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**.

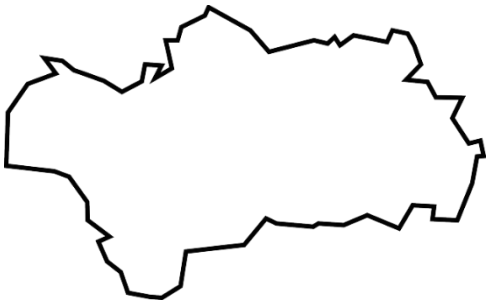
¿Qué te pedimos que hagas?

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.

Ejercicio 1: provincias de Andalucía.



Clker-Free-Vector-Images (Pixabay License)

Del mismo modo que en los contenidos de la unidad tienes un ejemplo donde se definen **unidades de medida de volumen**, crea ahora un pequeño programa en Java que defina un enumerado con los **nombres de las provincias de la comunidad autónoma de Andalucía**: **`ALMERIA`**, **`CADIZ`**, **`CORDOBA`**, **`GRANADA`**, etc. para posteriormente mostrar por pantalla cada valor del enumerado declarado.

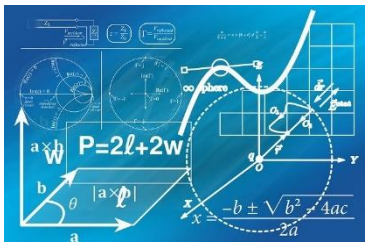
La salida por pantalla podría ser algo así:

```
PROVINCIAS DE ANDALUCÍA
-----------------------
Provincia 1: ALMERIA
Provincia 2: CADIZ
Provincia 3: CORDOBA
Provincia 4: GRANADA
Provincia 5: HUELVA
Provincia 6: JAEN
Provincia 7: MALAGA
Provincia 8: SEVILLA
```

Vamos a procurar **evitar las tildes** tanto en los nombres de las variables y de los **`enum`**, como en los identificadores de cada uno de los posibles valores del **`enum`**.

Dado que se trata de un programa tan simple en el que no se llegan a introducir datos de entrada ni se hace ningún cálculo o procesamiento, **no es necesario estructurarlo en las tres partes de entrada de datos, procesamiento y salida de resultados**.

Ejercicio 2: cálculos aritméticos.



Gerd Altmann (Pixabay License)

Escribe un programa en Java que solicite dos **números reales** y lleve a cabo los siguientes cálculos:

- el triple del primer número,
- la décima parte del segundo número,
- el cuadrado del doble del producto de ambos números,
- la mitad del cuadrado de la suma de ambos números.

Para ello tendrás que utilizar **operadores aritméticos** tales como la suma, el producto o la división. Además, es posible que en algunos casos necesites hacer uso de los **paréntesis** o, si así lo prefieres, de **variables auxiliares** para simplificar tus cálculos y no repetir operaciones.

A partir de este ejercicio deberás 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.

Ejemplos de ejecución

Aquí tienes algunos ejemplos de ejecución.

Un ejemplo de ejecución del programa podría ser:

```
CÁLCULOS ARITMÉTICOS

--------------------

Introduzca dos números reales:

Primer número: 2

Segundo número: 3


RESULTADO

---------

Triple del primer número: 6.0

Décima parte del segundo número: 0.3

Cuadrado del doble del producto de ambos números: 144.0

Mitad del cuadrado de la suma de ambos números: 12.5
```

Otro ejemplo de ejecución podría ser:

```
CÁLCULOS ARITMÉTICOS
```

```
--------------------
Introduzca dos números reales:
Primer número: 0
Segundo número: 1

RESULTADO
---------
Triple del primer número: 0.0
Décima parte del segundo número: 0.1
Cuadrado del doble del producto de ambos números: 0.0
Mitad del cuadrado de la suma de ambos números: 0.5
```

Y aquí tienes otro ejemplo más:

```
CÁLCULOS ARITMÉTICOS
--------------------
Introduzca dos números reales:
Primer número: -2
Segundo número: 2

RESULTADO
---------
Triple del primer número: -6.0
Décima parte del segundo número: 0.2
Cuadrado del doble del producto de ambos números: 64.0
Mitad del cuadrado de la suma de ambos números:
```

Ejercicio 3: análisis de un número.

Jae Rue (Pixabay License)

Escribe un programa en Java que solicite un **número entero** (sin decimales) y lo analice averiguando lo siguiente:

- si es distinto de cero,
- si es negativo,
- si está entre cero y diez, ambos incluidos,
- si es un múltiplo de tres, positivo y menor que veinte.

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

Recuerda también que un **número es múltiplo de n si es divisible entre n**, es decir, si el resto de la división entera de ese número entre n es cero. En Java dispones del operador **módulo** (**%**) para calcular el resto de la división entera.

Ejemplos de ejecución

Aquí tienes algunos ejemplos de ejecución.

Un ejemplo de ejecución del programa podría ser:

```
ANÁLISIS DE UN NÚMERO

---------------------

Introduzca un número entero: 18


RESULTADO

---------

El número es distinto de cero: true

El número es negativo: false

El número está entre cero y diez (ambos incluidos): false

El número es múltiplo de tres, positivo y menor que veinte: true
```

Otro ejemplo de ejecución podría ser:

```
ANÁLISIS DE UN NÚMERO

---------------------

Introduzca un número entero: 0
```

```
RESULTADO
---------
El número es distinto de cero: false
El número es negativo: false
El número está entre cero y diez (ambos incluidos): true
El número es múltiplo de tres, positivo y menor que veinte: false
```

Y otro ejemplo más:

```
ANÁLISIS DE UN NÚMERO
---------------------
Introduzca un número entero: -2

RESULTADO
---------
El número es distinto de cero: true
El número es negativo: true
El número está entre cero y diez (ambos incluidos): false
El número es múltiplo de tres, positivo y menor que veinte: false
```

Ejercicio 4: trayecto en dos etapas.

pixelRaw (Pixabay License)

Un vehículo realiza un trayecto en dos etapas: en la primera etapa se consume la mitad del combustible contenido en el depósito del vehículo mientras que en la segunda consumirá el 25% de lo que quede en el depósito.

Escribe un programa en Java que pida la **cantidad de litros iniciales** con los que se rellena el depósito del vehículo y calcule **cuántos litros se consumen en cada uno de los trayectos**, así como **cuántos litros quedarán en el depósito una vez finalizados ambos trayectos**.

Ejemplos de ejecución

Aquí tienes un par de ejemplos de ejecución:

Un ejemplo de ejecución del programa podría ser:

```
TRAYECTO EN DOS ETAPAS

----------------------

Introduzca la cantidad inicial de litros que se introducen en el depósito del
vehículo: 100,0


RESULTADO

---------

Litros consumidos en el primer recorrido: 50.0

Litros consumidos en el segundo recorrido: 12.5

Litros sobrantes en el depósito: 37.5
```

Otro ejemplo de ejecución podría ser:

```
TRAYECTO EN DOS ETAPAS

----------------------

Introduzca la cantidad inicial de litros que se introducen en el depósito del
vehículo: 30,8


RESULTADO

---------

Litros consumidos en el primer recorrido: 15.4

Litros consumidos en el segundo recorrido: 3.85
```

Ejercicio 5: máquina expendedora.

Crystal Chen (Pixabay License)

Una máquina expendedora de productos debe devolver el cambio usando la menor cantidad de monedas posible. Sabiendo que únicamente dispone de **monedas de veinte céntimos**, **diez céntimos**, **cinco céntimos**, **dos céntimos** y **un céntimo de euro**, escribe un programa en Java que solicite por teclado:

1. el **precio del producto** (en céntimos de euro, número entero),
2. el **dinero introducido en la máquina para adquirir el producto** (en céntimos de euro, número entero).

Y a continuación calcule y muestre por pantalla el **cambio** (en céntimos) junto con **la cantidad de monedas de cada tipo que se deben devolver** y la **cantidad total de monedas**. Recuerda que hay que **devolver la menor cantidad de monedas posible**.

Recuerda que **la información de entrada que se pide es en céntimos de euro** (**número entero**) y no en euros, que podría ser un número real. Es decir si vas a introducir 80 céntimos tendrás que teclear 80 y no 0,80. Y si vas a introducir 2 euros con 23 céntimos tendrás que teclear 223 y no 2,23.

Ejemplos de ejecución

Aquí tienes algunos ejemplos de ejecución.

Un ejemplo de ejecución del programa podría ser:

```
MÁQUINA EXPENDEDORA DE PRODUCTOS
--------------------------------
Precio del producto (en céntimos): 16
Dinero introducido (en céntimos): 100


RESULTADO
---------
El cambio es: 84 céntimos.
Monedas de 20 céntimos: 4
Monedas de 10 céntimos: 0
```

```
Monedas de  5 céntimos: 0
Monedas de  2 céntimos: 2
Monedas de  1 céntimo:  0
Total de monedas : 6
```

Aquí tienes otro ejemplo de ejecución:

```
MÁQUINA EXPENDEDORA DE PRODUCTOS
--------------------------------
Precio del producto (en céntimos): 67
Dinero introducido (en céntimos): 100

RESULTADO
---------
El cambio es: 33 céntimos.
Monedas de 20 céntimos: 1
Monedas de 10 céntimos: 1
Monedas de  5 céntimos: 0
Monedas de  2 céntimos: 1
Monedas de  1 céntimo:  1
Total de monedas: 4
```

Y otro más:

```
MÁQUINA EXPENDEDORA DE PRODUCTOS
--------------------------------
Precio del producto (en céntimos): 85
Dinero introducido (en céntimos): 100

RESULTADO
---------
El cambio es: 15 céntimos.
Monedas de 20 céntimos: 0
Monedas de 10 céntimos: 1
Monedas de  5 céntimos: 1
Monedas de  2 céntimos: 0
Monedas de  1 céntimo:  0
Total de monedas: 2
```

La forma más sencilla de resolver este tipo de ejercicios es utilizando el **cociente** (operador `/`) y el **resto** (operador módulo `%`) de la **división entera**. El primero para calcular cuántas monedas son necesarias de un tipo y el segundo para calcular cuánto dinero queda para el siguiente tipo de moneda.

Ejercicio 6: análisis de un texto.

Free-Photos (Pixabay License)

Escribe un programa en Java que lea de teclado un texto y lo analice averiguando lo siguiente:

- si el texto contiene **más de cinco caracteres**,
- si el texto **comienza por una letra mayúscula**,
- si el texto **termina con una letra minúscula**,
- si el texto **finaliza con unos puntos suspensivos** ("...").

Ejemplo de ejecución

Aquí tienes algunos ejemplos de ejecución.

Un ejemplo de ejecución del programa podría ser:

```
ANÁLISIS DE UN TEXTO

--------------------

Introduzca un texto: Hola


RESULTADO

---------

El texto contiene más de cinco caracteres: false

El texto comienza con una letra mayúscula: true

El texto termina con una letra minúscula: true

El texto termina con unos puntos suspensivos (...): false
```

Y aquí tienes otro ejemplo:

```
ANÁLISIS DE UN TEXTO

--------------------

Introduzca un texto: 12 meses


RESULTADO

---------

El texto contiene más de cinco caracteres: true

El texto comienza con una letra mayúscula: false

El texto termina con una letra minúscula: true

El texto termina con unos puntos suspensivos (...): false
```

Y otro más:

```
ANÁLISIS DE UN TEXTO

--------------------

Introduzca un texto: hola.


RESULTADO

---------
```

1. 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**.
- 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 - 21.95 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.

- 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 (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 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.

Free-Photos (Pixabay License)

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 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:

```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

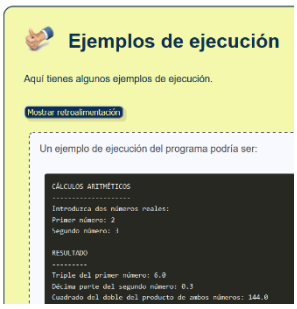
almacenarlos en las **variables auxiliares** que declaraste en la primera parte de tu programa (declaración de variables) mientras que los r**esultados finales** podrás guardarlos en las variables que hayas definido como **variables de salida**. En este bloque no deberían declararse variables (eso se 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.

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** si fuera necesario.

2.3.- Consejos para la realización de los ejercicios.

¿Qué debes revisar en los ejercicios de la tarea?

mohamed Hassan (Licencia Pixabay)

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**.
- El código debe contener **comentarios** apropiados.

3.- Evaluación de la tarea.

Criterios de evaluación implicados

Peggy_Marco (Pixabay License)

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?

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.

2. 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.	3.	10 puntos
Punto de control 2: Comenta los ejercicios con distintos tipos de comentarios. • CE i) Se han introducido comentarios en el código.	4.	10 puntos
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.	5.	10 puntos
Punto de control 4: Define y utiliza correctamente constantes y literales. • CE f) Se han creado y utilizado constantes y literales.	6.	10 puntos
Punto de control 5: Utiliza correctamente los operadores.	7.	10 puntos

• CE g) Se han clasificado, reconocido y utilizado en expresiones los operadores del lenguaje.	
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.	10 puntos
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.	10 puntos