**DOCUMENTO TALLER PRÁCTICA**

**ESTRUCTURA: PROGRAMACION ORIENTADA A OBJETOS**

**MODULO: Programación en JAVA**

**DOCUMENTO TALLER: TRES (3)**

**OBJETIVOS**

* Concepto de vectores
* Concepto de captura
* Distinguir mediante ejercicios prácticos los vectores y como realizar capturas.
* Concepto de sobrecarga
* Realización de ejemplos de pasos de argumentos a la aplicación.

El siguiente documento taller nos permitirá analizar, modificar, comprender y desarrollar nuestros propios ejercicios, recuerde que para comprender mejor cada uno de los temas vistos en nuestro curso es necesario que usted también tenga la tarea de investigar.

**VECTORES**

**class texto**

**{**

**String[] primnom = {"Raul","Ana","Sandra","Diego”,"Luz Marina"};**

**String[] apellido = new String [ primnom.length];**

**void imprimir ()**

**{**

**for (int I = 0 ; i< primnom.length ; i++)**

**System.out.println(primnom [i]+ " " + apellido[i]);**

**}**

**public static void main(String variable[])**

**{**

**texto aux = new texto ();**

**aux.imprimir();**

**aux.apellido[0]="Ortiz";**

**aux.apellido[1]="Ruiz";**

**aux.apellido[2]="Muñoz";**

**aux.apellido[3]="Guatame";**

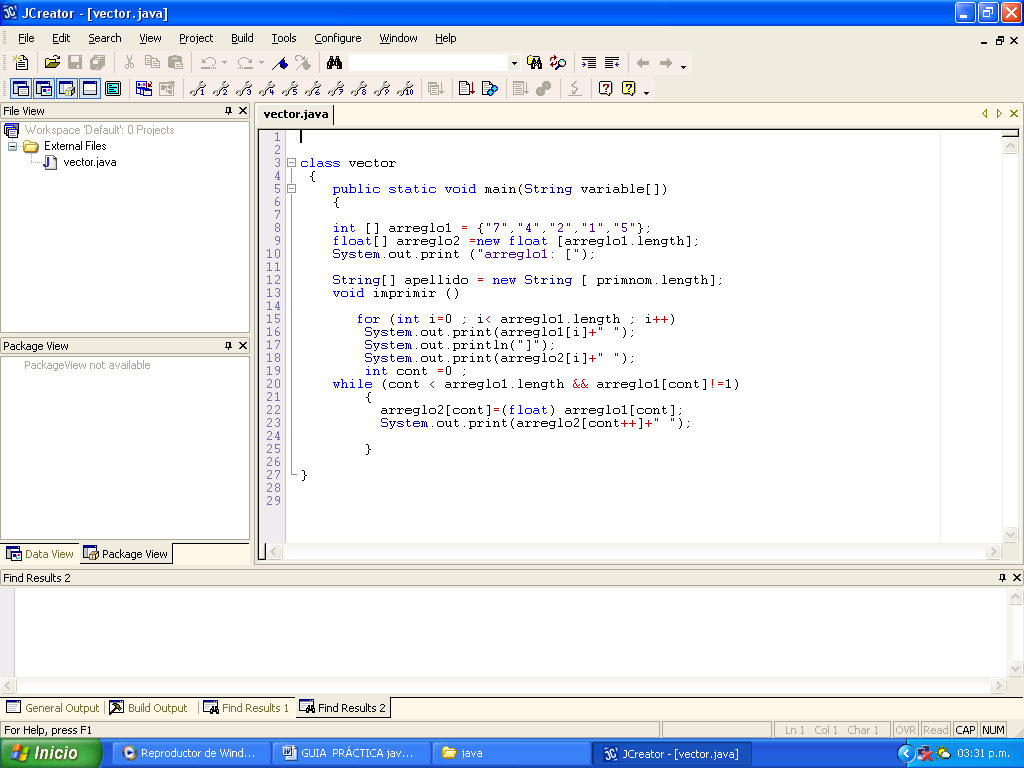
**aux.apellido[4]="Pinzón";**

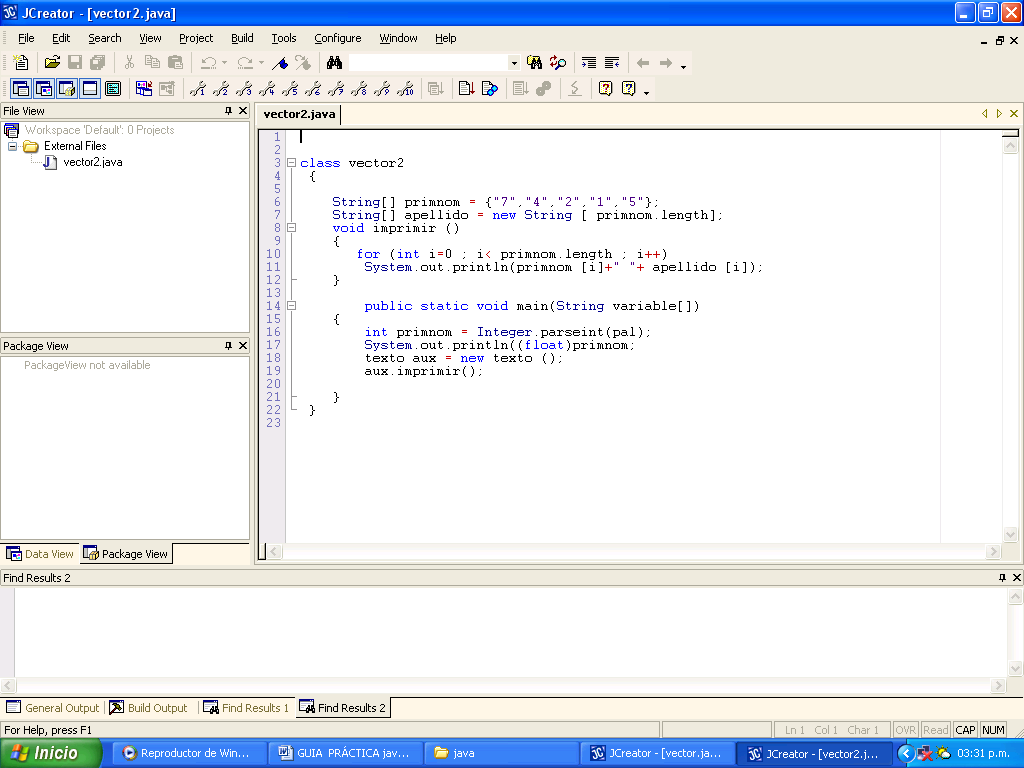
**aux.imprimir();**

**}**

**}**

**Aquí se presenta el IDE Jcreator para que desde ya te vayas acostumbrando a usarlo**





**Será que Diego es una clase?**

**INTRODUCCIÓN A LOS FLUJOS O STREAMS. COMPLETE**

Los flujos surgen por la necesidad de las aplicaciones Java de interaccionar con el exterior de dos posibles formas:

* Generando salida a la consola del Main, a un fichero, etc.
* Capturando datos procedentes del teclado, de ficheros, de páginas web, etc.

**Concepto de flujo:** es como un río. **El agua en movimiento es el flujo, su contenido son los datos. Lo que permite que esos datos viajen de un origen a un destino es el agua en movimiento, es decir, el flujo. En el caso de la captura, desde un programa Java, de datos introducidos por un usuario mediante teclado, el origen es el procesador, el destino, el programa Java y los datos, lo tecleado por el usuario.**

**Java modela flujos mediante clases del paquete java.io.**

**CAPTURA DE DATOS DESDE TECLADO**

Se necesita lo siguiente:

* **Obtener un objeto que modele la fuente de entrada de datos “teclado” en un programa Java**.

Para ello, **lo primero que debe tenerse presente es que el Teclado es la entrada estándar de datos en un PC.**

A continuación, debe tenerse en cuenta, que **cualquier entrada genérica de datos a un programa Java, se modela mediante un objeto de la clase abstracta java.io.InputStream.**

Bueno, pues en la API se observa que u**n objeto de esta clase es lo que devuelve la variable de campo estática "in" de la clase java.lang.System. Además,** la explicación API de esta variable nos dice que **representa la entrada estándar de datos de un programa Java.** Pues ya está.

|  |
| --- |
| **CONCLUSIÓN:** en los códigos que capturen datos procedentes del teclado, se empleará **Clases API para modelar el teclado**. |

Para la salida es lo mismo: **la salida estándar de datos en un PC es la consola del DOS. La salida genérica de datos de un programa Java se modela mediante un objeto de la clase abstracta java.io.OutputStream.**

En la API, se observa que **un objeto PrintStream (hereda de OutputStream y, por tanto, es un objeto OutputStream) es lo que devuelve la variable de campo estática "out" de System. Además,** la explicación API de esta variable dice que **representa la salida estándar de datos de un programa Java.**

|  |
| --- |
| **CONCLUSIÓN:** en los códigos que generen salida a través de la consola del DOS, se empleará **clases API para modelar la consola del DOS.**  ¿Cómo se escribe en la consola?  A través de métodos adecuados de PrintStream. Concretamente se han empleado print(..) y println(..) durante el tiempo que se lleva de curso |

No es esta la única forma de interaccionar con el exterior que tiene un código Java. Más adelante se verá cómo se puede generar salida creando un fichero, por ejemplo.

* **Crear un flujo que permita al programador leer datos del teclado**. Este flujo va a ser de entrada o lectura de datos y **va a modelarse con una objeto de la clase InputStreamReader**.
* **Crear un filtro encargado de leer los datos de forma óptima mediante la aplicación de  un método de lectura adecuado.** El filtro se modela con un objeto de la **clase java.io.BufferedReader y el método de lectura adecuado va a ser String readLine()** que, ya se verá, **lanza una excepción del tipo IOException que debe ser gestionada correctamente**. En un principio, se gestionará **mediante la cláusula throws IOException** a continuación del método main.

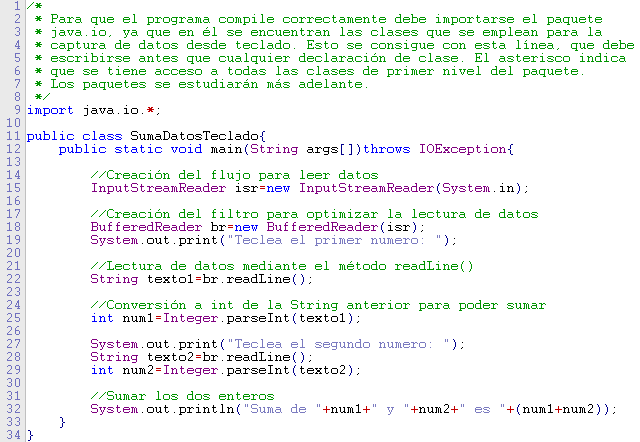
|  |
| --- |
| **CONCLUSIÓN GENERAL:**   * permite leer datos **InputStreamReader.** * El System.in u objeto InputStream es el flujo de InputStreamReader y modela el origen de los datos que se leen, es decir, el teclado. * El filtro u objeto **BufferedReader** permite, mediante la utilización del método String readLine(), leer de forma óptima del flujo. |

Vídeo explicativo del proceso de lectura de datos desde teclado.

**Ejemplo:** muestra cómo se programa un código Java que tiene que capturar datos introducidos por el usuario desde teclado.

Al ejecurlo, se mostrará por consola un mensaje indicando al usuario que introduzca un número. El usuario lo tecleará y pulsará enter. Después, se mostrará otro mensaje invitándole a que teclee otro número. El usuario lo tecleará y pulsará de nuevo enter. Finalmente, el código calculará la suma de los dos números introducidos. **Todos los códigos  en** [**c:\cursojava\tema3**](file:///c:\cursojava\tema3) **si no se emplea IDE o en jcreator\_home\MyProjects\tema3 si se emplea**

**Ejemplo:**



[Código fuente](http://www.adrformacion.com/udsimg/java/3/SumaDatosTeclado.txt)

Explica el programa.

**Por consola:**

Teclea el primer numero: 10

Teclea el segundo numero: 23

Suma de 10 y 23 es 33

|  |
| --- |
| **NOTA:** la creación del filtro podría hacerse en una sola línea de este modo:  BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); |

A continuación observara dos ejemplos de captura transcríbalos y ejecútelos, recuerde que pueden tener algún error si es así modifíquelos y completelos según lo aprendido.

**import java.lang.\*;**

**import java** **DataInputStream;**

**public class Captura {**

**public static void main(String[] var)throws IOException {**

**int num1,num2,result;**

**String cap;**

**DataInputStream dato = new DataInputStream(System.in);**

**num1=0;**

**DataInputStream dato = new DataInputStream(System.in);**

**num2=0;**

**result=0;**

**cap=" ";**

**System.out.println("digite 1er numero");**

**cap = dato.readLine();**

**num1=** Integer.parseInt(cap)**;**

**System.out.println("digite 2do numero");**

**cap = dato.readLine();**

**num2=** Integer.parseInt(cap)**;**

**System.out.println("el resultado es:");**

**result=(num1+num2);**

**System.out.println(result);**

**}**

**}**

**ejemplo2**

import java.io.\*;

public class Captura3

{

**public static void main** (String[] var)throws IOException

{

int num1, num2, result;

String Cap;

DataInputStream dato = new DataInputStream (System.in);

num1 = 0;

DataInputStream dato = new DataInputStream (System.in);

num2 = 0;

result = 0;

Cap = " ";

**System.out.println** (“digite el primer numero");

Cap=dato.readLine();

num1= Integer.parseInt(Cap);

System.out.println("digite el segundo numero");

Cap=dato.readLine();

num2= Integer.parseInt(Cap);

System.out.println("el resultado es");

result=(num1+num2);

System.out.println(result);

}

}