



# Flujo

## Sesión Conceptual 1





# Inicio

{desafío}  
latam\_



- Entender en qué consiste la programación.
- Conocer la definición de algoritmo y su importancia dentro de la programación.
- Conocer los entidades que forman parte de un diagramas de flujo.
- Conocer la relación entre pseudocódigo y diagramas de flujo.
- Conocer la importancia de la lógica independiente del lenguaje.
- Conocer la importancia del desarrollo del pensamiento lógico.

## Objetivo



# Desarrollo

{desafío}  
latam\_



# Introducción a la Programación

# ¿Que es programar?

Google

un programador es |



un programador es **un hacker**

un programador es **un desarrollador**

un programador es **un ingeniero**

es un programador **de computadora**

**el salario de** un programador es

# Programar es más que escribir código

- Analizar problemas
- Descomponerlos en partes
- Solucionar el problema
- Implementar las soluciones.



# ¿Código o algoritmos?

**Código** es la codificación en un lenguaje de programación

**Algoritmo** es la secuencia de pasos a realizarse, para solucionar el problema.

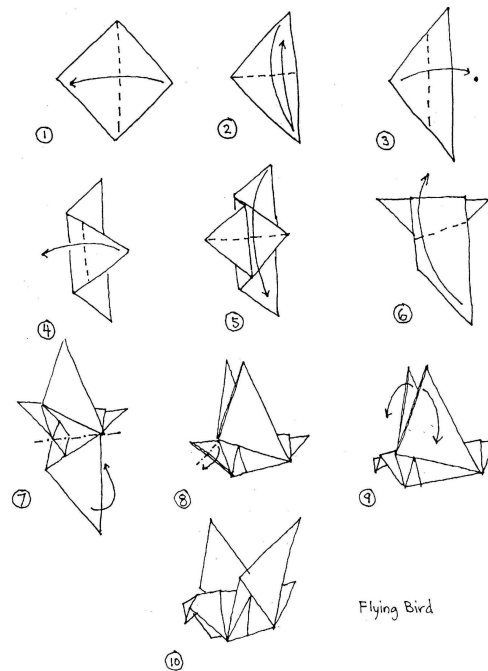
# ¿Que es un algoritmo?

Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución a un problema.

## Ejemplos de algoritmo

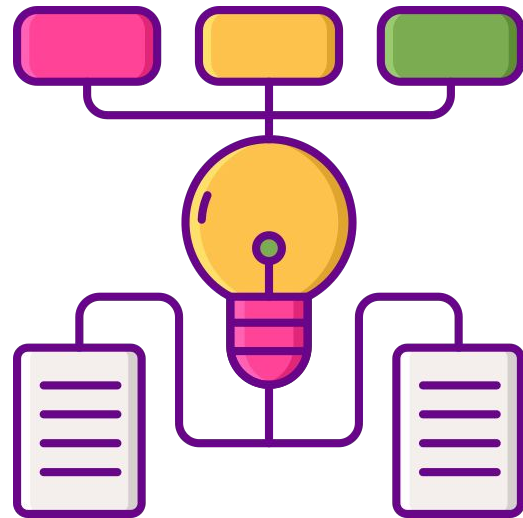
- Para ensamblar un mueble debemos seguir todos los pasos del manual de forma secuencial
- Si queremos hacer un pastel, no podemos meter al horno la harina, sin haberla mezclado antes con los otros ingredientes en un orden específico para que quede bien hecha la mezcla.
- En el caso de un modelo de origami

Una serie de pasos para resolver un problema



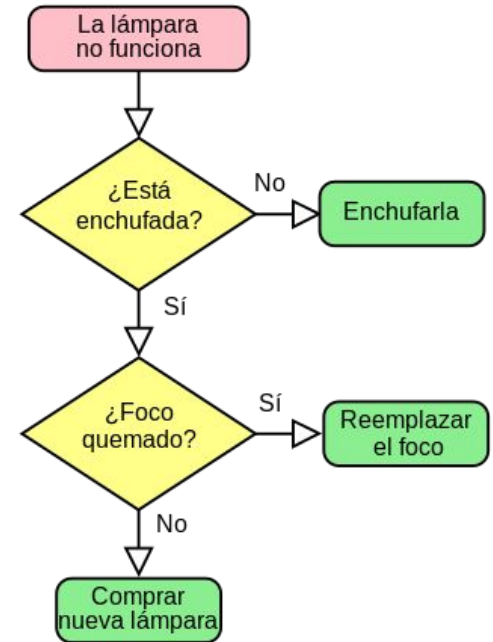
# Formas de escribir un algoritmo

- Diagrama de flujo
- Pseudocódigo
- Implementando directamente en algún lenguaje de programación.



# Símbolos de un diagrama de flujo

- Inicio y fin del proceso
- Datos de entrada y salida
- Procesos (instrucciones que se le entrega a la máquina)
- Decisiones

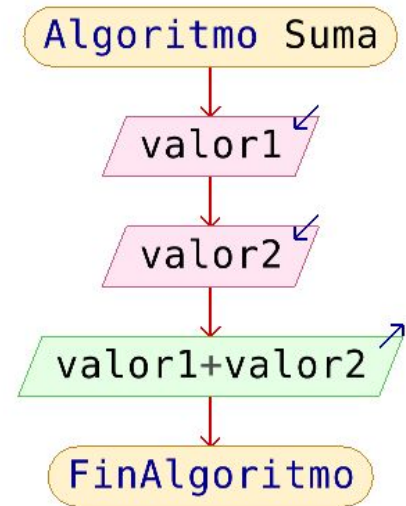


# Pseudocódigo

```
Algoritmo Suma
  Leer valor1
  Leer Valor2
  Mostrar valor1 + valor2
FinAlgoritmo
```

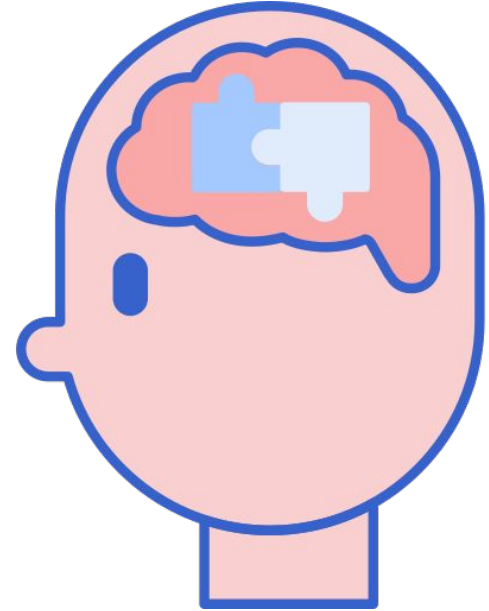
# De pseudocódigo a diagrama de flujo

- Lineal.
  - Reemplazar las instrucciones específicas por los símbolos correspondientes
  - Tener cuidado con las fechas



# La importancia de desarrollar el pensamiento lógico

- Rapidez en solucionar problemas cotidianos en programación.
- Nuestros programas harán más a menudo lo que esperamos que hagan



# Introducción a Java



## Java es el lenguaje más usado en el mundo

- 9 millones de desarrolladores,
- presente en más de 7 mil millones de dispositivos.

# Java permite

- Escribir software en una plataforma y ejecutar virtualmente en otra
- Crear programas que se puedan ejecutar en un explorador y acceder a servicios Web disponibles
- Desarrollar aplicaciones de servidor para foros en línea, almacenes, encuestas, procesamiento de formularios HTML y mucho más
- Combinar aplicaciones o servicios que utilizan el lenguaje Java para crear aplicaciones o servicios con un gran nivel de personalización
- Escribir aplicaciones potentes y eficaces para teléfonos móviles, procesadores remotos, microcontroladores, módulos inalámbricos, sensores, gateways, productos de consumo y prácticamente cualquier otro dispositivo electrónico

# ¿Por qué utilizar Java?

- Máquina Virtual de Java (**JVM**).
- Orientado a objetos, que es el paradigma que más se acerca a la manera de pensar del ser humano.
- No existen problemas con la liberación de la memoria
- Es relativamente fácil de aprender comparado con otros
- Tiene librerías estándar: Java [Api](#)
- Variedad de IDEs

# Desventajas de Java

- Al ser un lenguaje interpretado, su rendimiento suele ser un poco menor
- Solo se puede ejecutar si se posee una JVM.
- Su sintaxis compara con lenguajes como C# o Python puede ser engorrosa.

# No confundir Java JDK con Java JRE

- Java JRE: **Java Runtime Environment**
- Java JDK: **Java Development Kit**

# Instalando Java

# Instalando Java en Windows

- [Link descarga](#)
- Instalar Java
- Cambiar variables de entorno
- Comprobar correcta instalación

Abrir Símbolos del Sistema (cmd) y ejecutar el comando `java -version`, y `javac -version`

```
java -version
java version "11.0.2" 2019-01-15 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.2+9-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.2+9-LTS, mixed mode)

javac -version
javac 11.0.2
```

# Instalación en MacOS y Ubuntu

- Descargar Java desde el link anterior, e instalarlo.
- Comprobar ejecutando los comandos `java -version`, y `javac -version`





# Ventajas de utilizar un IDE

- Tienen soporte del lenguaje, agregando autocompletados, repositorios.
- Permite debugear código.
- Muestra los ficheros donde existan errores de sintaxis.
- Conoce las funciones declaradas en la clase.
- Desplazamiento ágil entre las funciones y ficheros

# Elección de IDE

- NetBeans
- Eclipse
- AndroidStudio
- BlueJ
- Otros



# Eclipse

- Para trabajar con Java utilizaremos [Eclipse](#), la versión 2021-09.

# Utilizando Java

- Aprender las formas de trabajar en java
- Ejecutar programas creados en java
- Crear un proyecto desde cero en Java utilizando Eclipse

# Objetivo

# Formas de trabajar en Java

En un IDE:

- Manera directa su compilación
- En tiempo real va corrigiendo errores de sintaxis

En terminal:

- compilación manual
- se pueden generar errores sobre la marcha

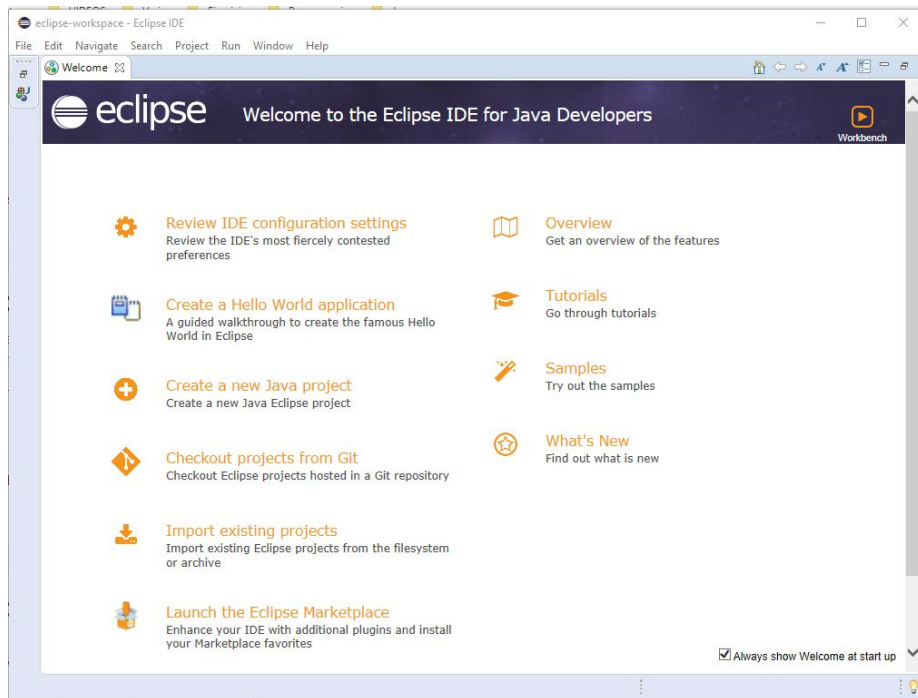
Compilando por terminal:

- Formato de archivo ***Nombre.java***.
- Para compilar: `javac Nombre.java`
- Para ejecutar: `java Nombre`



# Utilizando Eclipse

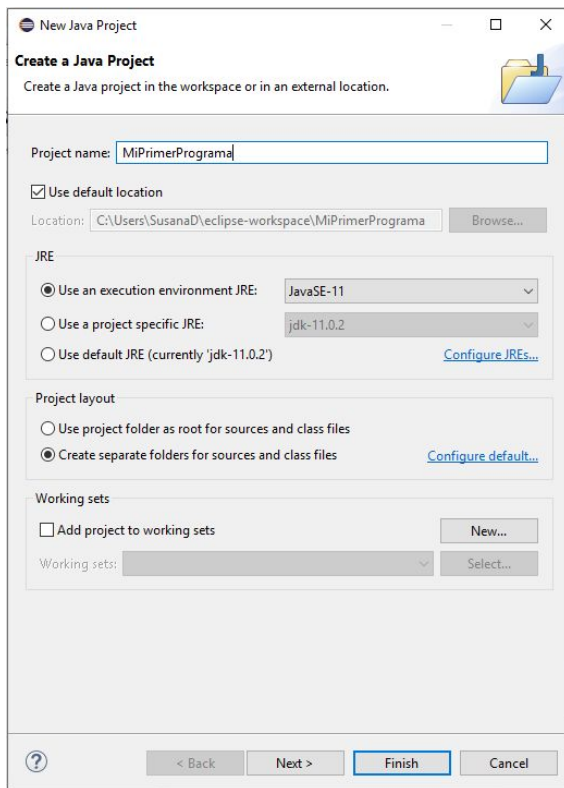
## Crear un nuevo proyecto Java



# Utilizando Eclipse

## Nuevo proyecto:

- nombre del proyecto.
- ubicación.
- versión de Java.



The screenshot shows the 'New Java Project' dialog box in Eclipse. The title bar says 'New Java Project'. Below the title bar, it says 'Create a Java Project' and 'Create a Java project in the workspace or in an external location.' There is a folder icon on the right. The dialog is divided into several sections: 'Project name' with a text field containing 'MiPrimerPrograma'; 'Use default location' with a checked checkbox and a 'Location' field showing 'C:\Users\SusanaD\eclipse-workspace\MiPrimerPrograma' and a 'Browse...' button; 'JRE' section with three radio buttons: 'Use an execution environment JRE:' (selected) with a dropdown showing 'JavaSE-11', 'Use a project specific JRE:' with a dropdown showing 'jdk-11.0.2', and 'Use default JRE (currently 'jdk-11.0.2')' with a 'Configure JREs...' link; 'Project layout' section with two radio buttons: 'Use project folder as root for sources and class files' and 'Create separate folders for sources and class files' (selected) with a 'Configure default...' link; 'Working sets' section with a checkbox 'Add project to working sets' and a 'New...' button, and a 'Working sets:' dropdown with a 'Select...' button. At the bottom, there are buttons for '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

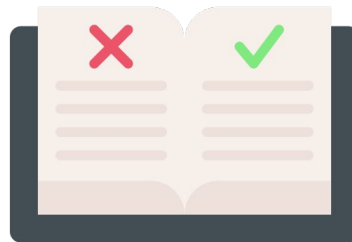


# Creando el primer programa en Java

1. Clic sobre **src** -> **New** -> Package, al cual le colocaremos el nombre **miprimerprograma**.
2. Clic sobre el package **miprimerprograma** -> new -> class, a la cual le pondremos el nombre **MiPrimerPrograma**, donde aparecerán varias opciones.

De momento momento marcaremos **public static void main(String[] args)** y finalizar.

- clase: **MiPrimerPrograma**.
- paquete: **miprimerprograma**
- [Reglas de Java](#)



# Hola Mundo

```
public class MiPrimerPrograma {  
    public static void main(String[] args) {  
        System.out.printf("Hola Mundo!\n");  
    }  
}
```



# Elementos Básicos de Java

- Definir comentarios, variables primitivas, variables de referencia a objetos, y constantes en Java
- Conocer y utilizar operaciones aritméticas en Java
- Manipular Strings utilizando concatenación en Java
- Manipulaciones más complejas con Strings en Java
- Transformación de tipos de datos
- Manipular entrada y salida de datos utilizando Scanner y System.out.printf en Java

# Objetivo

# Comentarios

```
// Esta línea es un comentario
// Los comentarios son ignorados
// Pueden ser una línea nueva
int a = 2 // 0 puede acompañar una línea de código existente
// a = 2 + 3 Si comentamos al principio todo la línea será ignorada
```

```
/* Todo el código

escrito
en estas líneas
será ignorado
*/
int i; //esta variable ya no será ignorada
```

# Introducción a las variables

- Partes de una variable
  - un nombre o identificador
  - un valor
  - tipo de dato

Declaración	Identificador	Tipo	Valor
int i;	i	entero	no asignado
int b = 2;	b	entero	2
String s;	s	referencia a string	no asignado
boolean b = false;	a	booleano	false

# Tipos de datos

Primitivos:

- int

```
int a = 2;
```

- float

```
float a = 3.5f;
```

String- referencia a objetos:

- El String básicamente es una cadena de caracteres

```
"Esto es un String"  
"hola"  
"a"
```

- En este enlace encontrarás los [Tipos de datos](#)

# Creando un String

```
System.out.println("Hola Mundo!");
```

 implícitamente un String

```
String cadena = "Hola Mundo!";
```

```
String cadena = new String("El primer programa");
```

Si queremos crear un String vacío (o nulo) existen estas dos alternativas:

```
String cadena = "";  
String cadena = new String();
```



# Declaración de un String vacío

Un string nulo es aquél que no contiene caracteres, pero es un objeto de la clase *String*.

Sin embargo, al escribir:

```
String cadena;
```

está declarando un objeto `cadena` de la clase *String*, pero aún no se ha creado ningún objeto de esta clase.

Esto quiere decir que para poder usar la variable `cadena`, deberemos usar una de las alternativas antes mencionadas.

# Operando con variables

Sumas y restas

```
int a = 4;
```

```
int b = 1;
```

```
System.out.println(a+3); //7
```

```
System.out.println(b-a); //-1
```

# Operando con variables

Multiplicaciones y divisiones

```
int a = 10;  
int b = 3;  
  
System.out.println(a*3); //30  
System.out.println(a/b); //3
```

Sobreescribiendo variables

```
int a = 10;  
a = 3;  
  
System.out.println(a); //3
```

# Operando con variables

Modificando una variable existente

```
int a = 4;  
a = a + 1;  
System.out.println(a); //5
```

Operando con variables de tipo String

```
String a = "Hola";  
String b = " Mundo";  
System.out.println(a+b);  
//Hola Mundo
```

En programación, la acción de **sumar** dos o más Strings se conoce como concatenación.

# Crear String a partir de variables

```
int edad = 34;  
String nombre = "William";  
String salida = String.format("%s  
tiene %d años.", nombre, edad);  
System.out.println(salida);  
//William tiene 34 años.
```

Otros especificadores para el formato son los siguientes

```
// int %d  
// char %c  
// float %f  
// String %s
```

[String format](#)

# Substring

Buscando un Substring

Aquí tenemos dos casos para encontrar una subcadena dentro de un String

- substring(int startIndex):
- substring(int startIndex, int endIndex):

```
String s="Paralelepipedo";  
System.out.printf("%s\n",s.substring(4)); // lelepipedo  
System.out.printf("%s\n",s.substring(0,4));// Para
```

# Operando con variables

Información del String

Podemos ver la longitud de un String

```
String cadena = "Mi primer programa";  
int longitud = cadena.length(); // 18
```

Si comienza con un determinado sufijo

```
String cadena = "Mi primer programa";  
boolean resultado = cadena.startsWith("Mi");
```

En este ejemplo la variable resulta

- Si se quiere obtener la posición de la primera ocurrencia de la letra 'p'

```
int pos = cadena.indexOf('p');
```

En caso de no existir, retornará el valor -1.

# Constantes

```
final int DIAS_SEMANA = 7;  
final int MAX_ITERACIONES = 10;
```

Convención sobre las constantes.

Estas variables deben escribirse completamente con mayúsculas, y si es más de una palabra, separadas por guión bajo.



# Entrada de datos

Entrada de datos

`Scanner`, permitirá acceder a lo que vayamos ingresando por teclado.

```
Scanner sc = new Scanner(System.in);  
String cadena = sc.nextLine();
```

pero antes, al inicio de

```
package testeandoTiposDeDatos;  
import java.util.Scanner;  
  
public class TesteandoTiposDeDatos{  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        String i = sc.nextLine();  
        System.out.println(i);  
    }  
}
```

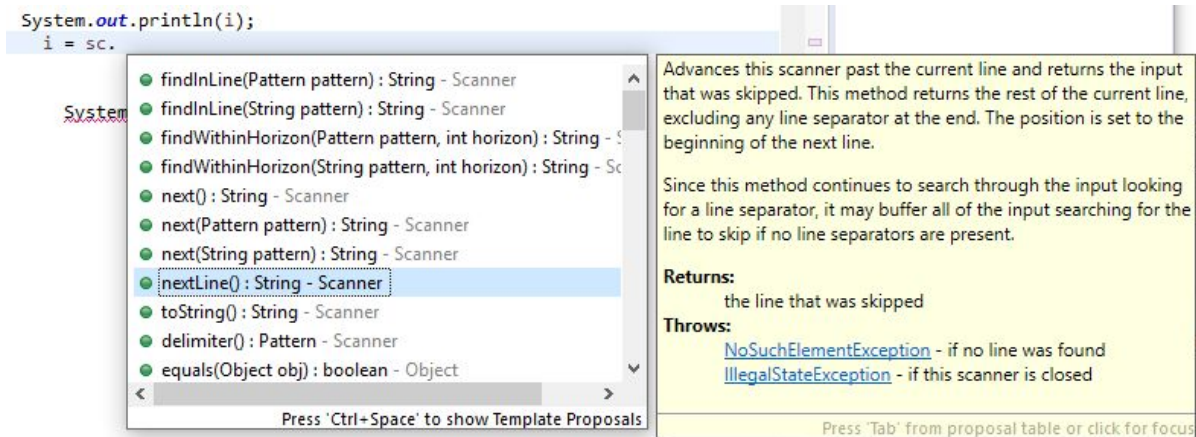
con esto obtendremos  
Flotante (`nextFloat()`)

obtener el próximo Entero (`nextInt()`),

# Operando con variables

Ventaja del IDE

Lista de sugerencias con las distintas acciones que podemos escribir.



# Operando con variables

Transformación de datos

```
float a = 8.61f;  
int b;  
b = (int)a;
```

- transformar un número a String,

```
int number = -782;  
String numeroAString = String.valueOf(number);  
String numeroAString2 = String.valueOf(-782);
```

- transformar un String a entero

```
String a = "45";  
int numero = Integer.parseInt(a);
```

# Introducción a objetos

# Objetivos

- Conocer la documentación oficial de Java
- Conocer qué es un objeto en orientación a objetos
- Utilizar métodos de los objetos creados
- Introducción a Standard Input

# Objetos y métodos

Un objeto se compone de

- campos o atributos.
- métodos o rutinas.

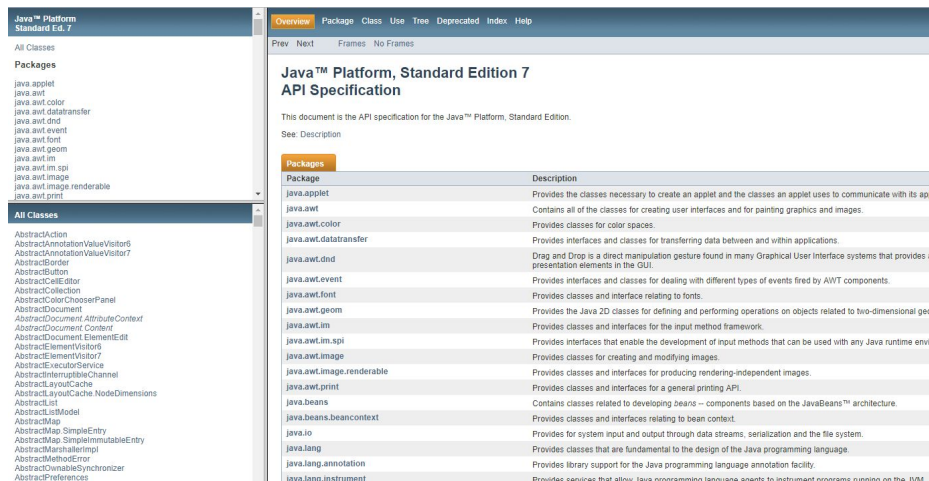
# Revisando la documentación

- Para conocer qué operadores son válidas sobre un objeto debemos leer su [documentación oficial](#)
- hábito que debemos adoptar
- Eclipse sugiere una gran cantidad de opciones.

# ¿Como se lee la documentación?

Nos enfrentaremos a 3 secciones en el link anterior

- Filtro por paquetes
- Las clases contenidas en el paquete
- La explicación de la clase seleccionada



**Java™ Platform, Standard Edition 7**

Overview Package Class Use Tree Deprecated Index Help

Prev Next Frames No Frames

### Java™ Platform, Standard Edition 7 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet viewer.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing beans – components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.instrument	Provides a package that allows Java programming language agents to instrument processes running on the JVM.



# ¿Que es una clase?

```
String cadena = new String ("Hola");
```

clase

objeto

constructor

valor

# ¿Cómo ocupamos los métodos?

```
String i= "Ver el tamaño del String";  
i.length();
```

Definición de llamar

Utilizar un método es sinónimo de llamarlo o invocarlo, el término más frecuente utilizado es el de **llamar**

# Métodos con opciones

método subString

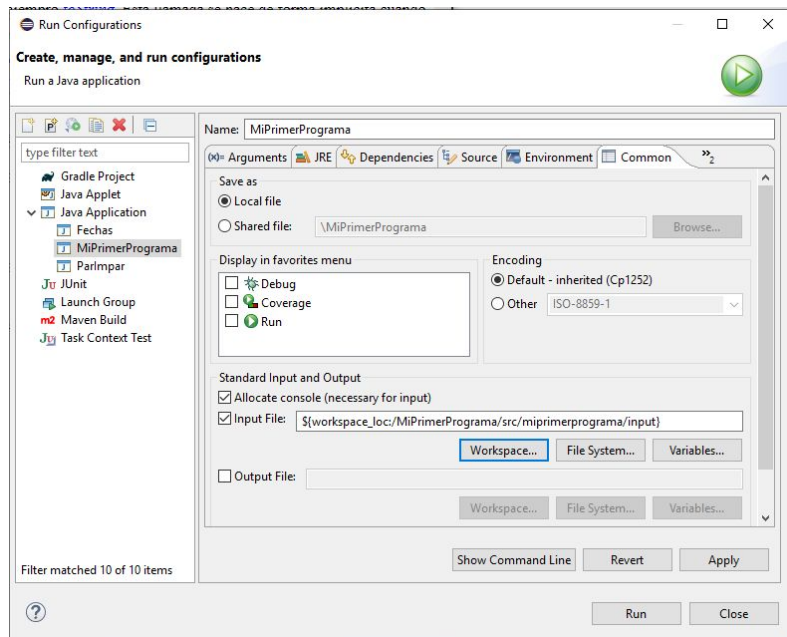
```
String s="Paralelepipedo";  
System.out.println(s.substring(4)); // lelepipedo  
System.out.println(s.substring(0,4));// Para
```

¿Y los paréntesis?

- Si no recibe ningún parámetro estos deben quedar vacío
- Si recibe uno o más parámetros, estos deben estar separados por comas

# Standard input/output

Manejo de datos de entrada y salida de manera más ordenada



```
java Nombre.java <input >output
java Nombre.java <input
java Nombre.java >output
```

# Operaciones aritméticas

# Objetivos

- Construir aplicaciones para calcular una función aritmética
- Conocer operadores aritméticos y utilizarlos
- Conocer la procedencia de los operadores aritméticos
- Utilizar paréntesis para priorizar operaciones
- Operar sobre números enteros y flotantes

# Operadores aritméticos

Operador	Nombre	Ejemplo	Resultado
+	suma	2+3	5
-	resta	2-3	-1
*	multiplicación	2*4	8
/	división	12/3	4
%	módulo o resta	5/2	1

# Operaciones con variables

```
int a = 2;  
int b = 3;  
System.out.println(a+b);
```



# Creando una calculadora

```
Scanner sc = new Scanner(System.in);  
int a = sc.nextInt(); //2  
int b = sc.nextInt(); //3  
System.out.printf("a + b es igual a %d \n", a+b);  
System.out.printf("a * b es igual a %d \n", a*b);  
  
//a + b es igual a 5  
//a * b es igual a 6
```

# Profundizando en el método System.out.printf()

```
int edad = 34;  
String nombre = "William";  
String salida = String.format("%s tiene %d años.",  
    nombre, edad);  
System.out.println(salida);
```

```
System.out.printf("a * b es igual a %d \n", a*b);
```

# Precedencia de operadores

Operador	Nombre
Math.pow(a,b)	potencia
*, /, %	Multiplicación, división, módulo
+, -	suma, resta

Cuando dos operaciones tienen el mismo nivel de prioridad, se resuelve de izquierda a derecha

# Operaciones y paréntesis

```
System.out.println((10 - 5)*2); // 10  
System.out.println(10-5*2);    //0
```

Los paréntesis si importan!

# Operaciones con números enteros y decimales

```
System.out.println(5/3); // = 1
```

## Enteros y floats

```
System.out.println(5.0f/3.0f);
```

```
float a = 5.0f;  
float b = 2.0f;  
System.out.println(a/b);
```

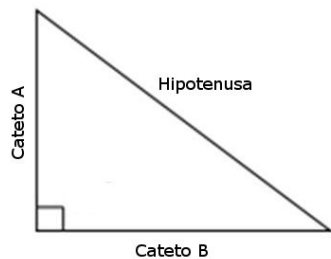
## De enteros a floats

```
int a = 1;  
int b = 2;  
System.out.println((float)a/b);
```

# Casting en Java

```
System.out.println((int) 4.5); // 4  
System.out.println((float) 4); // 4.0
```

# Ejercicio de Pitágoras



$$h = \sqrt{ca^2 + cb^2}$$

## Código

```
Scanner sc = new Scanner(System.in);  
int ca = sc.nextInt();  
int cb = sc.nextInt();  
float h = (float) Math.sqrt(ca*ca + cb*cb); //  
retorna tipo double  
System.out.println(h);
```

# Ejercicio Fahrenheit

Transformar grados Fahrenheit a Celsius usando la siguiente fórmula

$$c = (f - 32) * \frac{5}{9}$$

Código

```
int fahrenheit = sc.nextInt();  
int celcius = (fahrenheit-32)*5/9;  
System.out.printf("%d grados celcius son  
%d grados fahrenheit\n",  
celcius,fahrenheit);
```





# Quiz

{desafío}  
latam\_





Cierre

{desafío}  
latam\_



¿Hay algún contenido que aún no tengas  
la seguridad de haberlo aprendido  
totalmente?

{desafío}  
latam\_

Reflexionemos





*Academia de  
talentos digitales*

[www.desafiolatam.com](http://www.desafiolatam.com)



/DesafioLatam



/DesafioLatam



/DesafioLatam



/DesafioLatam