Tarea online

Título de la tarea: Introducción a la programación.

Unidad: 01

Ciclo formativo y módulo: DAM/DAW, Programación.

Curso académico: 2025/2026

¿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?

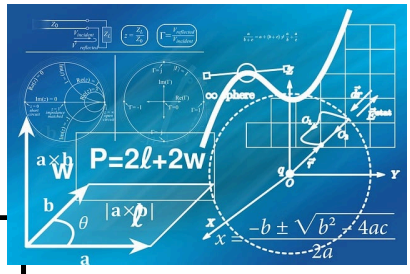
En esta tarea hay varios ejercicios para resolver e implementar como programas en **Java** usando el IDE **NetBeans**, declarando las variables necesarias con cuidado de elegir los tipos más adecuados para cada caso y de nombrarlas siguiendo la nomenclatura que se recoge en los convenios comúnmente aceptados 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 1. Expresiones algorítimicas y conversiones de tipo.

Escribe un programa en Java que convierta en **expresiones algorítmicas** las siguientes expresiones físicas y/o matemáticas. Antes, define un enumerado con las cadenas que muestren las operaciones: OPERACION, FUERZA_PESO, NUMERO_VUELTAS y AREA_CIRCULO, para posteriormente mostrar por pantalla cada valor del enumerado antes del resultado de cada una de ellas, las cuales se definen de la siguiente manera:



Licencia: Pixabay License

1. Operación matemática:

$$operación = \frac{x + \frac{x}{4.0}}{6.0 - \frac{x}{2.0}}$$

donde x es una variable (de tipo entero).

2. Fuerza peso:

$$F_{peso} = m * g$$

donde:

- m es la masa en kilogramos (de tipo entero).
- g es la constante de la gravedad con valor 9.8 (de tipo real).

3. Número de vueltas completas:

$$numVueltasCompletas = f * t$$

donde:

- f es la frecuencia en hercios (de tipo real).
- t es el tiempo en segundos (de tipo entero).

4. Área de un círculo:

$$A = \pi * r^2$$

donde:

- π es el número PI.
- r es el radio en metros (de tipo real).

Manuel Marín Peral. Expresiones físicas y/o matemáticas (CC0)

Para ello tendrás que utilizar **operadores aritméticos** tales como la suma, la resta, el producto o la división. Además, debes tener muy en cuenta el orden de **precedencia de operadores** para minimizar el uso de los **paréntesis**, **usándolos sólo cuando sea estrictamente necesario**.

Con respecto a las **conversiones de tipo**, también deberás considerar los casos de **casting implícito** y **casting explícito** entre tipos de datos, indicando mediante comentarios en código **qué tipos** de casting se produce y **de qué tipo** de dato **a qué tipo** de dato en cada una de las cuatro expresiones implementadas. Debes tener en cuenta que es posible que en el cálculo de una misma expresión se produzca más de una conversión de tipo.

El programa, para comprobar que las expresiones se han escrito de manera correcta, debe solicitar cinco parámetros y asignárselos a las variables **x**, **masa**, **tiempo**, **frecuencia** y **radio**. A continuación, tendrá que evaluar cada una de las expresiones y mostrar los resultados por pantalla.

Cabe destacar que a la hora de utilizar el **número π**, **no está permitido** recurrir al uso de librerías como por ejemplo *Math*, ya que aún no hemos estudiado el uso de librerías en el curso. Es por este motivo por el que se deberá utilizar una aproximación calculada mediante la **serie de Leibniz**, la cual se define de la siguiente manera en su **forma general**:

$$\frac{\pi}{4} = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$$

Forma general de la serie de Leibniz. (Dominio público)

En nuestro caso, haremos uso de la **aproximación** utilizando únicamente los **5 primeros términos**, la cual se obtendrá de la siguiente manera y será aquella que deberemos **implementar** en nuestro programa para obtener el valor de π (que será constante):

$$\pi \approx 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9}\right)$$

Aproximación de Leibniz 5 términos. (Dominio público)

Ejemplos de ejecución

Aquí tienes algunos ejemplos de ejecución.

Con los valores **5**, **13**, **2**, **4,8** y **1,6**.

```
CÁLCULO EXPRESIONES FÍSICAS/MATEMÁTICAS
-----------------------------------
Introduce el valor de X: 5
Introduce la masa (kg): 13
Introduce el tiempo (s): 2
Introduce la frecuencia (hz): 4,8
Introduce el radio del círculo (m): 1,6

RESULTADOS
----------
```

```
OPERACION: 1.7857142857142858
FUERZA_PESO: 127.4000015258789
NUMERO_VUELTAS: 9
AREA_CIRCULO: 8.549587906731501
```

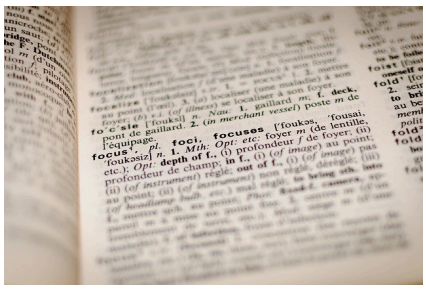
Con los valores **10**, **2, 5**, **5,5** y **2,8**.

```
CÁLCULO EXPRESIONES FÍSICAS/MATEMÁTICAS
-----------------------------------
Introduce el valor de X: 10
Introduce la masa (kg): 2
Introduce el tiempo (s): 5
Introduce la frecuencia (hz): 5,5
Introduce el radio del círculo (m): 2,8

RESULTADOS
----------
OPERACION: 12.5
FUERZA_PESO: 19.600000381469727
NUMERO_VUELTAS: 27
AREA_CIRCULO: 26.1831100282215
```

Con los valores **1**, **5**, **10**, **3,3** y **6,6**.

```
CÁLCULO EXPRESIONES FÍSICAS/MATEMÁTICAS
-----------------------------------
Introduce el valor de X: 1
Introduce la masa (kg): 5
Introduce el tiempo (s): 10
Introduce la frecuencia (hz): 3,3
Introduce el radio del círculo (m): 6,6

RESULTADOS
----------
OPERACION: 0.22727272727272727
FUERZA_PESO: 49.0
NUMERO_VUELTAS: 33
AREA_CIRCULO: 145.4765632750496
```

1.2.- Ejercicio 2. Uso de cadenas.



Free-Photos (Pixabay License)

Necesitamos un programa que nos permita realizar sobre la cadena inicial ***" Don Quijote de la Mancha "*** las operaciones y comprobaciones que se especifican a continuación:

1. Elimina los espacios al **comienzo** y al **final** de la cadena.
2. Elimina **"Don Quijote"**.
3. Sustituye el **carácter 'M'** de la palabra Mancha por el **carácter 'L'**.
4. Añade al final de la cadena un **guion** seguido del **carácter** con la **letra** del **grupo** al que perteneces **(sin utilizar métodos)**.
5. Obtén la **longitud** de la **cadena resultante** de aplicar las operaciones anteriores.
6. Obtén el **carácter** que se encuentra en la **posición central** de la **cadena resultante** de aplicar las operaciones anteriores.
7. Comprueba si la cadena **contiene** el **carácter 'M'**.
8. Comprueba si la cadena **termina** en **consonante (sin utilizar métodos)**.

Escribe un programa en Java que lea de teclado un carácter que sea la **letra** del **grupo de Programación** al que perteneces y que **aplique** sobre la **cadena inicial** propuesta todas las **operaciones** y **comprobaciones** enumeradas anteriormente.

Puedes usar el **operador condicional** `?` para informar sobre si la cadena contiene el carácter 'M' o termina en consonante. Ten en cuenta que **aún no hemos estudiado las estructuras de control condicionales** (como por ejemplo `if`) y no podemos utilizarlas.

Ejemplos de ejecución

Aquí tienes algunos ejemplos de ejecución.

Ejemplo 1. Introduciendo 'B' como grupo.

```
USO DE CADENAS DE CARACTERES
----------------------------
Introduzca la letra del grupo al que perteneces: B
Cadena original:   Don Quijote de la Mancha

RESULTADOS
----------
La cadena final es: de la Lancha-B
La cadena final contiene: 14 caracteres
La cadena final tiene el caracter: a en su posicion central
```

```
La cadena final contine el carcacter 'M' : NO
La cadena final termina en consonante : SI
```

Ejemplo 2. Introduciendo 'M' como grupo.

```
USO DE CADENAS DE CARACTERES
----------------------------
Introduzca la letra del grupo al que perteneces: M
Cadena original:   Don Quijote de la Mancha

RESULTADOS
----------
La cadena final es: de la Lancha-M
La cadena final contiene: 14 caracteres
La cadena final tiene el caracter: a en su posicion central
La cadena final contine el carcacter 'M' : SI
La cadena final termina en consonante : SI
```

Ejemplo 3. Introduciendo 'E' como grupo.

```
USO DE CADENAS DE CARACTERES
----------------------------
Introduzca la letra del grupo al que perteneces: E
Cadena original:   Don Quijote de la Mancha

RESULTADOS
----------
La cadena final es: de la Lancha-E
La cadena final contiene: 14 caracteres
La cadena final tiene el caracter: a en su posicion central
La cadena final contine el carcacter 'M' : NO
La cadena final termina en consonante : NO
```

2.- Información de interés.

Recursos necesarios y recomendaciones

mohamed Hassan (Licencia Pixabay)

- Para escribir los programas en Java que resuelven los ejercicios de esta unidad tendrás que utilizar el entorno **Netbeans 16** y **Java Development Kit 17 (JDK)**.
- 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 para cada ejercicio con la plantilla integrada (salvo en el caso del primero, que es tan simple que no vale la pena plantear esa estructura). Puedes descargar el proyecto desde el siguiente enlace: Proyecto base (zip - 24,90 KB). **Debes utilizar este proyecto base para realizar tu tarea** . Ese proyecto se llamará *PROG01_Tarea_2024_25_Alumnado*. Por favor, renómbralo en Netbeans para que se llame ***Apellido1_Apellido2_Nombre_PROG01_Tarea***. Si no, todos los proyectos que nos enviéis se llamarán igual.
- 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.
- Procura utilizar las estructuras que se te indican en cada ejercicio y **no utilices herramientas que no se te hayan pedido o aún no hayamos visto** (sentencias condicionales, bucles, etc.) pues no es eso lo que se está evaluando en esta actividad.
- 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.
- Siempre que lo consideréis oportuno, **incluid 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 Java de esta tarea formarán parte de un mismo proyecto hecho con el **IDE NetBeans 16 (no con ningún otro)**. En este caso no deberías tener problema, pues te proporcionamos el proyecto base sobre el que realizar el ejercicio.

Indicaciones de entrega

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

Apellido1_Apellido2_Nombre_PROG01_Tarea

Recuerda que **tu proyecto Netbeans también debe llamarse así** (tanto la carpeta donde se encuentra el proyecto como el nombre del proyecto en Netbeans tienes que renombrarlos respecto al nombre original del proyecto base que vas a utilizar). **Si no lo haces, todos los proyectos se llamarán igual y la corrección de tu tarea podría confundirse con la de otra persona**.

Por tanto, **si no entregas tu tarea siguiendo estas reglas, se te devolverá para que lo hagas correctamente** .

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. Procuraremos, siempre que sea posible, evitar 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 elaborado por nosotros de **claridad**, **limpieza** y **uniformidad**. Eso permitirá a cualquier otra persona poder trabajar con ese código sin perderse ni liarse con un código enrevesado, de estructura irregular, con variables con nombres extraños o contradictorios, 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:

```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.");
    }
}
```

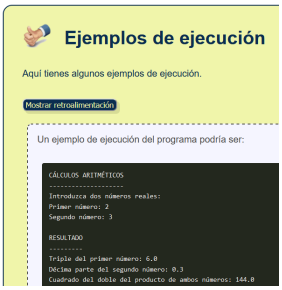
Como podéis observar es la que ya vimos en el apartado 7.5.1. ("*Estandarización del código*") de la **unidad 1** y que recomendábamos usar para resolver los ejercicios propuestos. Para las tareas, **no se trata de una recomendación sino de algo obligatorio** para que todos tengamos la misma estructura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

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, aunque no tiene por qué ser lo único que pruebe.

> **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.

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** o por **videoconferencia** 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. Eso implica, entre otras cosas, una **redacción correcta** de los textos que se incluyan en esos comentarios.
- 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**.

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.

¿Cómo valoramos y puntuamos tu tarea?

Cada uno de los CE cubiertos por esta tarea será evaluado a través de los criterios de corrección y puntuación indicados detalladamente en la rúbrica de la tarea que puedes consultar a continuación:

ÍTEMS	PUNTUACIONES			
(RA1.a,RA1.b,RA1.c) EJERCICIO 1: Se sigue la estructura propuesta de declaración de variables, entrada de datos, procesamiento y salida de resultados, tal y como se indica en la plantilla y el enunciado.	No se sigue la estructura de la plantilla **0 puntos**	Sí se sigue la estructura de la plantilla **0.75 puntos**		
(RA1.a) EJERCICIO 1: El código del programa está correctamente indentado.	No se ha indentado correctamente **0 puntos**	Se ha indentado correctamente **0.25 puntos**		
(RA1.d,RA1.e) EJERCICIO 1: Se declara y se utiliza correctamente el enumerado.	No se ha declarado el enumerado **0 puntos**	Se ha declarado el enumerado pero no se utiliza correctamente **0.4 puntos**	Se ha declarado y utilizado correctamente el enumerado **0.75 puntos**	
(RA1.d,RA1.e) EJERCICIO 1: Se declaran las variables necesarias para almacenar los valores introducidos por teclado, con el tipo adecuado.	No se han declarado variables **0 puntos**	Se han declarado variables pero no son del tipo adecuado **0.4 puntos**	Se han declarado algunas variables con el tipo adecuado **0.7 puntos**	Se han declarado todas las variables con el tipo adecuado **1 punto**
(RA1.f) EJERCICIO 1: Se han creado y utilizado correctamente constantes y literales.	No se han utilizado constantes ni literales. **0 puntos**	Se han creado y utilizado constantes y literales pero con errores. **0.4 puntos**	Se han creado y utilizado correctamente constantes y literales **0.6 puntos**	
(RA1.d) EJERCICIO 1: Se declaran identificadores con nombres descriptivos que representen la información que están almacenando. Los identificadores cumplen con el convenio sobre "asignación de nombres a identificadores" establecido para el lenguaje Java para constantes, variables, métodos, clases, etc.	Los identificadores no son descriptivos y no cumplen con el convenio **0 puntos**	Los identificadores no son descriptivos o no cumplen con el convenio **0.05 puntos**	Los identificadores son descriptivos y cumplen con el convenio **0.25 puntos**	
(RA1.g) EJERCICIO 1: Se han utilizado los operadores adecuados.	No se han utilizado los operadores adecuados en ninguna de las expresiones **0 puntos**	Se han utilizado los operadores adecuados en alguna de las expresiones **0.5 puntos**	Se han utilizado los operadores adecuados en la mayoría de las expresiones **1 punto**	Se han utilizado los operadores adecuados en todas las expresiones **1.5 puntos**

(RA1.h) EJERCICIO 1: Se han aplicado correctamente conversiones de tipo explícitas e implícitas.	No se han aplicado conversiones de tipo. **0 puntos**	Se han aplicado conversiones de tipo pero con errores. **0.2 puntos**	Sólo se han aplicado conversiones implícitas. **0.4 puntos**	Se han aplicado correctamente conversiones de tipo explícitas e implícitas. **0.8 puntos**
(RA1.a,RA1.b,RA1.c) EJERCICIO 2: Se sigue la estructura propuesta de declaración de variables, entrada de datos, procesamiento y salida de resultados, tal y como se indica en la plantilla y el enunciado.	No se sigue la estructura de la plantilla **0 puntos**		Si se sigue la estructura de la plantilla **0.75 puntos**	
(RA1.a) EJERCICIO 2: El código de los programas está correctamente indentado.	No se ha indentado correctamente **0 puntos**		Se ha indentado correctamente **0.25 puntos**	
(RA1.f) EJERCICIO 2: Se han creado y utilizado correctamente constantes y literales.	No se han utilizado constantes ni literales. **0 puntos**	Se han creado y utilizado constantes y literales pero con errores. **0.4 puntos**	Se han creado y utilizado correctamente constantes y literales **0.6 puntos**	
(RA1.e,RA1.g) EJERCICIO 2: Se eliminan correctamente las partes de la cadena inicial especificadas en la tarea (espacios al principio y final y la cadena “Don Quijote”).	No se realiza correctamente. **0 puntos**	Se eliminan solamente algunos caracteres de forma correcta. **0.25 puntos**	Se eliminan correctamente las partes de la cadena inicial especificadas. **0.5 puntos**	
(RA1.e,RA1.g) EJERCICIO 2: Se sustituye el carácter “M” por “L” y se añade al final un guión con la letra del grupo correctamente.	No se realizan las modificaciones especificadas. **0 puntos**	Se ha realizado una de las dos operaciones solicitadas. **0.25 puntos**	Se realizan correctamente las modificaciones solicitadas. **0.5 puntos**	
(RA1.e,RA1.g) EJERCICIO 2: Se obtienen la longitud y el carácter central de la cadena resultante correctamente.	No se realizan correctamente ninguna de las dos operaciones descritas. **0 puntos**	Se obtiene correctamente la longitud o el carácter central. **0.25 puntos**	Se obtienen correctamente tanto la longitud como el carácter central de la cadena resultante. **0.5 puntos**	
(RA1.e,RA1.g) EJERCICIO 2: Se comprueba si la cadena resultante contiene el carácter “M” y si termina en consonante correctamente.	No se realizan correctamente ninguna de las dos operaciones descritas. **0 puntos**	Se comprueba correctamente una de las dos operaciones solicitadas. **0.25 puntos**	Se comprueban correctamente tanto si la cadena resultante contiene el carácter “M” y si termina en consonante. **0.5 puntos**	
(RA1.i) EJERCICIOS 1 y 2: El código está apropiadamente comentado. Se observa una corrección ortográfica y gramatical, así como la coherencia en las expresiones lingüísticas en los comentarios incluidos.	El código no está comentado **0 puntos**	El código incluye comentarios pero no son suficientes para entender el funcionamiento de parte del programa **0.25 puntos**	El código está correctamente comentado **0.5 puntos**	

Tanto en la tarea en general como en cada una de las actividades que tengas que desarrollar en particular, se tendrán en cuenta los siguientes puntos de control o elementos evaluables a la hora de corregir tu tarea:

- Se ha creado un único proyecto con todos los ejercicios en él utilizando el **IDE Netbeans**.
- Los programas **compilan**.
- Se sigue la **estructura** propuesta de **declaración de variables**, **entrada de datos**, **procesamiento** y **salida de resultados**, tal y como se indica en la plantilla y el enunciado.
- Se declaran las **variables necesarias** con el **tipo adecuado** y con **nombres apropiados**.
- No se incluyen **elementos supérfluos** ni **innecesarios**, ni se llevan a cabo **operaciones sin sentido**.
- Se declaran **identificadores** que con **nombres significativos o descriptivos** que representen de alguna manera la información que están almacenando para que **el código quede lo más claro, legible y autodocumentado posible**.
- Se llevan a cabo los **cálculos y comprobaciones necesarias** para llevar a cabo las tareas que se pide en cada actividad. Para ello habrá que construir **expresiones apropiadas** y hacer uso de los **operadores correctos**.
- El código de los programas está correctamente **indentado**.
- El código está apropiadamente **comentado**. Se observa una **corrección ortográfica y gramatical**, así como la **coherencia en las expresiones lingüísticas** en los comentarios incluidos.
- Los **identificadores** cumplen con el **convenio** sobre **"*asignación de nombres a identificadores*"** establecido para el lenguaje Java para constantes, variables, métodos, clases, etc.

- En los programas **se muestra la información esperada y correcta por pantalla en un formato apropiado**. Se observa una **corrección ortográfica y gramatical,** así como la coherencia en las expresiones lingüísticas, tanto en los textos de los mensajes que aparezcan en pantalla para pedir información de entrada al usuario como a la hora de mostrar resultados de salida. Se evitan mensajes de entrada de datos y/o salida de resultados inapropiados, descontextualizados, insuficientes o incorrectos.

Aparte de todos estas cuestiones de carácter "técnico", también se valorará en cada uno de los apartados:

- que la respuesta sea personal y no copiada de Internet o de otras fuentes;
- que sea coherente con el resto de respuestas y con los conceptos de la unidad;
- que muestre que se comprenden adecuadamente los conceptos tratados.

IMPORTANTE: Motivos de devolución de la tarea o de calificación mínima

La tarea obtendrá automáticamente una calificación de **cero** si:

- No se entrega **un único proyecto** que incluya los dos ejercicios que se piden en la tarea utilizando el IDE NetBeans.
- Alguno de los programas presenta **errores de compilación** impidiendo su ejecución y prueba.
- **No se respeta el proyecto base** que se entrega al alumno para realizar la tarea.
- Se utilizan **contenidos** en alguno de los ejercicios del proyecto **que no se han estudiado** en la unidad actual o anteriores a la misma.
- Se presenta un proyecto total o parcialmente copiado de otro alumno.

Recordatorio: Si una tarea obtiene calificación cero por alguno de los motivos previamente mencionados, será devuelta al alumno. Este tendrá derecho a una segunda entrega únicamente en los siguientes casos:

- Que se trate de la primera vez que presenta la tarea.
- Que la fecha de entrega inicial haya sido al menos una semana antes de la fecha límite establecida para dicha tarea.