ISSD

AED1

Algoritmos y Estructuras de Datos 1

Prof. Ing. Eduardo Mansilla

Módulo didáctico - 2016

–Asc–
Analista de Sistemas

Unidad 4

Clase 8

MÉTODOS - VECTORES

Métodos y Clases en JAVA. Zonas de un programa. Métodos que retornan valores. Vectores Paralelos.

Al final de esta clase ya podrás valorar:

| Las ventajas de reutilizar código mediante la aplicación de métodos.

| El paradigma orientado a objetos como una forma de pensar innovadora frente a la programación estructurada tradicional.

Orientarse a objetos

Hasta hace un tiempo, cuando hablábamos de "*Programación*" nos representábamos a una persona trabajando solitaria en una computadora. Sin embargo, hoy es cada vez más común pensar en programar objetos externos a la computadora.

Durante esta semana empezaremos con un tema apasionante: la programación orientada a objetos, solo a modo de introducción, ya que profundizarás este tema el próximo semestre.
Como objetivo para esta clase nos hemos propuesto estudiar las ventajas de reutilizar código mediante la aplicación de métodos y asomarnos al paradigma orientado a objetos como una forma de pensar innovadora frente a la programación estructurada tradicional.

La clase de hoy se estructura en cuatro temas:
| En el primero abordaremos los **Métodos**, teniendo en cuenta las **clases en Java**.
| En el segundo, trabajaremos sobre las **zonas de un programa completo en Java**.
| Posteriormente, estudiaremos los **métodos que devuelven** o **retornan un valor** y también, veremos **vectores utilizando métodos**.
| Finalmente, trabajaremos sobre **vectores paralelos**.

Para cada tema te proponemos ejemplos para que vayas practicando.

¡A seguir aprendiendo!

Antes de comenzar con la clase, leé el siguiente artículo.

Drones al rescate... en la playa

Nuevas tecnologías. *Por primera vez se los puso en práctica en Francia. En 30 segundos llegan hasta el bañista en peligro y le sueltan un salvavidas.*

En menos de 30 segundos despega del puesto de socorro, se posiciona sobre el bañista en peligro y le suelta un salvavidas: en las playas de Biscarrosse (suroeste de Francia) se está probando el drone Helper para rescates en el mar.

Esta innovación, probada por primera vez en Europa a partir de este miércoles, es el resultado del encuentro entre Fabien Farge, médico de emergencia de esa ciudad balnearia y de Gérald Dumartin, al frente de Terra Drone, una empresa local especializada en cartografía con drones.

"*Desde hace 15 años vi trabajar a los socorristas. Son verdaderos atletas, pero no se puede ir más allá de lo que permite la máquina del cuerpo humano. Hace dos años que estoy pensando en eso, cómo asociar un dron a un salvavidas*", explica a la AFP Fabien Farge.

Primero realizaron pruebas con un drone común pero la experiencia fue poco concluyente. "*Necesitábamos un drone capaz de resistir grandes variaciones de viento y con gran estabilidad*", explica Farge. Luego decidieron concebir un drone específico y se asociaron con la empresa Mywebteam, especializada en informática y objetos conectados. El resultado es un aparato de 3,9 kg con una cámara de alta definición para visualizar la persona en peligro, un salvavidas capaz de inflarse automáticamente al entrar en contacto con el agua y un sistema para soltarlo.

Para los que lo concibieron, las ventajas del drone Helper (*Human Environment and Life Protection Emergency Response*) son numerosos. Tiene una velocidad de desplazamiento de entre 55 y 80 km/hora e indica rápidamente a la víctima que ha sido tomada en cuenta, lo que evita que entre en pánico. Una cámara permite ver en qué estado se encuentra para decidir qué medidas hay que adoptar. Además el uso del drone como guía visual permite orientar a los socorristas que salen a buscarlo.

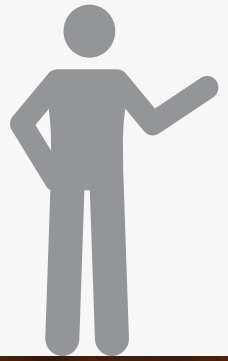
"El drone también permite despejar dudas sobre una presunta víctima y evitar poner en peligro la vida del socorrista", destaca Fabien Farge. Los inventores destacan que el drone se puede usar con condiciones meteorológicas poco favorables, vientos de hasta 50 km/hora y un oleaje que justifique bandera amarilla para el baño.

Durante una demostración, el aparato llegó en apenas 22 segundos a la altura de uno de sus colegas que hacía el papel de bañista en dificultades, a 100 metros de la orilla.

Salido a nado desde la playa en el mismo instante, un socorrista aún se encuentra a unos cincuenta metros del "ahogado" cuando el drone larga el salvavidas. En cuanto al jet-ski, herramienta indispensable para los socorristas, tiene que ser llevado hasta el agua, domar las olas y abrirse paso con precaución entre los bañistas.

Clarín – Julio de 2016

Los robots son dispositivos capaces de actuar de manera autónoma o semiautónoma, con la capacidad de adquirir datos del ambiente a partir de sensores y de tomar decisiones en consecuencia, de acuerdo a las instrucciones previstas por un programador.



Uno de los dispositivos que más terreno ha ganado en los útimos años, es el dron. Estos aparatos han crecido en masividad y consecuentemente, disminuido su costo, convirtiéndose de esta manera en asequibles para el gran público.

No es extraño leer noticias como la precedente, en las que se describen situaciones de nuevos usos o aplicaciones para esta tecnología.

Métodos

Con lo que has experimentado programando hasta aquí, te habrás dado cuenta de que hay porciones de código que se repiten una y otra vez y que son útiles en la resolución de diferentes tipos de problemas.

Es una pérdida de tiempo reescribir lo que ya hicimos, y sería muy bueno poder reutilizarlo o "convocarlo" de manera sencilla cada vez que lo precisemos, ya que sin dudas, optimizaría nuestro trabajo.

Una manera para lograr esto, es agrupar la lógica de programación en módulos que cumplan una determinada función, como por ejemplo, cargar un vector, o mostrar en pantalla un resultado. Una vez desarrollada esa porción modular de código, bastará con que la llamemos cada vez que sea necesaria, pasándole las particularidades de cada situación.

Supongamos que desarrollamos un módulo que dibuja un gráfico estadístico en la pantalla. Podríamos reutilizarlo si previéramos que cada vez que lo ejecutemos, le pudiéramos pasar los datos correspondientes a cada caso particular (parámetros), haciéndolo más flexible.

Tomemos ahora como ejemplo a un **dron**.
Si realizamos su programación, deberemos prever diferentes Métodos, como por ejemplo, *Ascender*, *Descender* o *Filmar*.
Cada Método a su vez, puede contemplar la recepción de parámetros. Por ejemplo, cuando llamemos al método *Filmar* le indicaremos el tiempo que durará la grabación y el nombre del archivo en donde almacenará el resultado.
O el Método *Ascender* debería recibir como parámetro la altura a la cual queremos que llegue. Posiblemente el Método *Descender* no lleve parámetros, ya que simplemente bajará de altura hasta que toque la superficie y apague sus motores.

Cada método, será una porción de código con numerosas instrucciones. La ventaja está en que una vez definido, nos limitaremos a llamarlo indicándole los nuevos datos en forma de parámetros.
Se puede decir que en Java, toda la lógica de

Se define a una Función o Método, como una fracción de programa que cumple una misión específica. Generalmente el término **Función** es utilizado en la Programación Estructurada y el término **Método**, es utilizado en la Programación Orientada a Objetos (POO).

1

```
void cargarDatos()
 {
    Sentencias;
 }
```

Este Método se llama cargarDatos, no recibe ningún parámetro ni devuelve ningún valor.

```
int cubo (int x)
 {
    Sentencias;
 }
```

Este Método se llama cubo, recibe un valor entero como parámetro, y devuelve un valor entero.

```
void listadoDePrecios()
 {
    Sentencias;
 }
```

Este Método se llama listadoDePrecios, no recibe ningún parámetro ni devuelve ningún valor

programación está agrupada en Funciones o Métodos.
Sobre la Programación Orientada a Objetos aprenderás en la siguiente materia, por ahora veremos solamente algunos elementos básicos, sin meternos en su escencia.

Básicamente un Método es:
| Un bloque de código que tiene un nombre,
| Recibe opcionalmente parámetros o argumentos,
| Contiene sentencias o instrucciones,
| Devuelve opcionalmente un valor de algún tipo conocido.

La sintaxis es la siguiente:

```
TipoDevuelto nombreDelMetodo (
ListaDeParametros )
      {
        Bloque de código;
      }
```

El **TipoDevuelto** puede ser cualquiera de los tipos vistos hasta el momento, por ejemplo: *int, char, float, String*.

En caso de que el método no devuelva ningún valor se coloca la palabra **void**.

Los nombres de los métodos siempre deben comenzar con una letra en minúscula; cuando se desea asignarles nombres formados por dos o más palabras, se coloca la primera letra de la primera palabra en minúscula y la primera letra de las siguientes palabras en mayúsculas.

Hay que tener en cuenta que, al igual que los nombres de las variables, no se pueden colocar espacios en blanco entre las palabras del nombre del método.

Observá en la imagen 1 algunos ejemplos válidos de nombres de métodos.

Se define a una Clase como una colección de variables y métodos, a partir de la cual se pueden crear diversos ejemplares. Cada ejemplar de una clase, recibe el nombre de Instancia de la clase y se pueden crear tantas instancias como se quiera.

¿Qué significa *enviarle un parámetro*? ¿Qué quiere decir que *devuelve un valor*?
Por ejemplo, podemos crear un Método cuya función consista en calcular la superficie de un cuadrado. En ese caso, cada vez que necesitemos ejecutarlo, tendremos que indicarle el tamaño del lado del cuadrado (parámetro), y como resultado nos dará su superficie (valor que devuelve).

Clases en Java

Sin embargo, no nos meteremos por ahora en este concepto, solo lo mencionaremos para entender cuáles son las zonas de un programa en Java.

Hasta ahora, hemos pasado por alto algunas de las primeras líneas de cada programa que hicimos en Java, marcándolas en negro. Veamos en el próximo tema un poco más de cerca, qué significan.

Zonas de un programa completo en Java

Las diferentes zonas que intervienen en un programa en lenguaje Java son las siguientes:

| Importación de librerías. Por ejemplo:
import hsa.Console

| Nombre de la clase. Por ejemplo:
class Listado

| Zona de declaración de variables miembros de la clase. Por ejemplo:
String nombre;
int edad;
float sueldo;

| Declaración del Constructor (Opcional).

| Zona de declaración de Métodos.

| Programa principal. Por ejemplo:
public static void main(String[] args)

Analicemos ahora estas partes, mediante un ejemplo.

Ejemplo 37

Cálculo del promedio usando Métodos

Desarrollar un programa que tenga dos métodos, uno para la introducción de tres números y otro método para calcular e imprimir el promedio de ellos.

Observá primero la imagen 1 de la siguiente página.

Zonas que intervienen en un programa en Java:

| Importación de librerías.

| Nombre de la clase.

| Zona de declaración de variables miembros de la clase.

| Declaración del Constructor.

| Zona de declaración de Métodos.

| Programa principal.

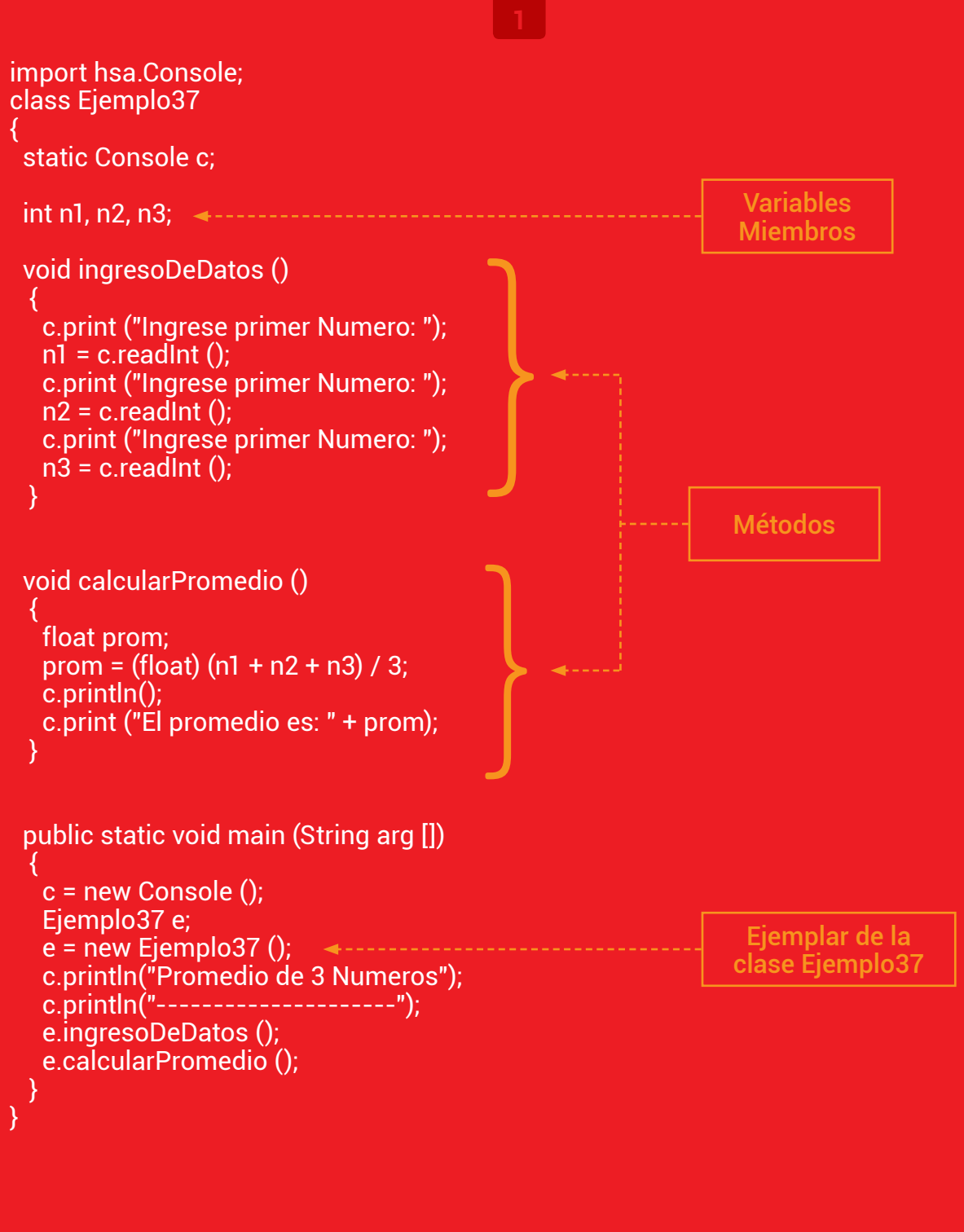
La explicación para el esquema de la imagen 1 es la siguiente: Al principio se coloca la línea de Importación de librerías:
import hsa.Console;

indicando que se incluyan en el programa todas las instrucciones de la clase **Console**.

En la segunda línea, se define la clase, llamada en este caso **Ejemplo37**.
Luego, en la tercera línea se define el objeto c de la clase **Console**, este objeto estará capacitado para utilizar los métodos de entrada y salida.

A continuación, se definen las variables miembros que se utilizarán en el programa, y luego se realizan los métodos de ingreso y de calcular el promedio de los números ingresados.

El programa principal es el encargado de llamar a cada uno de los métodos necesarios para la resolución del problema planteado.
En el programa principal, al igual que en los ejemplos anteriores, se crea un objeto de la clase **Console**, luego se define un objeto de la clase **Ejemplo37** llamado "e":
Ejemplo37 e;



Posteriormente se crea el objeto "e", con la línea:
e = new Ejemplo37 ();

Estas dos líneas realizadas anteriormente, se pueden organizar en una sola, de la siguiente manera:
Ejemplo37 e = new Ejemplo37 ();

Se coloca a modo de título, la leyenda "Promedio de 3 Números" y se la subraya, luego se llaman a los métodos realizados, teniendo en cuenta que para llamar a un método, se lo hace colocando el nombre del objeto, luego un punto y finalmente el nombre del método. En este caso:
e.ingresoDeDatos ();
e.calcularPromedio ();

Veamos otro ejemplo más.

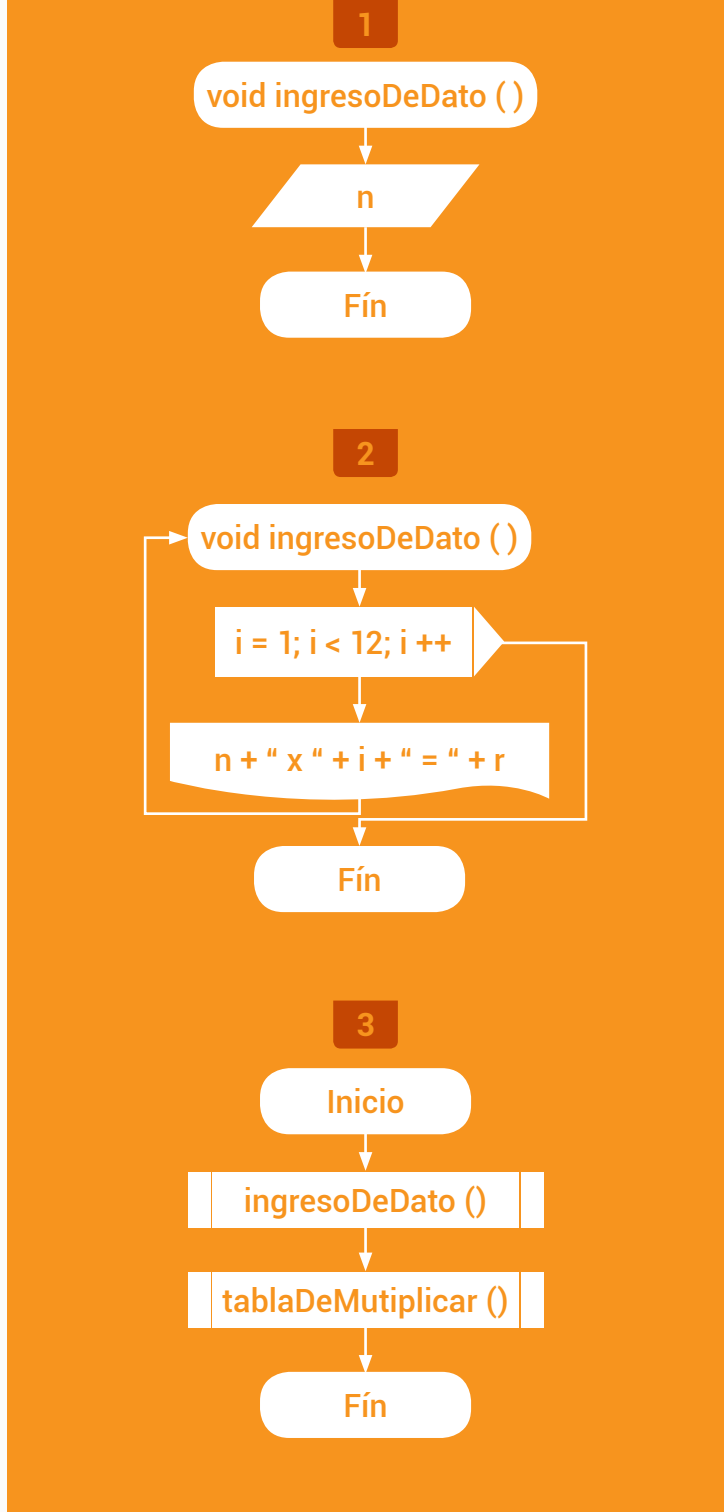
Ejemplo 38

Tablas de multiplicar

Realizar un programa que permita ingresar un número y muestre su tabla de multiplicar del 1 al 12.

Los diagramas de los métodos serían los que podés observar en la imagen 1 y 2 y el diagrama del programa principal sería el de la imagen 3. Su respectivo código se encuentra en la imagen 4 en la siguiente página.

Es importante apreciar la ventaja de trabajar con Métodos o Funciones: es mucho más sencillo seguir la lógica del programa principal, ya que el problema complejo se ha desagregado en pequeños. Además, un Método puede ser utilizado más de una vez, sin necesidad de repetir su programación.



La variable "*r*" que se define en el método **tablaDeMultiplicar**, recibe el nombre de **Variable Local**, y posee la particularidad de que tiene acción únicamente dentro del método, fuera de él, no existe. Es decir, que el espacio que estaba ocupando en el momento en que se encontraba utilizada, queda liberado cuando la ejecución del programa sale del método.

```
import hsa.Console;
class Ejemplo38
{
 static Console c;
 int n;

 void ingresoDeDato ()
 {
  c.print ("Ingrese un Numero: ");
  n = c.readInt ();
 }

 void tablaDeMultiplicar ()
 {
  int r;
  c.println ();
  c.println ("Tabla de Multiplicar del numero: " + n);
  c.println ();
  for (int i = 1 ; i <= 12 ; i++)
   {
    r = n * i;
    c.println (n + " x " + i + " = " + r);
   }
 }

 public static void main (String arg [])
 {
  c = new Console ();
  Ejemplo38 e = new Ejemplo38 ();
  e.ingresoDeDato ();
  e.tablaDeMultiplicar ();
 }
}
```

Métodos que devuelven o retornan un valor

Cuando se desea que un método devuelva un valor, es necesario colocar un **Tipo** en la cabecera del método, delante del nombre del mismo. El Tipo colocado, tiene que ver con el Tipo de valor que se desea que retorne el método.

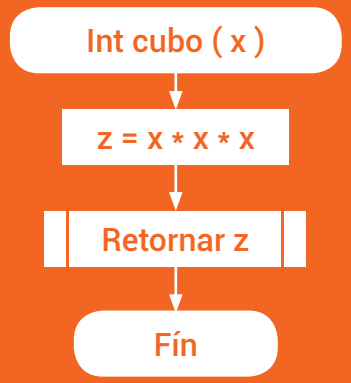
Veamos esto con algunos ejemplos.

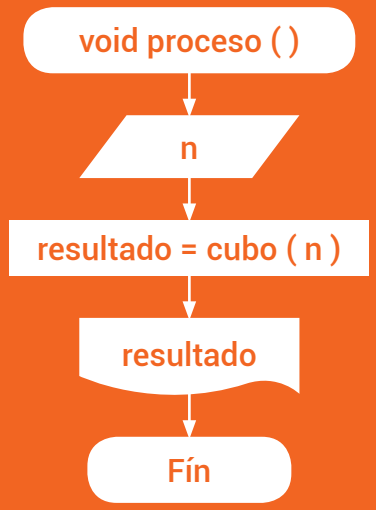
Ejemplo 39

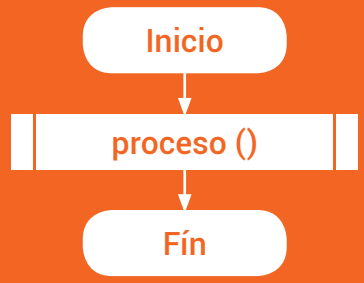
Cubo de un número

Desarrollar un programa que tenga un método que acepte un número entero y retorne el cubo del mismo.

1







Observá las imágenes 1 (en la página anterior) y 2. Ellas corresponden a los diagramas y al código respectivamente.

Es muy importante que no te quedes con dudas ya que todo lo que vamos aprendiendo es necesario para comprender los siguientes pasos. Recordá que podés hacerle cualquier consulta a tu tutor.

2

```
import hsa.Console;
class Ejemplo39
{
   static Console c;

   int cubo (int x)
   {
      int z;
      z = x * x * x;
      return z;
   }

   void proceso ()
   {
      int n, resultado;
      c.print ("Ingrese un Numero: ");
      n = c.readInt ();
      resultado = cubo (n);
      c.print ("Su cubo es: " + resultado);
   }
   public static void main (String arg [])
   {
      c = new Console ();
      Ejemplo38 e = new Ejemplo38 ();
      e.proceso ();
   }
}
```

Vectores utilizando Métodos

Sigamos analizando este tema con otro ejemplo.

Ejemplo 40

Sueldos

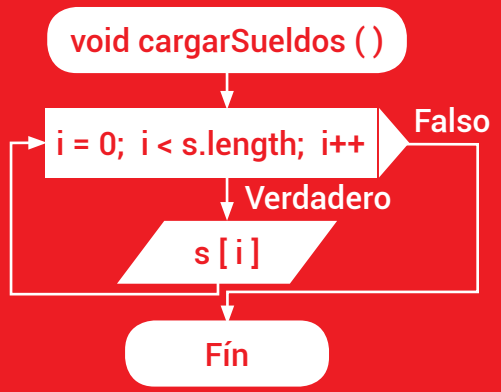
En una empresa se almacenaron los sueldos de 10 personas. Desarrollar un programa para realizar:

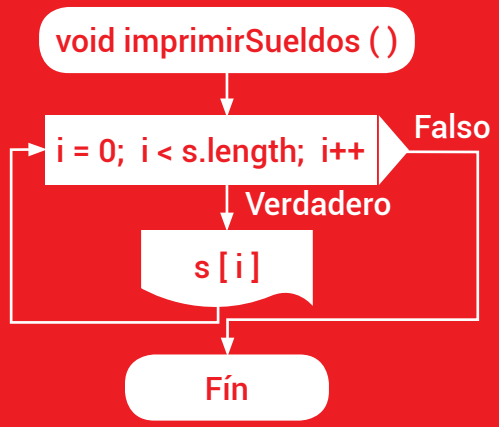
1 | Carga de los sueldos en un vector.

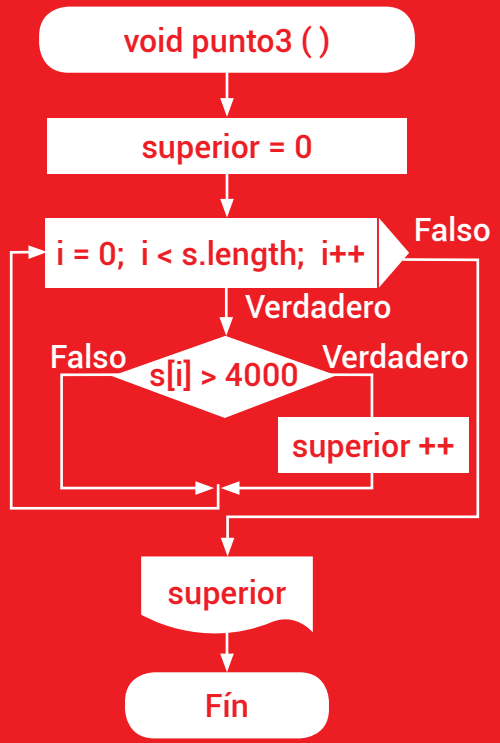
2 | Impresión de todos los sueldos.

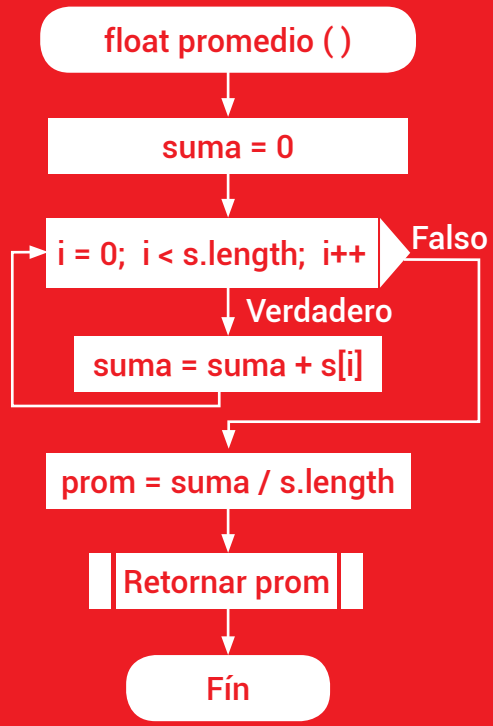
3 | ¿Cuántos tienen un sueldo superior a $4000?

4 | Mostrar todos los sueldos que están por debajo del promedio.







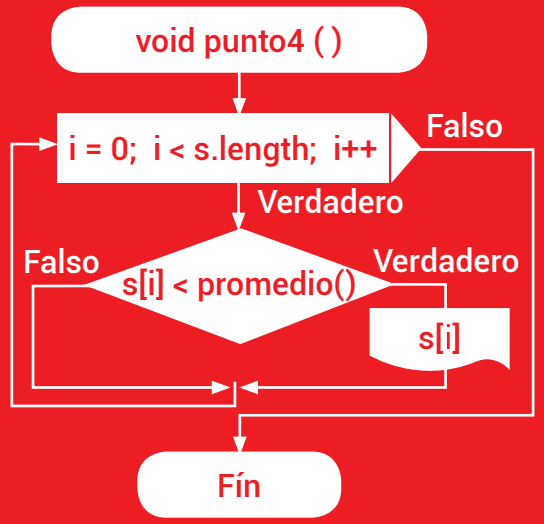


Continúa en la siguiente página...

Observá la imagen 1 (en esta página y la anterior) y la 2 (en la que sigue) que corresponden a los diagramas y al código respectivamente.

Intentá comprender de qué se trata vos solo. Como es un poco complicado en páginas posteriores te explicamos nosotros como es el proceso para que no te quedes con dudas.

1



Progama Principal

```
import hsa.Console;
class Ejemplo40
{
 static Console c;
 float [] s;

 Ejemplo40 ()
  {
    s = new float [10];
  }

 void cargarSueldos ()
 {
  c.println ("Cargue los sueldos de las 10
personas: ");
   for (int i = 0 ; i < s.length ; i++)
    {
     c.print ("Ingrese sueldo[" + (i + 1) + "]:
");
     s [i] = c.readFloat ();
    }
 }

 void imprimirSueldos ()
 {
  c.println ("Sueldos almacenados: ");
  for (int i = 0 ; i < s.length ; i++)
   {
    c.println ("    $ " + s [i]);
   }
 }

  void punto3 ()
  {
   int superior = 0;
   for (int i = 0 ; i < s.length ; i++)
    {
     if (s [i] > 4000)
        superior++;
    }
   c.println ();
   c.println ("Los sueldos superiores a
$4000 son: " + superior);
  }

  float promedio ()
   {
    float suma = 0;
    float prom;
    for (int i = 0 ; i < s.length ; i++)
{
  suma = suma + s [i];
}
prom = suma / s.length;
return prom;
  }
void punto4 ()
 {
  c.println();
  c.println("El promedio de sueldos es: $ "
+ promedio());

  c.println();
  c.println("Los sueldos inferiores al
promedio son: ");
  for (int i = 0 ; i < s.length ; i++)
   {
    if (s[i] < promedio() )
       c.print(s[i] + "  ");
   }
 }

 public static void main (String arg [])
  {
    c = new Console ();
    Ejemplo40 e = new Ejemplo40 ();
    e.cargarSueldos ();
    c.clear();
    e.imprimirSueldos();
    e.punto3();
    e.punto4();
  }
}
```

Veamos ahora la explicación de este ejercicio.

El programa comienza definiendo la variable "s" (sueldo) como un vector de componentes *float*, definiendo así, una **variable miembro** de la clase **Ejemplo40**.

En este ejemplo se ha colocado por primera vez, la zona del *Constructor*. En este caso se ha utilizado para crear el vector de sueldos.

Para que un método sea Constructor, es necesario que tenga el mismo nombre que la clase, en este caso, Ejemplo40, y debe llevar los paréntesis, dentro de los cuales, opcionalmente puede haber parámetros.

A continuación se encuentra el método **cargarSueldos()**, que es el encargado de pedir los sueldos que se quieren almacenar en el vector. La leyenda impresa en el **c.print:**

```
c.print ("Ingrese sueldo[" + (i + 1) + "]: ");
```

tiene por objeto mostrar la palabra sueldo con un subíndice que comienza en 1, debido a (i + 1). Esto es simplemente a modo visual, porque internamente, en la memoria, el vector se almacenará desde la posición cero. En la pantalla aparecerá:

```
Ingrese sueldo[1]:
Ingrese sueldo[2]:
        Etc.
```

Luego está el método para imprimir los sueldos almacenados en el vector, el que simplemente consta de un ciclo **for** con la impresión dentro de él.

El siguiente método, es el punto 3 del enunciado del ejercicio, es decir cuántos tienen un sueldo superior a $4000. Para ello, se utiliza un ciclo **for** y dentro de él, un **if** para preguntar si algunos de los sueldos es mayor a 4000, que en caso de ser verdadero, se incrementa en uno la variable **superior**, que es la encargada de almacenar los mayores a 4000. Fuera del ciclo **for**, se imprime el valor de la variable **superior**.

A continuación, se realizó el método del cálculo del promedio, el que retorna un valor, por lo tanto tiene que tener un tipo delante del nombre. Como un promedio siempre es un resultado numérico Real, se coloca el tipo **float** y para su cálculo, se realiza la suma de todos los sueldos y posteriormente se la

Un Constructor es un método que se utiliza para **inicializar las variables miembros de la clase**. Todo Constructor, tiene la particularidad de ser el primer método que se ejecuta cuando se inicia la ejecución del programa y no es necesario invocarlo en el programa principal.

divide por la cantidad de elementos. Antes de finalizar el método, se debe colocar la instrucción **return** con la variable calculada.

El método **punto4** es el encargado de llamar al método promedio para averiguar cuáles sueldos son inferiores al **promedio**. Este proceso consiste en un ciclo **for** y dentro de él, una pregunta solicitando al método **promedio()**. En caso de que la pregunta resulte verdadera, se imprime el sueldo en cuestión.
Por último, se realiza el programa principal, creando un objeto de la clase **Ejemplo40** para poder llamar a los diversos métodos de la clase.

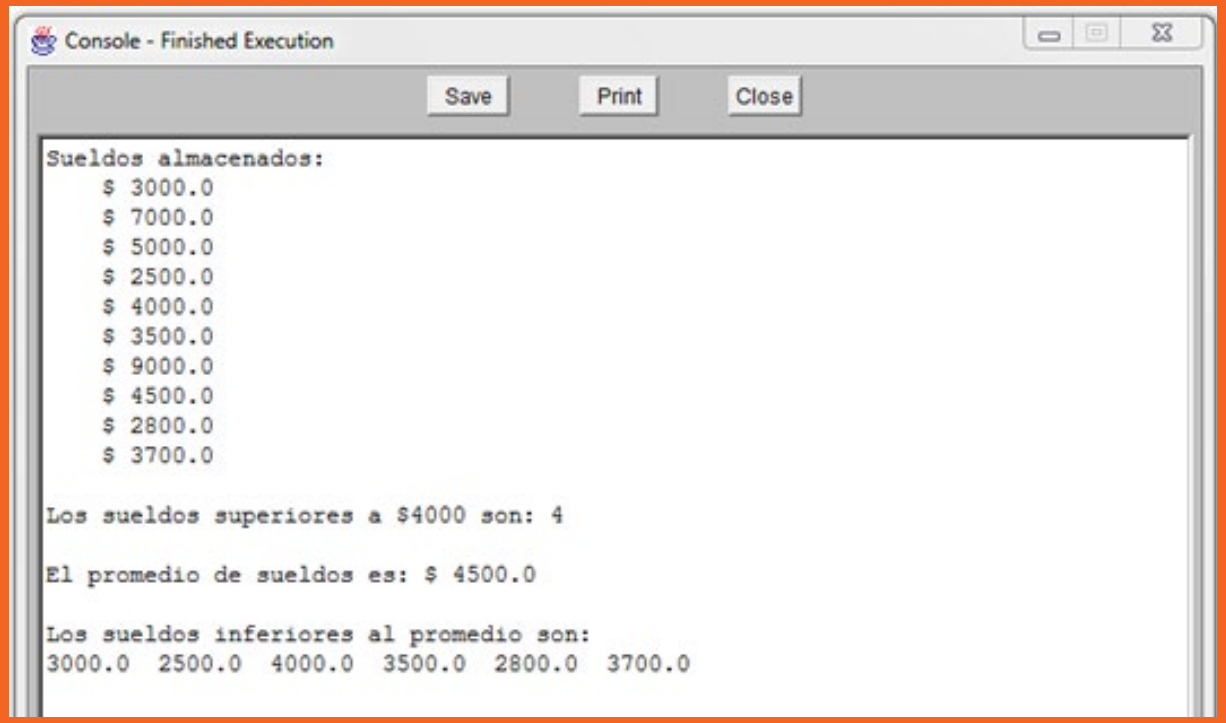
La ejecución del programa daría el resultado que observamos en la imagen 3.

Ejemplo 41

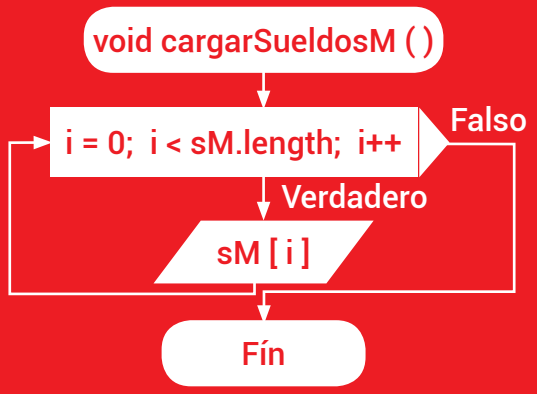
Sueldos por turno

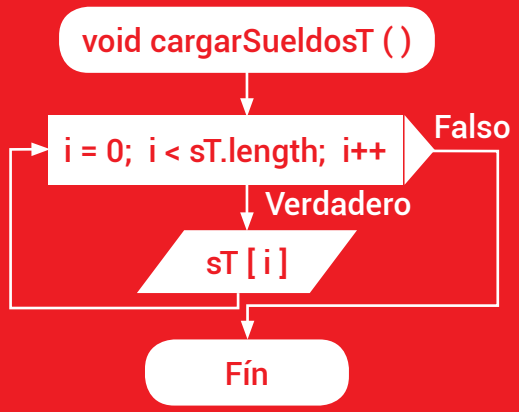
Un comercio tiene dos turnos (mañana y tarde) en los que trabajan 8 empleados (4 por la mañana y 4 por la tarde). Confeccionar un programa que permita almacenar los sueldos de los empleados agrupados por turno. Imprimir los gastos en sueldos de cada turno.

3



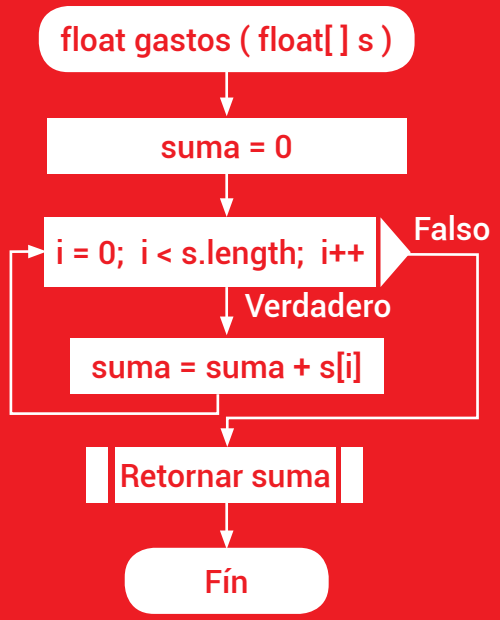
1



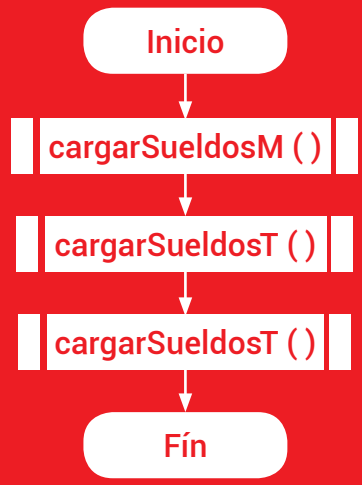


Continúa en la siguiente página...

Observá primero los diagramas de la imagen 1 que comienza en la página anterior.



Progama Principal



El programa quedaría como te muestra la imagen 2.

En las páginas siguientes te contamos qué es lo nuevo en este ejemplo.

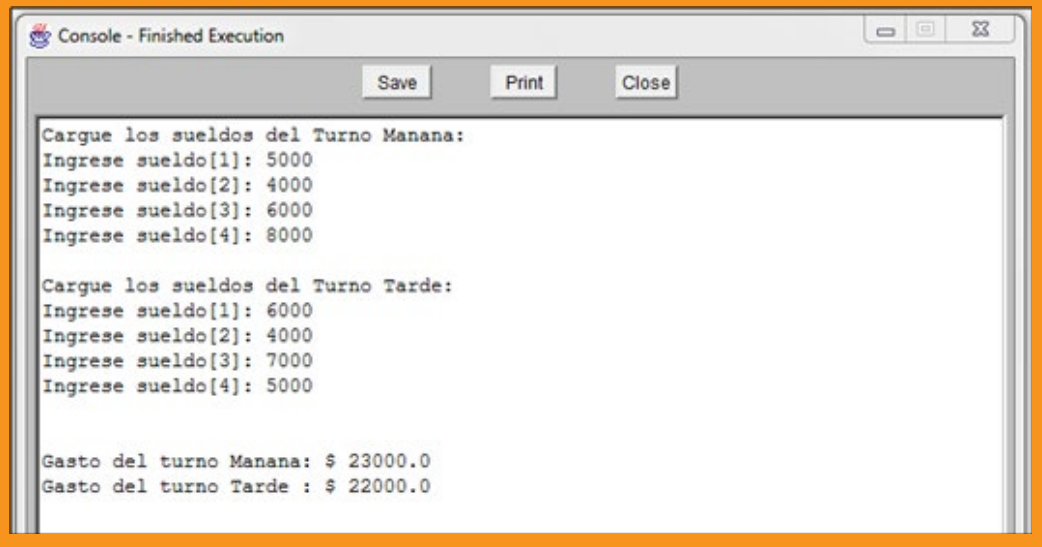
2

```
import hsa.Console;
class Ejemplo41
{
 static Console c;
 float [] sM;
 float [] sT;

 Ejemplo41 ()
 {
  sM = new float [4];
  sT = new float [4];
 }

 void cargarSueldosM ()
 {
  c.println ("Cargue los sueldos del
Turno Manana: ");
  for (int i = 0 ; i < sM.length ; i++)
  {
   c.print ("Ingrese sueldo[" + (i + 1) +
"]: ");
   sM [i] = c.readFloat ();
  }
 }

 void cargarSueldosT ()
 {
  c.println ("Cargue los sueldos del
Turno Tarde: ");
  for (int i = 0 ; i < sT.length ; i++)
  {
   c.print ("Ingrese sueldo[" + (i + 1) +
"]: ");
   sT [i] = c.readFloat ();
  }
 }

 float gastos (float [] s)
 {
  float suma = 0;
  for (int i = 0 ; i < s.length ; i++)
  {
   suma = suma + s [i];
  }
  return suma;
 }

 void imprimirGastos ()
 {
  c.println ();
  c.println ("Gasto del turno Manana: $ " +
gastos(sM));
  c.println ("Gasto del turno Tarde : $ " +
gastos(sT));
 }
 public static void main (String arg [])
 {
    c = new Console ();
    Ejemplo41 e = new Ejemplo41 ();
    e.cargarSueldosM ();
    c.println();
    e.cargarSueldosT ();
    c.println();
    e.imprimirGastos ();
 }
}
```

Definición de los Vectores.

Creación de los Vectores

Llamado al método gastos con los vectores sM y sT.



La ejecución del programa muestra la salida que podés ver en la imagen 3.

Lo nuevo en este ejemplo, es el método:

```
float gastos (float [] s)
 {
  float suma = 0;
  for (int i = 0 ; i < s.length ; i++)
   {
    suma = suma + s [i];
   }
  return suma;
 }
```

Este método acepta como parámetro un vector de componentes tipo **float** y retorna el valor de la suma de los componentes de dicho vector pasado como parámetro y establece una de las ventajas de trabajar con procesos independientes, ya que a él se le puede pasar cualquier vector y entregará la suma de sus componentes.

En este caso, se le enviarán los vectores del turno mañana y del turno tarde, cosa que se realiza en el método **imprimirGastos**.

Es de suma importancia comprender la **reutilización de los métodos**, ya que justamente esa es una de las potencialidades de la programación.

En el ejercicio anterior, el método gastos, es utilizado en dos oportunidades y recibe distintos vectores realizando el cálculo de la suma de cada uno sin ningún problema lo que ahorra escribir una nueva función para que trabaje con el segundo vector.

Vectores Paralelos

El concepto de Vectores Paralelos se da cuando hay dos o más vectores relacionados entre sí a través del subíndice.

Por ejemplo:

	0	1	2	3	4
Producto	Televisor	Notebook	Radio	Dvd	Ventilador
Precios	4500	3100	340	650	280

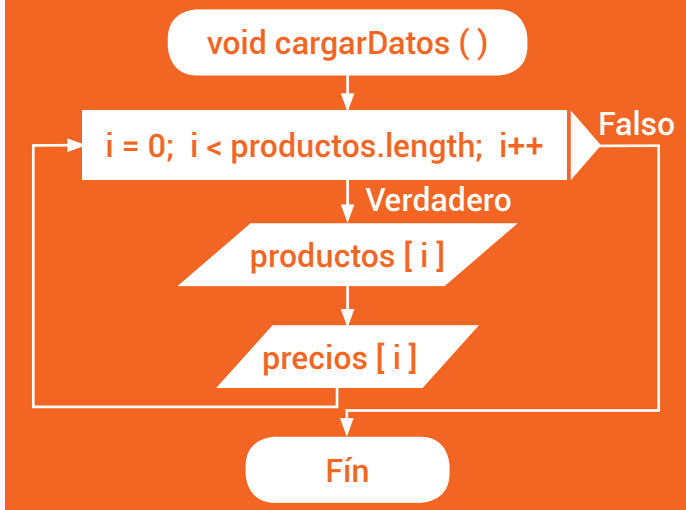
Los dos vectores, **Productos y Precios**, se encuentran relacionados por medio de la posición que ocupa cada elemento; así, al producto **Televisor** que está en la posición cero, le corresponde el precio de **4500**, que se encuentra también en la posición cero, al producto Notebook que está en la posición 1 le corresponde el precio 3100, y así sucesivamente. Apliquemos este concepto en el siguiente ejemplo.

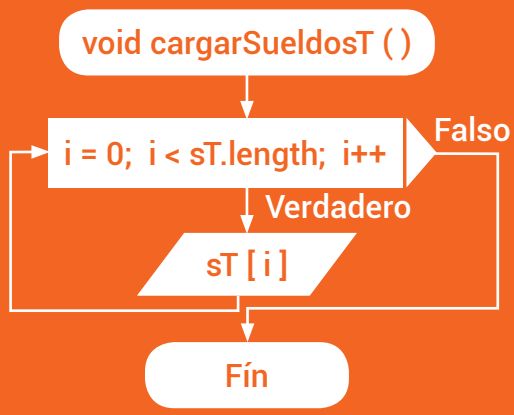
Ejemplo 42

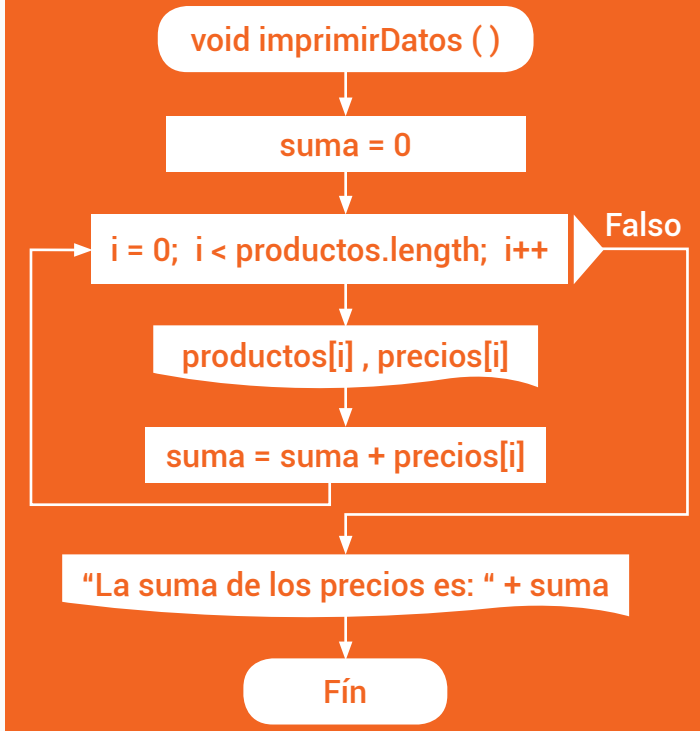
Productos y Precios

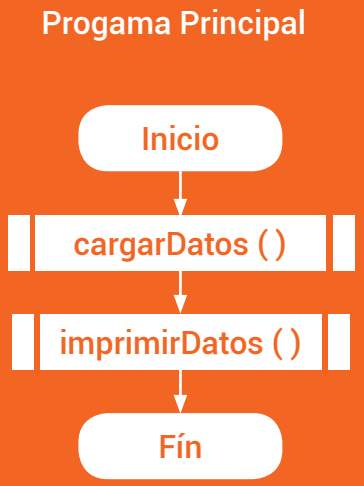
Realizar la carga de los vectores Productos y Precios de 5 artículos. Posteriormente, realizar un listado en dos columnas indicando el valor acumulado de todos los productos.

Los diagramas correspondientes, se muestran en la imagen 1. Al programa podés observarlo en la imagen 2 que se encuentra en la página siguiente.







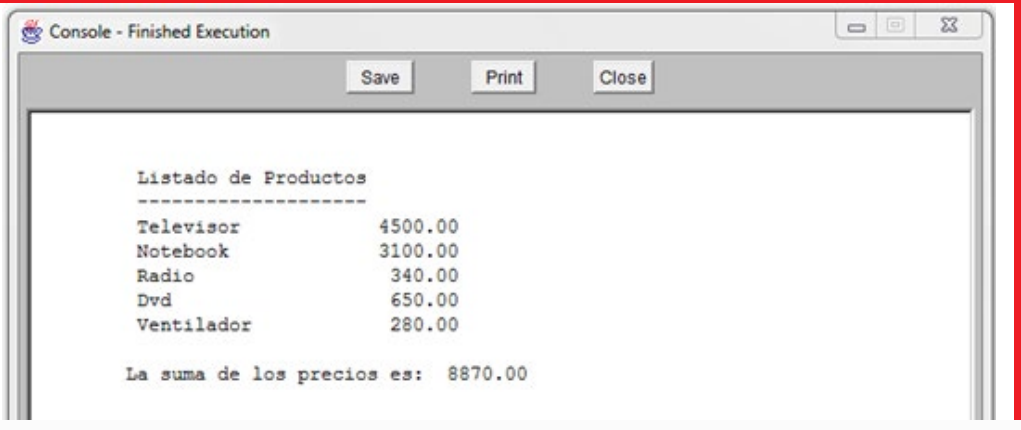


```
import hsa.Console;
class Ejemplo42
{
   static Console c;

   String [] productos;
   float [] precios;

   Ejemplo42 ()
   {
      productos = new String [5];
      precios = new float [5];
   }

 void cargarDatos ()
 {
  c.println ("Cargue los datos de los
articulos: ");
  c.println ();
  for (int i = 0 ; i < productos.length ; i++)
   {
    c.print ("Ingrese producto[" + (i + 1) +
"]: ");
    productos [i] = c.readLine ();
    c.print ("Ingrese su precio: ");
    precios [i] = c.readFloat ();
    c.println ();
   }
 }


 void imprimirDatos ()
 {
  float suma = 0;
  int linea = 5;
  c.setCursor (3, 10);
  c.println ("Listado de Productos");
  c.setCursor (4, 10);
  c.println ("--------------------");
  for (int i = 0 ; i < productos.length ; i++)
   {
    c.setCursor (linea, 10);
    c.print (productos [i]);
    c.setCursor (linea, 30);
    c.print (precios [i],8,2);
    linea++;
    suma = suma + precios [i];
   }
  c.println ();
  c.println ();
  c.print ("       La suma de los precios es:
");
  c.print (suma,8,2);
 }


 public static void main (String arg [])
 {
  c = new Console ();
  Ejemplo42 e = new Ejemplo42 ();
  e.cargarDatos ();
  c.clear ();
  e.imprimirDatos ();
 }
}
```

Definición de los Vectores

Creación de los vectores

Se cargan los dos vectores en el mismo ciclo for



Lo nuevo en este programa está en el método **imprimitDatos()**, en donde se utiliza la instrucción **c.setCursor** para posicionarse en un lugar de la pantalla.

Esta instrucción acepta como parámetros la fila y la columna en donde se quiere posicionar, en el ejemplo figura:

```
c.setCursor (3, 10);
c.println ("Listado de Productos");
c.setCursor (4, 10);
c.println ("--------------------");
```

La primera línea establece que se posicione el cursor en la fila 3 y columna 10 y luego imprima la leyenda **"Listado de Productos"**, posteriormente, se posiciona una línea más abajo, línea 4, y en la misma columna 10, para subrayar el título.

Posteriormente, se continúa con el ciclo **for** para realizar el listado completo de todos los productos con sus respectivos precios. Se posiciona el cursor donde lo indica el valor de la variable "linea", que ha sido inicializada en 5 y en la columna 10, allí se imprime el nombre del primer producto, o sea, **productos[0]**, porque la primera vez la variable "i" vale cero, luego en la misma línea 5, pero en la columna 30, se imprime el valor del precio correspondiente a la posición cero.

El valor de **8,2** que figura en la impresión, tiene el siguiente significado: **8** significa el valor del campo en donde se debe imprimir, es decir 8 espacios, y **2** indica la cantidad de decimales con que se deberá imprimir el precio del producto. Es para que todos los precios se muestren alineados a la derecha y con dos decimales.

Antes de cerrar el ciclo **for**, se debe incrementar la variable "linea", para seguir imprimiendo en las líneas sucesivas los otros pares de datos. Además, también antes de cerrar el **for**, se deben acumular los precios pera poder imprimir fuera de él, su suma.

La salida del programa se verá como lo muestra la imagen 3.

Desempeños

Desempeño 86

Desarrollá un programa que permita ingresar un vector de 7 elementos y luego informe:
a | El valor acumulado de todos los elementos.
b | El valor acumulado de todos los elementos mayores a 50.
c | La cantidad de valores mayores a 50.

Desempeño 87

Cargá 3 vectores de 5 elementos cada uno. Se deberá imprimir un vector por línea junto con el promedio de cada uno. Por último, determiná cuál es el que tiene mayor promedio.

Desempeño 88

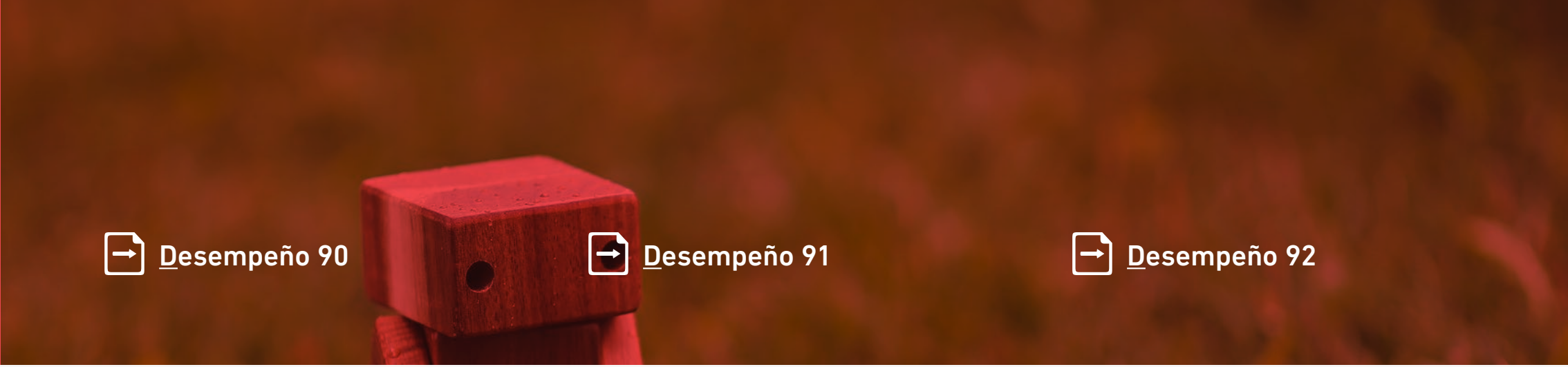
En un curso de 8 alumnos se registraron las notas de sus exámenes y se deben procesar de acuerdo a lo siguiente:
a | Ingresar Nombre y Nota de cada alumno.
b | Realizar un listado en tres columnas de Nombres, Notas y Condición de alumno. En la columna Condición, colocar "Muy Bueno" si la nota es mayor o igual a 8, "Bueno" si la nota está entre 4 y 7.99, y colocar "Insuficiente" si la nota es inferior a 4.
c | Imprimir cuántos alumnos tienen la leyenda "Muy Bueno".

Por ejemplo:

Listado de notas de alumnos		
Nombre	Nota	Condición
Javier	6	Bueno
Marcelo	9	Muy Bueno
Viviana	7	Bueno
Raúl	3	Insuficiente
Ricardo	10	Muy Bueno
Roxana	5	Bueno
Hugo	2	Insuficiente
Rita	8	Muy Bueno
Cantidad de alumnos Muy Buenos: 3		

Desempeño 89

Cargá en dos vectores Nombres de productos con sus respectivos Precios. Posteriormente ingresá un valor y mostrá todos los nombres que tienen un precio mayor al valor ingresado.



Desempeño 90

Desarrollá un método que acepte dos vectores del mismo tamaño como parámetros, y retorne el producto escalar de los mismos.
El Producto Escalar de dos vectores, es un número que se calcula acumulando los productos de las componentes del mismo índice.
Por ejemplo:

V1[i] = [2, 8, 5, 3, 8]
V2[i] = [6, 3, 9, 4, 7]

Producto Escalar = 2 * 6 + 8 * 3 + 5 * 9 + 3 * 4 + 8 * 7 = 149

Desempeño 91

En una empresa de 12 empleados se necesitan realizar las siguientes operatorias:
a | Ingresar los nombres de los empleados con sus respectivos sueldos.
b | Indicar cuántos cobran más de $3500.
c | Incicar cuántos cobran menos de $2500.
d | Imprimir en dos columnas, los nombres de los empleados con sus respectivos sueldos.
e | Imprimir el monto total que debe tener la empresa para abonar el sueldo a sus empleados.

Desempeño 92

Desarrollar un programa que permita cargar las frecuencias de 7 emisoras de FM con sus respectivos nombres. Luego realizá lo siguiente:

a | Impresión en dos columnas de las frecuencias y los nombres de emisoras.
b | Ingresar un valor de emisora y mostrar todos los nombres de emisoras que tienen una frecuencia menor a la ingresada.

Bibliografía

Imágenes

www.pexels.com

www.pixabay.com

www.flickr.com

Una vez cargados los vectores, es posible realizar con ellos operaciones muy interesantes tales como la búsqueda y el ordenamiento.
La próxima clase aprenderemos a desarrollar algoritmos que realizan este tipo de tareas, pero quizá lo más valioso, sea la “gimansia” mental que nos habilitará para pensar y crear algoritmos cada vez más sofisticados.

¡Nos vemos!