio 3. Palabras encadenadas
-----
Introduce la PRIMERA palabra: algoritmo
Introduce la SEGUNDA palabra: Objetos
Introduce la TERCERA palabra: Secuencia

RESULTADO
-----
La longitud de las dos primeras palabras es menor de 6 o la longitud de la tercera es mayor de 8 caracteres
La segunda palabra es la palabra de mayor longitud : NO
Las tres palabras introducidas son palabras encadenadas: SI
Las tres palabras introducidas son palabras circulares: SI

Fin del programa.
```

Aquí tienes otro ejemplo de ejecución: Coche - Bicicleta - Patinete

### Ejercicio 3. Palabras encadenadas

-----  
Introduce la PRIMERA palabra: coche  
Introduce la SEGUNDA palabra: bicicleta  
Introduce la TERCERA palabra: PATINETE

#### RESULTADO

-----  
La longitud de las dos primeras palabras es menor de 6 o la longitud de la tercera es mayor de 8 caracteres  
La segunda palabra es la palabra de mayor longitud : SI  
Las tres palabras introducidas son palabras encadenadas: NO  
Las tres palabras introducidas son palabras circulares: NO  
  
Fin del programa.

Y otro ejemplo más: Caballo - Oveja - Abeja

### Ejercicio 3. Palabras encadenadas

-----  
Introduce la PRIMERA palabra: CABALLO  
Introduce la SEGUNDA palabra: oveja  
Introduce la TERCERA palabra: Abeja

#### RESULTADO

-----  
La longitud de las dos primeras palabras es menor de 6 o la longitud de la tercera es mayor de 8 caracteres  
La segunda palabra es la palabra de mayor longitud : NO  
Las tres palabras introducidas son palabras encadenadas: SI

## 1.5.- Ejercicio 4: gestión de recursos hídricos

Diseña un programa en Java que permita monitorizar el nivel de agua de un embalse. El embalse tiene una capacidad fija máxima de **2.000 hm<sup>3</sup>** (hectómetros cúbicos).

El usuario deberá introducir por el teclado el **volumen de agua embalsada** en un momento determinado. A partir de esa cantidad, el programa calculará la cantidad de hectómetros cúbicos de agua que son **necesarios para que el embalse se llene completamente**. Igualmente, se calculará el **porcentaje** que corresponde la cantidad de agua embalsada respecto a la capacidad total del embalse.

- Si el porcentaje actual de agua embalsada es **superior al 95% de la capacidad total** del embalse se realizará una **liberación controlada** de agua de un **10% del volumen de agua embalsada**.



Licencia: ([Imagen de Wirestock](#) en Freepik)

En este caso, se debe informar al usuario del porcentaje de liberación que se ha realizado, de cuántos hectómetros cúbicos se han vaciado en esa operación, cual es el volumen actual del depósito, y cuál es el porcentaje actual.

Por contra, si el porcentaje actual del embalse es inferior al 95% no se realizará ningún tipo de liberación. En este caso simplemente se mostrará el mensaje *"No es necesario considerar la liberación controlada de agua en este momento."*

**IMPORTANTE:** Recuerda que el uso de métodos o estructuras que **NO** se han visto aún en los contenidos de la Unidad 1 está **prohibido** (más adelante sí podremos utilizarlos, pero en esta unidad no está permitido ya que, en esta limitación, se encuentra parte de la dificultad del ejercicio).



### Recomendación

Una vez se realice la evaluación del operador ternario para la distinción de los posibles casos que se plantean en el problema te recomendamos guardar el mensaje que vaya a salir por pantalla, en una variable de tipo cadena, en lugar de mostrarla directamente. Una vez terminados todos los posibles casos se presentará por pantalla dicha cadena resultado.

**Nota:** No es necesario que controles valores incoherentes (volumen actual negativo o superior a la capacidad máxima del embalse). En las próximas unidades dispondremos de más herramientas para controlar estas situaciones.



### Ejemplos de ejecución

Aquí tienes un par de ejemplos de ejecución:

Un ejemplo de ejecución del programa podría ser: (embalse a baja capacidad, no se requiere liberación controlada)

Ejercicio 4. Gestión de recursos hídricos

Introduce el volumen actual de agua almacenada en el embalse (hectómetros cúbicos): 533

Faltan 1467,00 hectómetros cúbicos para llenar completamente el embalse.  
El embalse está a un 26,65% de su capacidad máxima.

RESULTADO

No es necesario considerar la liberación controlada de agua en este momento.

Fin del programa.

Otro ejemplo de ejecución podría ser: (embalse a más del 95% de su capacidad máxima, liberación del 10%)

Ejercicio 4. Gestión de recursos hídricos

Introduce el volumen actual de agua almacenada en el embalse (hectómetros cúbicos): 1950

Faltan 50,00 hectómetros cúbicos para llenar completamente el embalse.  
El embalse está a un 97,50% de su capacidad máxima.

RESULTADO

Se ha realizado una liberación del 10.0% vaciando un total de 195.0 hectómetros cúbicos.  
En el embalse quedan ahora 1755.0 hectómetros cúbicos, que representan un 87.75% de su capacidad máxima.

Fin del programa.



## 1.6.- Ejercicio 5: ¡vamos al parque acuático!

Escribe un programa en Java que permita calcular el importe a pagar por los clientes del **Parque Acuático AquaTrass** teniendo en cuenta los siguientes requisitos:

- ✓ existirán dos tipos de entradas:
  - **Infantil** con un precio base de **10,00 €**
  - **Adulto** con un precio base de **15,00 €**.
- ✓ en caso de que el coste total de las entradas sea **superior a 50,00 €** se aplicará automáticamente un **5% de descuento**. Igualmente, si el importe base total es **superior a 100,00 €** el descuento será de un **15%**.



Guy Percival (CC0 Public Domain)

Además, una vez realizados los descuentos pertinentes se deberá aplicar un **IVA del 21%** al importe resultante.

El programa solicitará la **cantidad de entradas de cada tipo** que se desean comprar, realizará todos los cálculos necesarios y mostrará por pantalla los siguientes resultados:

1. el **número de entradas de adulto y de entradas infantiles** que comprará el cliente.
2. el **importe total** de las entradas **antes de aplicar** (si procede) **el descuento**.
3. el **porcentaje de descuento** que se aplicará **o el texto "No procede descuento en esta compra"** en caso de que no se cumplan los requisitos para aplicar descuento
4. el **importe total aplicando el descuento (si procede) pero no el IVA**.
5. el **importe total** de la compra tras aplicar el descuento que corresponda y el IVA indicado 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 **32,67 €** la cantidad final que debería pagar el cliente sería **32 €**).

**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). Por ahora **no controlaremos que la cantidad de entradas** introducidas tenga que ser **positiva** (aunque una cantidad negativa de entradas en nuestro ejemplo no tendría sentido). Más adelante veremos cómo poder controlar estas cosas.



### 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.

Un ejemplo de ejecución del programa en el que se aplique el descuento del 15%:

Ejercicio 5. ¡Vamos al parque acuático!

-----

Introduce la cantidad de entradas DE ADULTO que deseas adquirir: 6

Introduce la cantidad de entradas INFANTILES que deseas adquirir: 5

RESULTADO

-----

Se comprarán 6 entradas de tipo ADULTO y 5 entradas de tipo INFANTIL

El coste de las entradas antes de aplicar descuentos es de 126,00 €

Se aplicará un descuento del 15.0%

Tras aplicar posibles descuentos el importe total de las entradas (sin IVA) es de 107,10 €

El importe IVA incluido es de 129,59 €

La cantidad final a pagar por el cliente es de 129 €

Otro ejemplo de ejecución en el que se aplica un descuento del 5%:

Ejercicio 5. ¡Vamos al parque acuático!

-----

Introduce la cantidad de entradas DE ADULTO que deseas adquirir: 4

Introduce la cantidad de entradas INFANTILES que deseas adquirir: 0

RESULTADO

-----

Se comprarán 4 entradas de tipo ADULTO y 0 entradas de tipo INFANTIL

El coste de las entradas antes de aplicar descuentos es de 54,00 €

Se aplicará un descuento del 5.0%

Tras aplicar posibles descuentos el importe total de las entradas (sin IVA) es de 51,30 €

El importe IVA incluido es de 62,07 €

La cantidad final a pagar por el cliente es de 62 €

Por último, otro ejemplo en el que no se pueden aplicar descuentos:

Ejercicio 5. ¡Vamos al parque acuático!

-----

Introduce la cantidad de entradas DE ADULTO que deseas adquirir: 2

Introduce la cantidad de entradas INFANTILES que deseas adquirir: 0

RESULTADO

-----

Se comprarán 2 entradas de tipo ADULTO y 0 entradas de tipo INFANTIL

El coste de las entradas antes de aplicar descuentos es de 27,00 €

No procede descuento en esta compra

Tras aplicar posibles descuentos el importe total de las entradas (sin IVA) es de 27,00 €

El importe IVA incluido es de 32,67 €

La cantidad final a pagar por el cliente es de 32 €



## 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)

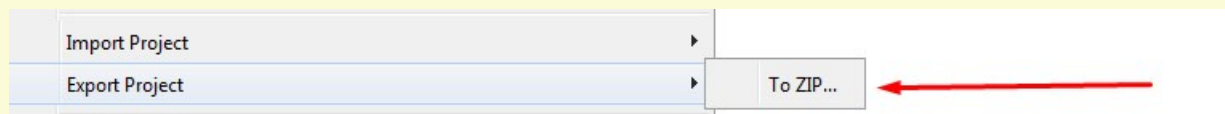
 [Proyecto base de la Tarea Online 01](#) (zip - 17,8 KB) . **Debes utilizar este proyecto base para realizar tu tarea**

- ✓ 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, genera el archivo comprimido que debes entregar a través de la plataforma. Para ello, haz clic sobre el proyecto que has realizado en el Explorador de Proyectos y, a continuación, localiza en el menú **File** de NetBeans la opción **Export Project > To ZIP** para generar un archivo comprimido (ZIP) con el contenido del proyecto.



Selecciona dónde quieres guardarlo y nombra el fichero a generar siguiendo las siguientes pautas:

**TuApellido1\_TuApellido2\_TuNombre\_PROG\_Tarea01.zip**

Por último, accede al espacio de la **Tarea Online 01** en **Moodle** y realiza la entrega subiendo el archivo ZIP generado a la plataforma (la posibilidad de entrega estará abierta durante unos periodos determinados, que debes respetar).

## 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**.

**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 quienes lo escribieron inicialmente. Para ello es fundamental que todos sigamos ciertas normas, algunas de las cuales ya hemos visto. Entre ellas se encuentran:



[Free-Photos \(Pixabay License\)](#)

- ✓ 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., sino utilizar nombres que puedan identificar su utilidad en el programa o su contenido fácilmente);
- ✓ 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:

Ocultar retroalimentación

```
/*
 * 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 **resultados 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 al final del plazo de entrega y no tenemos más remedio que subir lo que llevamos de tarea (sabiendo que contiene errores) porque ya no nos da tiempo a terminarla para poder corregirlos. Es posible que, en ocasiones, incluso haya algunos ejercicios o partes de la tarea que no nos haya 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 1** donde siempre podéis pedir **ayuda**, así como del **correo** de vuestros profesores, 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:

- ✓ 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**.



[mohamed Hassan](#) (Licencia Pixabay)

### 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	
<b>Punto de control 1: Conoce las estructuras de bloques de un programa. Ha creado un proyecto con todos los ejercicios. Utiliza IDE.</b>  ✔ 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%
<b>Punto de control 2: Comenta los ejercicios con distintos tipos de comentarios.</b>  ✔ CE i) Se han introducido comentarios en el código.	7,14%
<b>Punto de Control 3: Creación y uso adecuado de variables (tipos, identificadores, y uso).</b>  ✔ 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%

<p><b>Punto de control 4: Define y utiliza correctamente constantes y literales.</b></p> <p>✓ CE f) Se han creado y utilizado constantes y literales.</p>	7,14%
<p><b>Punto de control 5: Utiliza correctamente los operadores.</b></p> <p>✓ CE g) Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje.</p>	7,14%
<p><b>Punto de control 6: Realiza correctamente la conversión de tipos de datos.</b></p> <p>✓ CE h) Se ha comprobado el funcionamiento de las conversiones de tipo explícitas e implícitas.</p>	7,14%
<p><b>Punto de control 7: La aplicación funciona correctamente de acuerdo a las especificaciones requeridas en el enunciado.</b></p> <p>✓ CE a) Se han identificado los bloques que componen la estructura de un programa informático.</p> <p>✓ CE b) Se han creado proyectos de desarrollo de aplicaciones.</p> <p>✓ CE c) Se han utilizado entornos integrados de desarrollo.</p> <p>✓ CE d) Se han identificado los distintos tipos de variables y la utilidad específica de cada uno.</p> <p>✓ CE e) Se ha modificado el código de un programa para crear y utilizar variables.</p> <p>✓ CE f) Se han creado y utilizado constantes y literales.</p> <p>✓ CE g) Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje.</p> <p>✓ CE h) Se ha comprobado el funcionamiento de las conversiones de tipo explícitas e implícitas.</p> <p>✓ CE i) Se han introducido comentarios en el código.</p>	50%