

Diplomado en:

Programación en Java

Guía didáctica N° 1



Formación Virtual

.....educación sin límites

GUÍA DIDÁCTICA N°1

M2-DV59-GU01

MÓDULO 1: FUNDAMENTOS BÁSICOS

DIPLOMADO EN PROGRAMACIÓN EN JAVA

© DERECHOS RESERVADOS - POLITÉCNICO DE COLOMBIA, 2018
Medellín, Colombia

Proceso: Gestión Académica

Realización del texto: Diego Palacio, Docente

Revisión del texto: - Jehison Posada, Asesor Gramatical

Diseño: Cristian Quintero, Diseñador Gráfico

Editado por el Politécnico de Colombia

ÍNDICE

PRESENTACIÓN.....	4
COMPETENCIAS.....	5
TEMA 1 Conceptos Básicos	6
TEMA 2 Variables – Tipos de Datos.....	24
TEMA 3 Operadores	37
TEMA 4 Math.....	52
ASPECTOS CLAVES	60
REFERENCIAS BIBLIOGRÁFICAS	61

PRESENTACIÓN

La Guía Didáctica N°1 del MÓDULO 1: FUNDAMENTOS BÁSICOS, es un material que ha sido desarrollado para el apoyo y orientación del participante en el *Diplomado en Programación en Java*, especialmente, está orientada a la consolidación y/o desarrollo de las habilidades y destrezas necesarias para generar unas adecuadas bases en lo que concierne a la programación en Java.

Como bien conoces, el objetivo principal de este módulo número 1 es introducir al estudiante en todo a lo referente a los fundamentos de la programación en Java, todos sus componentes y conceptos básicos para un adecuado desarrollo idóneo de las bases.

Para ello, se ha organizado esta guía cuatro (4) contenidos temáticos, entre ellos: (a) conceptos básicos, (b) variables y tipos de datos, (c) operadores y (d) Math, que ayudarán a formar unas adecuadas bases.

COMPETENCIAS

Se espera que con los temas abordados en la Guía Didáctica N°1 del MÓDULO 1: FUNDAMENTOS BÁSICOS, el estudiante logre la siguiente competencia:



- Conocer los conceptos básicos y aplicaciones del lenguaje para la programación en Java.

Indicadores de logro:

- Comprende la sintaxis básica de la estructura en la programación en Java.
- Conoce la correcta detección y corrección de errores.
- Aplica correctamente los diferentes componentes que conforman el lenguaje de programación.
- Soluciona algoritmos con las herramientas y material desarrollado en el módulo.
- Comprende el correcto funcionamiento y uso de las variables.
- Realiza operaciones con variables y parametros.
- Entiende el uso adecuado de la clase Math.

TEMA 1

Conceptos Básicos

¿Qué es programación?

La programación es el proceso por medio del cual se diseña, codifica, limpia y protege el código fuente de programas computacionales. A través de la programación se dictan los pasos a seguir para la creación del código fuente de programas informáticos. De acuerdo con ellos el código se escribe, se prueba y se perfecciona.

El objetivo de la programación es la de crear software, que después será ejecutado de manera directa por el hardware de la computadora, o a través de otro programa.

La programación se guía por una serie de reglas y un conjunto pequeño de órdenes, instrucciones y expresiones que tienden a parecerse a una lengua natural acotada. El lenguaje de programación, son todas aquellas reglas o normas, símbolos y palabras particulares empleadas para la creación de un programa y con él, ofrecerle una solución a un problema determinado.

¿Qué es un lenguaje de programación?

Un lenguaje de programación es un lenguaje formal que especifica una serie de instrucciones para que una computadora produzca diversas clases de datos. Los lenguajes de programación pueden usarse para crear programas que pongan en práctica algoritmos específicos los cuales controlan el comportamiento físico y lógico de una computadora.

Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila (de ser necesario) y se mantiene el código fuente de un programa informático se le llama programación. (Wikipedia, 2019)

¿Qué es Java?

Antes de entrar en materia de conceptos, fundamentos y código, es importante conocer, ¿Qué es Java? Java es un lenguaje de programación creado en 1995 para el entorno de computación de mismo nombre por Sun Microsystems. Desde 2009, Oracle compró Sun Microsystems, convirtiéndose en el nuevo dueño de Java. (Aboutespanol, 2018)

Java se creó para acogerse a una filosofía de cinco (5) objetivos en todo el proceso de su creación e implementación:

- Debería usar un paradigma de programación orientada a objetos.
- Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Debería incluir por defecto soporte para trabajar en red (Software en línea).
- Debería diseñarse para la ejecución de código remoto de forma segura.
- Debería ser fácil de usar y tomar lo mejor de otros lenguajes de programación orientados a objetos.

Como la mayoría de lenguajes de programación, Java se utiliza para crear aplicaciones y procesos que funcionen en multitud de dispositivos. (Wikipedia, 2018).

Java, para su operatividad y correcto funcionamiento necesita una serie de herramientas que permiten esto, dentro del stack de herramienta se encuentran principalmente:

JRE (Java Runtime Environment): Su objetivo es aportar el entorno necesario para ejecutar una aplicación Java.

JDK (Java Development Kit): Es el paquete de herramientas precisas para llevar a cabo el desarrollo de dicha aplicación.

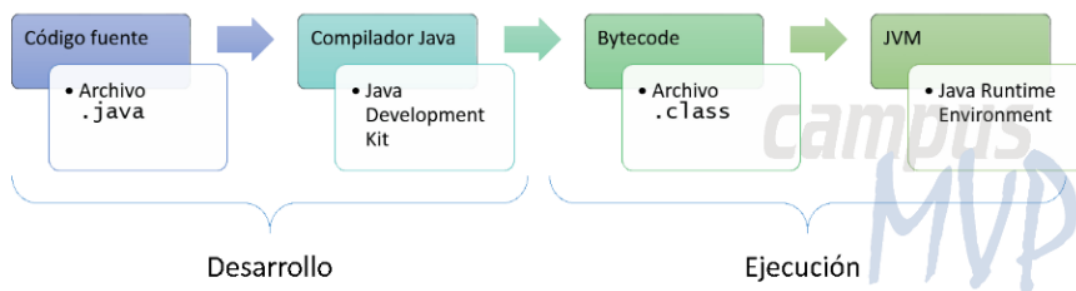


Ilustración 1: Desarrollo y Ejecución de un Programa en Java

Fuente: CampusMVP.

Recursos disponibles para el aprendizaje



Para desarrollar las habilidades y destrezas necesarias en cada competencia, es muy importante que tengas acceso a los recursos didácticos adecuados.

Entonces, si quieres ampliar la información que hemos presentado aquí, te sugerimos revisar el **Vídeo** sobre la *¿Qué es Java? Y su JVM*, disponible en: <https://www.youtube.com/watch?v=hPbvq7oe7dI>.

Ejercicio

Ahora que has revisado algunas definiciones de Java, conoces un poco su alcance, te invitamos a construir tu propia definición de **Java**, a partir de los conceptos facilitados en esta guía y que son fortalecidos con los recursos disponibles para el aprendizaje. El uso de mapas conceptuales te pueden ayudar a conocer la relación entre los elementos que conforman un concepto. ¡Inténtalo! 👍



Instalación de Java y Eclipse

Para descargar todas las herramientas necesarias para el desarrollo idóneo del diplomado y todo su material. Lo primero que se debe hacer es descargar el JDK de Java en el siguiente enlace.

- <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Hay que tener en cuenta varios factores:

- La versión del JDK.
- El sistema operativo.
- Tipo de sistema (x32 (x84) – x64).

Puede apoyarse a través del siguiente enlace:

- https://www.youtube.com/watch?v=3BNARxM_lg8

Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- [Java Developer Newsletter](#): From your Oracle account, select **Subscriptions**, expand **Technology**, and subscribe to **Java**.
- [Java Developer Day](#) hands-on workshops (free) and other events
- [Java Magazine](#)

JDK 8u191 [checksum](#)

JDK 8u192 [checksum](#)

Java SE Development Kit 8u191

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

☐ Accept License Agreement
 ☒ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.97 MB	jdk-8u191-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	jdk-8u191-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.89 MB	jdk-8u191-linux-i586.rpm
Linux x86	185.69 MB	jdk-8u191-linux-i586.tar.gz
Linux x64	167.99 MB	jdk-8u191-linux-x64.rpm
Linux x64	182.87 MB	jdk-8u191-linux-x64.tar.gz
Mac OS X x64	245.92 MB	jdk-8u191-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	133.04 MB	jdk-8u191-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.28 MB	jdk-8u191-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.04 MB	jdk-8u191-solaris-x64.tar.Z
Solaris x64	92.13 MB	jdk-8u191-solaris-x64.tar.gz
Windows x86	197.34 MB	jdk-8u191-windows-i586.exe
Windows x64	207.22 MB	jdk-8u191-windows-x64.exe

Posterior a esta descarga hay que seguir los pasos básicos de instalación. Recordar aceptar los términos y condiciones "Accept License Agreement".

Posterior a todos los pasos anteriores, Java y su JDK están correctamente instalados y funcionales. Prosigue la instalación del IDE de desarrollo Eclipse (NetBeans, Compiladores Online u otras herramientas también son válidas).

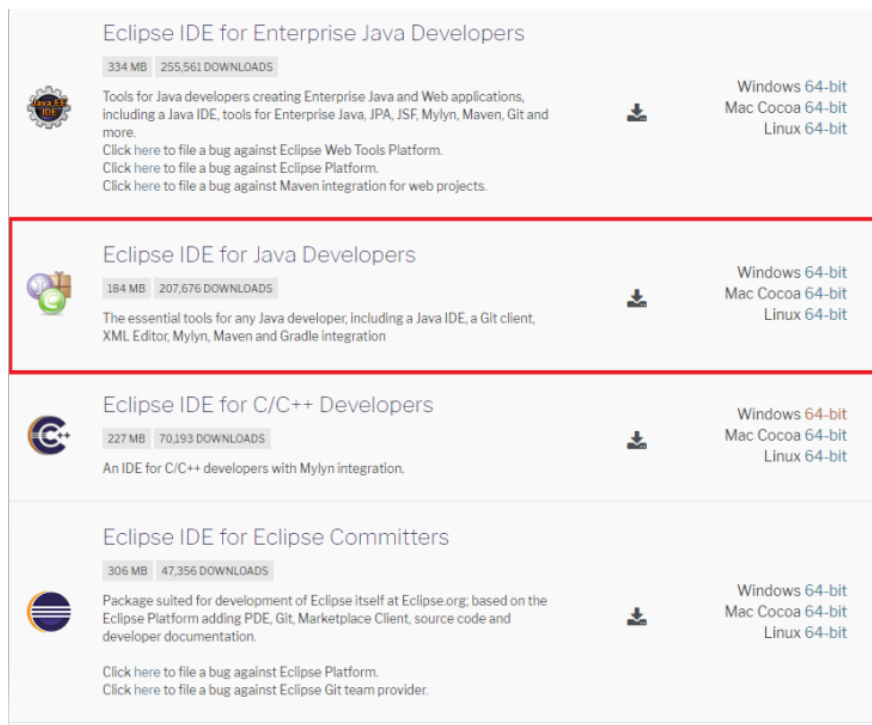
¿Qué es Eclipse?

Es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para codificar. Para proceder a la descargar, ir al siguiente enlace:

<https://www.eclipse.org/downloads/packages/>

Hay que tener en cuenta que Eclipse tiene varias distribuciones y versiones por lo que debes prestar atención en cuanto a:

- Versión del Eclipse.
- La distribución: Eclipse IDE for Java Developers.
- Tipo de sistema (x32 (x84) – x64



En este vídeo se ilustra en paso a paso de ambas instalaciones:

https://www.youtube.com/watch?v=3BNARxM_lg8

Sí presentas dificultades con la instalación, utiliza el [compilador Online](#) y escíbeme un correo.

Primer Programa: Hola Mundo

Ya que se conoce un poco lo qué es Java, se tienen instaladas las herramientas necesarias; se puede proceder a realizar un primer programa en el lenguaje de programación Java. En este caso será el famoso “Hola Mundo”. Donde adicionalmente se identificarán las primeras características que brinda el lenguaje de programación a la hora de ser empleado. Ver el siguiente vídeo para ilustrar el proceso.

[Aquí.](#)

Lo que se debe realizar son los siguientes pasos:

- Abrir el IDE de desarrollo, en este caso Eclipse, recordar que NetBeans y los editores de texto también aplican:

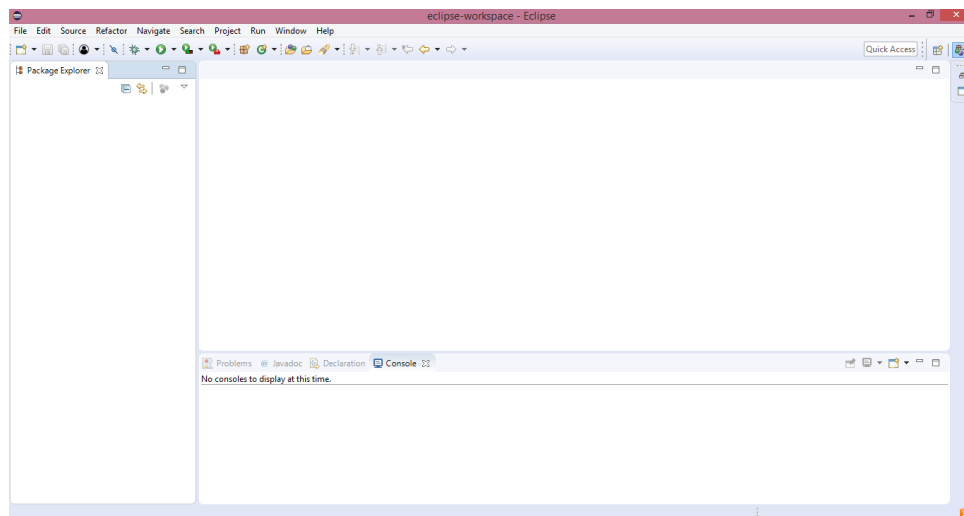


Ilustración 1: IDE Eclipse

Fuente: Eclipse.

- Se debe crear ahora un nuevo Proyecto en Java a través de las opciones proporcionadas por Eclipse en el menú:
 - File:

- New:
- Java Project:

(En caso de no encontrar la opción de Java Project ir a **Other** y de ahí a **Java Project**).

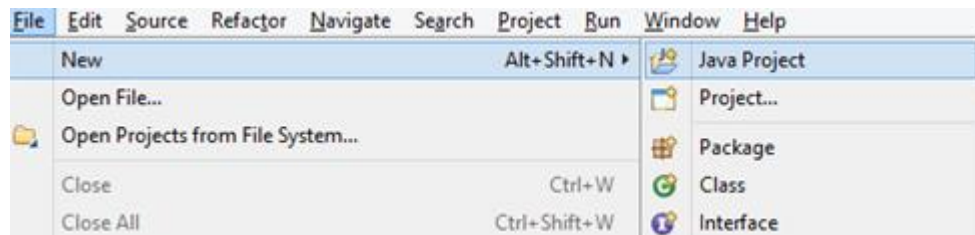


Ilustración 2: Creación de Proyectos en Eclipse.

Fuente: Eclipse.

- Ahora se debe configurar el proyecto creado (Nombre del proyecto, versión que se va a utilizar, marco del proyecto, entre otras opciones). Únicamente en este caso se establece el nombre del proyecto y la versión a utilizar:

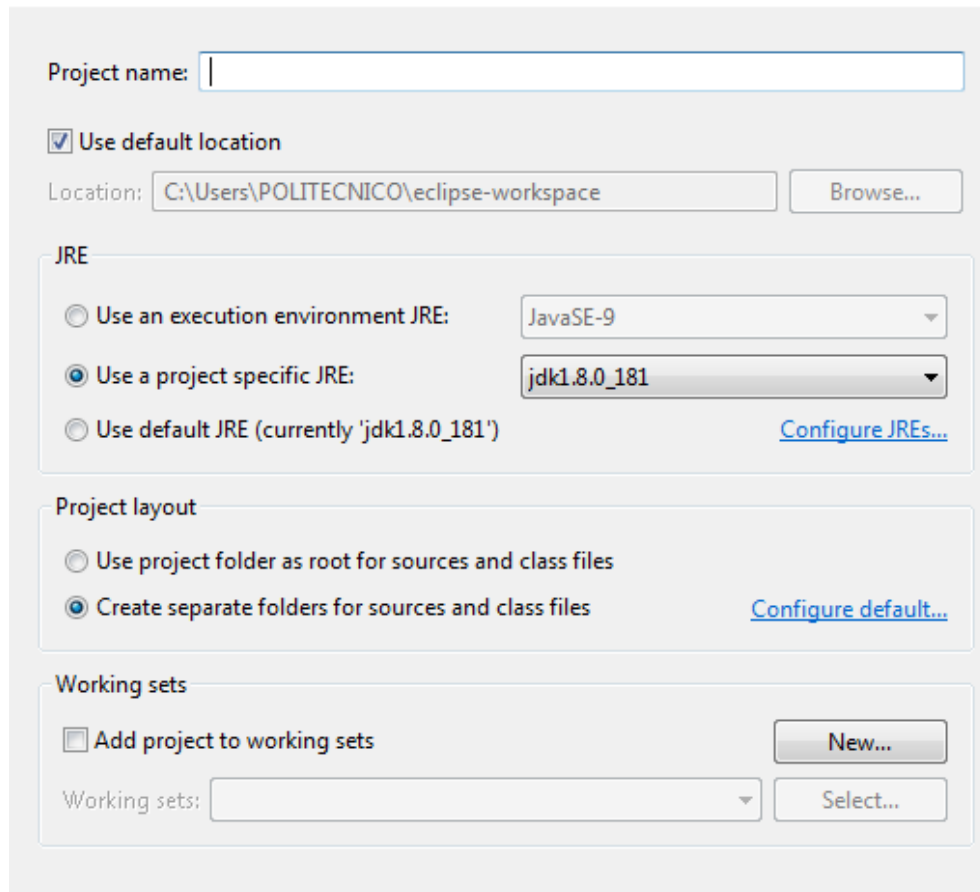


Ilustración 3: Configuración de Proyectos en Eclipse.

Fuente: Eclipse.

- El resultado será el siguiente:

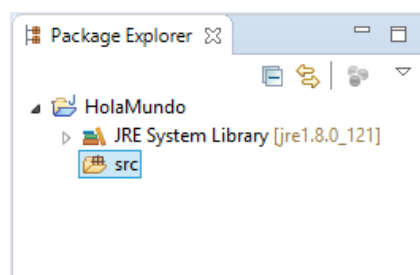
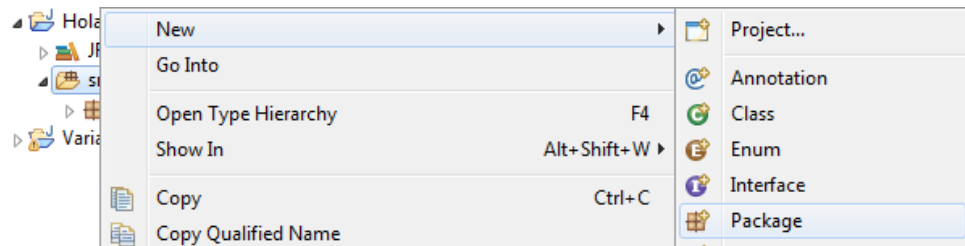


Ilustración 4: Vista previa del Proyecto en Eclipse.

Fuente: Eclipse.

- Ahora dentro de la carpeta creada por el IDE **src** (También se conoce como source) se debe crear un Paquete, el cual se encargará de contener todas las clases de Java:



- Src:
- New:
- Package:

Ilustración 5: Creación de Paquetes en Eclipse.

Fuente: Eclipse.

- Entre las configuraciones que se encuentran a la hora de crear un paquete sólo importa determinar el nombre del Paquete.

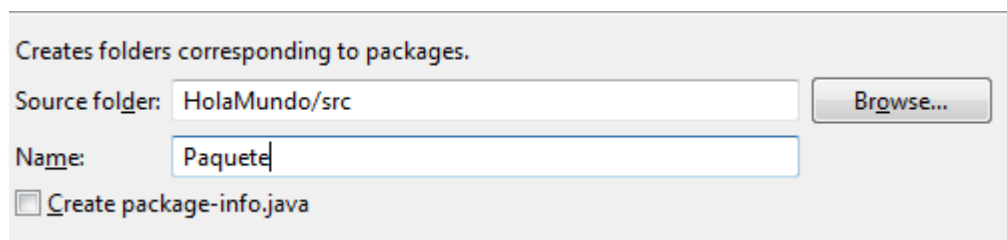
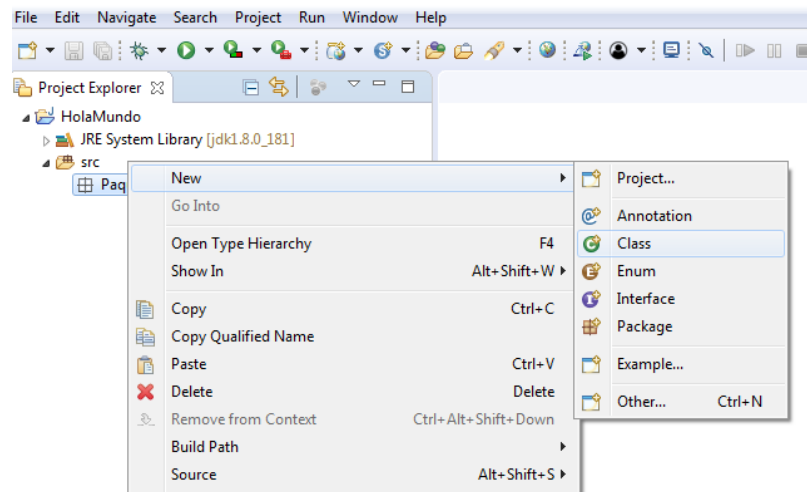


Ilustración 6: Configuración de Paquetes en Eclipse.

Fuente: Eclipse.

- Lo único que resta ahora es crear la clase Java dentro del Paquete para codificar el “Hola Mundo”.

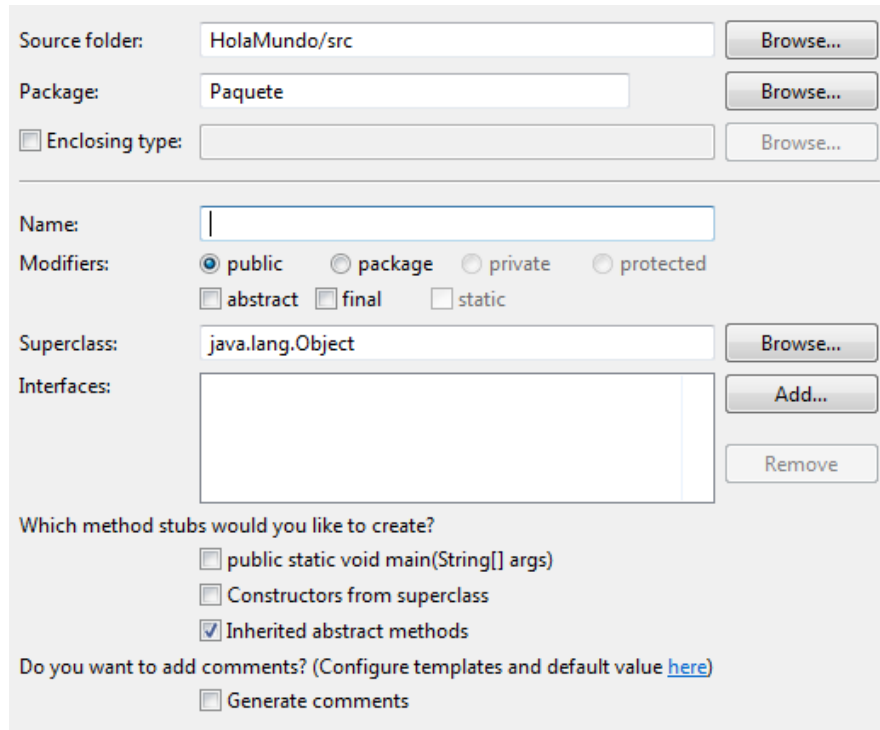


- Paquete:
- New:
- Class:

Ilustración 7: Creación de Clases en Eclipse.

Fuente: Eclipse.

- Ahora se configuran las características de la clase Hola Mundo (Nombre, Paquete, Tipo de clase, si va a ser Principal o no, entre otros).



Source folder: HolaMundo/src Browse...

Package: Paquete Browse...

☐ Enclosing type: Browse...

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Ilustración 8: Configuración de la clase HolaMundo.java

Fuente: Eclipse.

De esa forma se obtiene la siguiente estructura:

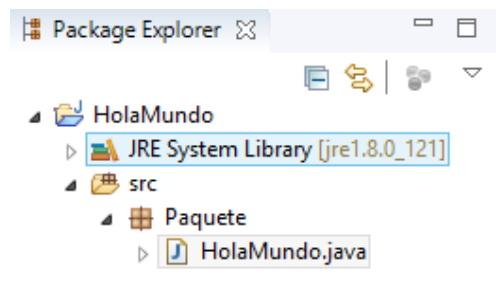


Ilustración 9: Estructura de la clase HolaMundo.java dentro del proyecto HolaMundo.

Fuente: Eclipse.

Se cuenta con un proyecto que se llama **HolaMundo** dentro de un de éste hay un source package llamado **Paquete** que contiene una clase **HolaMundo.java**; dicha clase cuenta con la siguiente estructura:

```
HolaMundo.java ✖
1 package Paquete;
2
3 public class HolaMundo
4 {
5
6 }
7
8
9
```

Ilustración 10: Clase HolaMundo.java.

Fuente: Eclipse.

Ésta es la estructura básica de una clase en Java, para el correcto funcionamiento y el desarrollo del programa “Hola Mundo”, se deben codificar las primeras líneas para obtener la siguiente estructura.

```
HolaMundo.java ✖
1 package Paquete;
2
3 public class HolaMundo
4 {
5     public static void main(String args[])
6     {
7         System.out.println("Hola Mundo");
8     }
9 }
```


`package Paquete;` Indica el paquete en el que se encuentra la clase HolaMundo.java.




`public class HolaMundo {` Indica el nombre de la clase y el nivel de acceso de ésta.

1. El nombre de la clase debe ser el mismo del archivo. Java.
2. Siempre debe contener la palabra **class**.
3. Debe abrir y cerrar llaves “{}” (Dentro de éstas va todo el código).

`public static void main(String[] args) {` Indica un método public (Publico) y main (Principal) que va a permitir ejecutar el código.

`System.out.println("Hola Mundo");` La instrucción "**System.out.println()**" permite mostrar en consola (Pantalla) el resultado que se indique dentro de los paréntesis **()**, de la siguiente forma; pero antes se debe ejecutar la clase.

El proceso de ejecutar se realiza desde una de las opciones del menú superior de la clase: , en orden se encuentran las siguientes opciones:

- Run (Ejecutar) - 
- Coverage (Cobertura) 
- Run External Tool (Ejecutar Herramienta Externa) 

Las tres (3) opciones cumplen la función de ejecutar, lo que cambia entre ellas es la forma de hacerlo.

- **Run**, ejecutar y realiza los procesos descritos en la clase.
- **Coverage**, ejecutar, realizar pruebas y consultar cobertura de aplicaciones.
- **Run External Tool**, ejecutar la clase con una herramienta externa.

Finalmente, en la parte inferior de la pantalla, en el apartado de consola se obtendrá el resultado a raíz de la ejecución de la clase. En este caso el mensaje de "Hola Mundo"

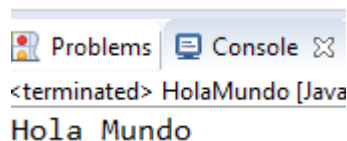


Ilustración 11: Clase HolaMundo.java ejecutada.

Fuente: Eclipse.

Otra opción que se puede usar para llevar a cabo el Hola Mundo sería la siguiente, donde le mensaje en este caso "Hola Mundo" se

encuentra contenido dentro de una variable y simplemente no se imprime el mensaje, sino la variable y el resultado será exactamente el mismo.

```
HolaMundo.java
1 package Paquete;
2
3 public class HolaMundo
4 {
5     public static void main(String args[])
6     {
7         String mensaje = "Hola Mundo";
8         System.out.println(mensaje);
9     }
10 }
11
```

Ilustración 12: Clase HolaMundo.java ejecutada manejando el mensaje por variable.
Fuente: Eclipse.

Recursos disponibles para el aprendizaje



¿Te pareció interesante el ejercicio del “Hola Mundo”? Entonces, te invitamos a que conozcas cómo se realiza en otros lenguajes de programación como: C#, Go, PHP, JavaScript, Scala, Python, entre otros en: <https://www.youtube.com/watch?v=zecueq-mo4M>.

Ejercicio

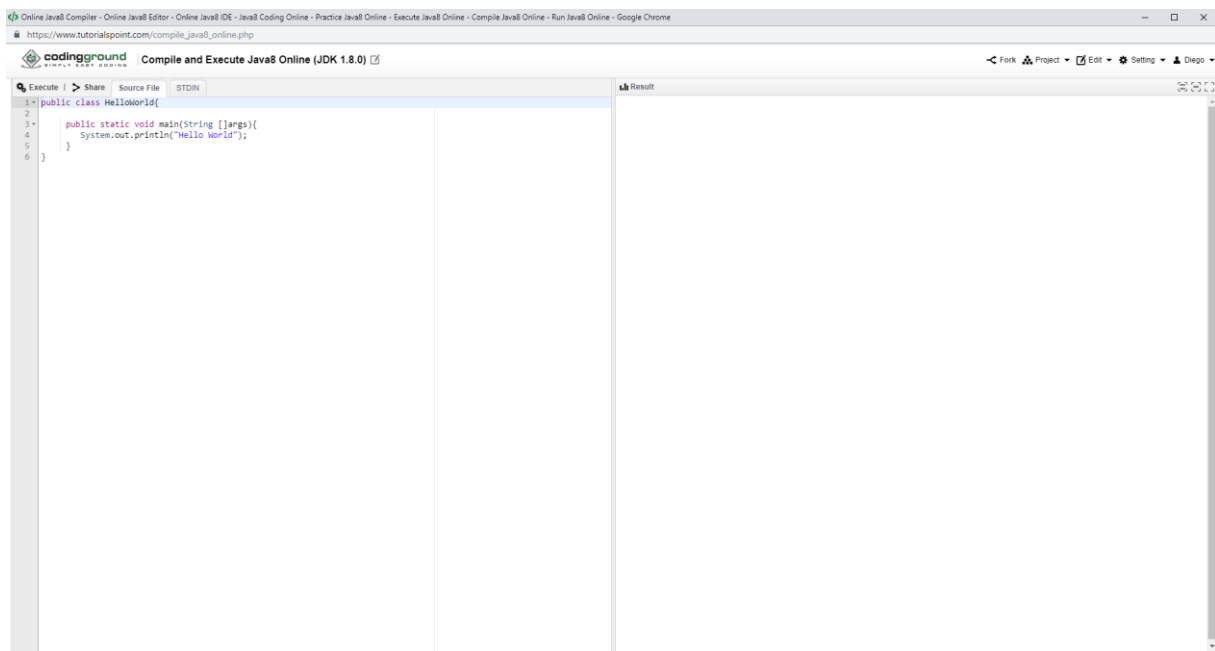
Ahora que realizaste tu primero programa en Java sobre el Hola Mundo, conociste las primeras características del lenguaje y viste cómo se realiza el ejercicio en otros 35 lenguajes, te pregunto ¿Por qué “Hola Mundo”? ¡Inténtalo! 👍



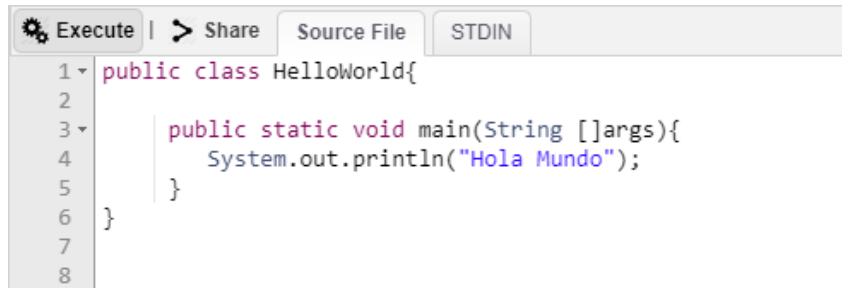
Compilador Online

Existen varias alternativas con las que se puede programar en Java, entre las más utilizadas se encuentran principalmente: NetBeans y Eclipse, ambas plataformas de escritorio, aunque son muy prácticas y efectivas, resultan un poco complicado descargar, instalar y configurar dichas herramientas, por lo que existe una alternativa muy simple y fácil de utilizar en caso de tener DIFICULTADES con la instalación de Eclipse o NetBeans hasta cierto punto, dado que es cómodo para trabajar con una clase, pero cuenta se pretende codificar más de una de estas, se convierte en mejor opción las dos principales plataformas que se mencionan al inicio. Veamos:

Puedes ingresar al compilador directamente en el siguiente hipervínculo: https://www.tutorialspoint.com/compile_java8_online.php



La plataforma ofrece una zona para codificar a mano izquierda que hace el papel de una clase, concepto que se verá a fondo más adelante:

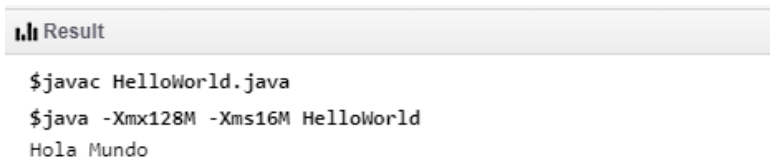


```
Execute | > Share | Source File | STDIN
1 public class HelloWorld{
2
3     public static void main(String []args){
4         System.out.println("Hola Mundo");
5     }
6 }
7
8
```

En la parte superior, se ubican dos opciones muy importantes:

- Execute: permite ejecutar el código escrito en la zona de codificación.
- Source File: permite acceder a la zona de codificación.

A mano izquierda se encuentra la zona de ejecución, donde se arrojan los datos de impresión proporcionado por la zona de codificación, por ejemplo, el hola mundo.



```
Result
$javac HelloWorld.java
$java -Xmx128M -Xms16M HelloWorld
Hola Mundo
```

Programación Estructurada

La programación estructurada es un [paradigma de programación](#) orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa de computadora recurriendo únicamente a subrutinas y tres estructuras básicas: secuencia, selección (if y switch) e iteración (bucles for y while).

A finales de los años 1970 surgió una nueva forma de programar que no solamente permitía desarrollar programas fiables y eficientes, sino que además estos estaban escritos de manera que se facilitaba su comprensión en fases de mejora posteriores.

El teorema del programa estructurado, propuesto por Böhm - Jacopini, demuestra que todo programa puede escribirse utilizando únicamente las tres instrucciones de control siguientes:

- Secuencia.
- Instrucción condicional.
- Iteración (bucle de instrucciones) con condición inicial.

Solamente con estas tres estructuras se pueden escribir todos los programas y aplicaciones posibles. Si bien los lenguajes de programación tienen un mayor repertorio de estructuras de control, estas pueden ser construidas mediante las tres básicas citadas.

TEMA 2

Variables – Tipos de Datos

Variables – Tipos de Datos

Todo programa de ordenador persigue ofrecer una funcionalidad determinada para la que, por regla general, necesitará almacenar y manipular información. Dicha información, que son los datos sobre los que se opera, deben almacenarse temporalmente en la memoria del ordenador. Para poder almacenar y recuperar fácilmente información en la memoria de un ordenador los lenguajes de programación ofrecen el concepto de variables, que no son más que nombres que "apuntan" a una determinada parte de la memoria y que el lenguaje utiliza para escribir y leer en esta de manera controlada.

El acceso a esta información se puede mejorar dependiendo del tipo de información que se almacena. Por ejemplo, no es lo mismo tener la necesidad de manejar números, que letras, que conjuntos de datos. Y dentro de éstos no es igual tener que almacenar un número entero que uno decimal. Aunque al final todo son ceros y unos dentro de la memoria de nuestro ordenador, es la forma de interpretarlos lo que marca la diferencia, tanto al almacenarlos como al recuperarlos.

Este el motivo por el que los lenguajes de programación cuentan con el concepto de tipos de datos: se trata de distintas maneras de interpretar esos "ceros y unos" en función de ciertas configuraciones que establecen el espacio utilizado, así como la representación aplicada para codificar y decodificar esa información. (Campusmvp, 2018).

Datos Primitivos en Java

Java cuenta con un pequeño conjunto de tipos de datos primitivos. Podríamos considerarlos fundamentales, ya que la mayor parte de los demás tipos, los tipos estructurados o complejos, son composiciones a partir de estos más básicos. Estos tipos de datos primitivos sirven para gestionar los tipos de información más básicos, como números de diversas clases o datos de tipo verdadero/falso (también conocidos como "valores booleanos" o simplemente "booleanos").

De estos tipos primitivos, ocho en total, seis de ellos están destinados a facilitar el trabajo con números. Se pueden agrupar en dos categorías: tipos numéricos enteros y tipos numéricos en punto flotante. Los primeros permiten operar exclusivamente con números enteros, sin parte decimal, mientras que el segundo grupo contempla también números racionales o con parte decimal.

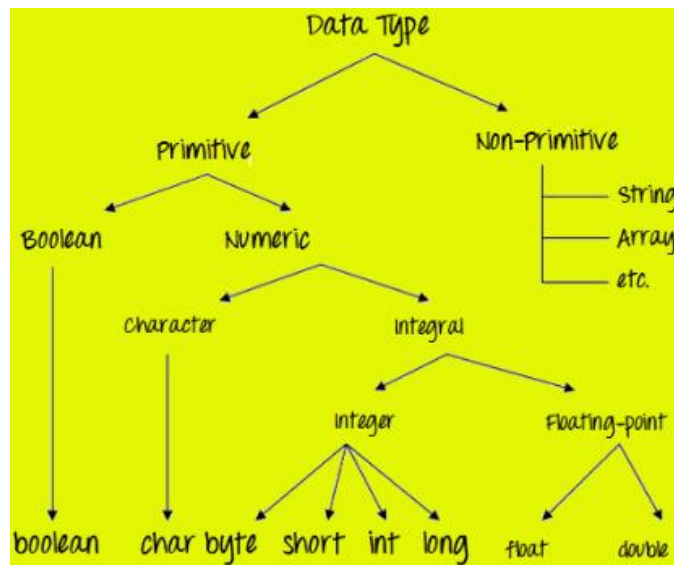


Ilustración 13: Tipos de Datos en Java

Fuente: Guru99.

Números enteros

En Java existen cuatro tipos destinados a almacenar números enteros. La única diferencia entre ellos es el número de bytes usados para su almacenamiento y, en consecuencia, el rango de valores que es posible representar con ellos. Todos ellos emplean una representación que permite el almacenamiento de números negativos y positivos. El nombre y características de estos tipos son los siguientes:

*Nota: Crear un proyecto de nombre **Variables**, que contenga un Paquete de nombre **Clases** y por ultimo una clase llamada **Variables**. Ahí se desarrollará el contenido temático – El texto que se encuentre entre “//” de color verde son comentarios que ayudan a describir lo que se está realizando en el código.*

```
//byte: Emplea un sólo byte (8 bits) de almacenamiento.  
//Esto permite almacenar valores entre [-128, 127].  
  
byte numeroByte = 9;  
  
//short: Emplea el doble almacenamiento de (byte).  
//Esto permite almacenar valores entre [-32.768, 32.767].  
  
short numeroShort = 32767;  
  
//int: Emplea un tamaño mayor, 4 bytes (32 bits).  
//Esto permite almacenar valores entre [-2147483648, 2147483647].  
  
int numeroInt = 41825;  
  
//long: Emplea el tamaño mayor de todos los enteros, 8 bytes (64 bits).  
//Esto permite almacenar valores entre [-9223372036854775808, 9223372036854775807].  
  
long numeroLong = 926465464697266565L;
```

Ilustración 14: Datos y Variables Primitivas Enteras.

Fuente: Eclipse.

Números flotantes

Los tipos numéricos en punto flotante permiten representar números tanto muy grandes como muy pequeños además de números decimales. Java dispone de 2 tipos concretos en esta categoría:

```
//float: Emplea un tamaño de 32 bits (4 bytes).  
//Esto permite almacenar valores entre [-3.4028234E+8, 1.40239846E-45].  
  
float numeroFloat = 5976464F;  
  
//double: Emplea un tamaño de 64 bits (8 bytes).  
//Esto permite almacenar valores entre [-1.7976931348623157E+309, 4.94065645841246544E-324]  
  
double numerDouble = 2654792142478F;
```

Ilustración 15: Datos y Variables Flotantes.

Fuente: Eclipse.

Booleanos y Caracteres

Aparte de los 6 tipos de datos que se acaban de ver, destinados a trabajar con números en distintos rangos, Java define otros dos tipos primitivos más:

```
//boolean: Se emplea con la finalidad de trabajar con valores verdaderos/falsos (booleanos).  
//Se traducen sus valores en true/falso.  
  
boolean variableBoolean = true;  
  
//char: Se emplea para almacenar caracteres individuales (letras, aunque puede contener números).  
//Utiliza 16 bits y se codifica sobre UTF-16 Unicode.  
  
char numeroChar = 1;  
char letraChar = 'D';
```

Ilustración 16: Datos y Variables Primitivas Booleanas - Char.

Fuente: Eclipse.

Datos Estructurados en Java

Los tipos de datos primitivos se caracterizan por poder almacenar un único valor. Salvo este reducido conjunto de tipos de datos primitivos, que facilitan el trabajo con números, caracteres y valores booleanos, todos los demás tipos de Java son objetos, también llamados tipos de datos estructurados o "clases".

Los tipos de datos estructurados se denominan así porque en su mayor parte están destinados a contener múltiples valores de tipos más simples. También se les llama muchas veces "tipos objeto" porque se usan para representar objetos.

Cadenas de Caracteres

Aunque las cadenas de caracteres no son un tipo simple en Java, sino una instancia de la clase String, el lenguaje otorga un tratamiento bastante especial a este tipo de dato, lo cual provoca que, en ocasiones, nos parezca estar trabajando con un tipo primitivo.

Aunque cuando se declare una cadena estamos creando un objeto, su declaración no se diferencia de la de una variable de tipo primitivo:

```
//String: Se emplea creando una instancia de la clase String,  
//aunque parezca trabajar con datos primitivos.
```

```
String variableString = "Hola a todos.";
```

Ilustración 17: Datos y Variables Estructuradas - String.

Fuente: Eclipse.

Las cadenas de caracteres se delimitan entre comillas dobles, en lugar de simples como los caracteres individuales. En la declaración, sin embargo, no se indica explícitamente que se quiere crear un nuevo objeto de tipo String, esto es algo que infiere automáticamente el compilador.

Las cadenas, por tanto, son objetos que disponen de métodos que permiten operar sobre la información almacenada en dicha cadena. Así, se encuentran métodos para buscar una subcadena dentro de la cadena, sustituirla por otra, dividirla en varias cadenas atendiendo a un cierto separador, convertir a mayúsculas o minúsculas, etc.

Vectores – Arrays

Los vectores son una estructura de datos que permite almacenar un grupo de datos de un mismo tipo. También son conocidos popularmente como arrays.

*También es habitual llamar **matrices** a los vectores*

Que trabajan con dos dimensiones.

Los elementos de un vector o array se empiezan a numerar en el 0, y permiten gestionar desde una sola variable múltiples datos del mismo tipo.

Por ejemplo, si se debe almacenar una lista de 10 números enteros, declararíamos un vector de tamaño 10 y de tipo entero, y no se declararían 10 variables separadas de tipo entero, una para cada número.

```
//Vector - Arreglo: Se emplea para almacenar
//un grupo de datos del mismo tipo.

//Forma 1:
int vectorNumeros1[] = new int[10];
//Forma 2:
int []vectorNumeros2 = new int[10];
//Forma 3:
int[] vectorNumeros3 = new int[10];
//Forma 4:
int vectorNumeros4[];
//Forma 5:
int vectorNumeros5[] = {};
//Forma 6:
int vectorNumeros6[] = {9,8,7,6,5,4,3,2,1,0};
//Forma 7:
int vectorNumeros7[] = new int[]{9,8,7,6,5,4,3,2,1,0};
```

Ilustración 18: Datos y Variables Estructuradas - Vectores.

Fuente: Eclipse.

```
//Matriz: Se emplea para almacenar un grupo de datos del mismo  
//de de forma bidimensional basados en [x],[y]
```

```
//Forma 1:  
int matrizNumeros1[][] = new int[4][5];  
//Forma 2:  
int [][]matrizNumeros2 = new int[4][5];  
//Forma 3:  
int[][] matrizNumeros3 = new int[4][5];  
//Forma 4:  
int matrizNumeros4[][];  
//Forma 5:  
int matrizNumeros5[][] = {};  
//Forma 6:  
int matrizNumeros6[][] = {{1,2},{3,9}};  
//Forma 7:  
int matrizNumeros7[][] = new int[][]{{6,2},{2,7}};
```

Ilustración 19: Datos y Variables Estructuradas - Matrices.

Fuente: Eclipse

Definidos por el Usuario

Además de los tipos estructurados básicos (cadenas y vectores) en Java existen infinidad de clases en la plataforma y de terceros, para realizar casi cualquier operación o tarea que se pueda ocurrir: leer y escribir archivos, enviar correos electrónicos, ejecutar otras aplicaciones o crear cadenas de texto más especializadas, entre un millón de operaciones más. Todas esas clases son tipos estructurados también.

Y por supuesto se pueden crear clases propias para hacer todo tipo de tareas o almacenar información. Serían tipos estructurados definidos por el usuario.

```
//Variable de tipo persona (Persona es una clase);  
Persona P;  
//Variable de tipo animal (Animal es una clase);  
Animal A;  
//Variable de Java de la clase Math.  
Math M;  
//Variable de Java de la clase Scanner.  
Scanner S;  
//Variable de Java de la clase BufferedReader  
BufferedReader B;
```

Ilustración 20: Datos y Variables Estructuradas – Definidas por el usuario.

Fuente: Eclipse.

Wrappers

Java cuenta con tipos de datos estructurados equivalentes a cada uno de los tipos primitivos.

Además, otra de las finalidades de estos tipos "envoltorio" es facilitar el uso de esta clase de valores allí donde se espera un dato por referencia (un objeto) en lugar de un dato por valor.

Así, por ejemplo, para representar un entero de 32 bits (int), Java define una clase llamada Integer que representa y "envuelve" al mismo dato pero le añade ciertos métodos y propiedades útiles por encima.

Estos tipos equivalentes a los primitivos pero en forma de objetos son:

<input type="checkbox"/>		<input type="checkbox"/>	
<input type="checkbox"/>	Byte	<input type="checkbox"/>	Integer
<input type="checkbox"/>	Long	<input type="checkbox"/>	Double
<input type="checkbox"/>	Float	<input type="checkbox"/>	Boolean
<input type="checkbox"/>	Short	<input type="checkbox"/>	Character

//Estos tipos son equivalentes a los primitivos pero en forma
//de objetos son: Byte, Short, Integer, Long, Float, Double,
//Boolean y Character (8 igualmente).

```
//Representación de byte en Byte.
Byte numeroByte = 1;
//Representación de short en Short.
Short numeroShort = 2416;
//Representación de int en Integer.
Integer numeroInteger = 95256712;
//Representación de long en Long.
Long numeroLong = 5213714121L;
//Representación de float en Float.
Float numeroFloat = 6591342543251F;
//Representación de double en Double.
Double numeroDouble = 9.3;
//Representación de boolean en Boolean.
Boolean variableBoolean = true;
//Representación de char en Character.
Character variableCharacter = 'A';
Character numeroCharacter = 2;
```

Ilustración 21: Datos y Variables Estructuradas – Wrappers.

Fuente: Eclipse.

Ya se conoce un poco las estructuras de las variables dependiendo de su tipo y de su función, hay características que se deben tener en cuenta al momento de utilizar las variables dentro del lenguaje:

- No pueden ser una palabra reservada del lenguaje o un literal booleano.

```
String class "Hola";  
int true = 87;  
int double = 9;
```

- No puede comenzar con un número.

```
int 5numero;  
int 5 = 9;
```

- No debe contener los símbolos que se utilicen como operadores.

```
char genero-estudiante = 'M';  
double /altura = 1.80;  
String %nombre = "Alejandro";  
int edad*docente;
```

- No debe contener espacios entre el nombre.

```
String edad estudiante = 19;  
int numero celular;
```

- Por convención los nombres compuestos deben ir con el primer carácter inicial minúsculo, el resto de los caracteres iniciales mayúsculos.

```
String nombreDocente = "Andrés";  
boolean esEstudiante = true;
```

- Se pueden crear dos variables del mismo tipo contiguamente.

```
int a = 8, b = 9;  
String j, h;
```

- Se pueden o no iniciar variables.

```
String dia = "Martes";  
float peso;  
Character generoEstudiante = 'M';  
String fecha;
```

- Se puede asignar el valor de una variable a una nueva variable.

```
int numerosA[] = {3,9,2,4,8};  
int numerosB[] = numerosA;  
String nombrePerro = "Doggy";  
String nombreGato = nombrePerro;
```

- Se puede castear el valor de las variables, es decir pasar de int a double, de String a int, entre otras combinaciones.

```
String edadTexto = "19";  
int edadNumero = Integer.parseInt(edadTexto);  
String alturaDouble = "182";  
double alturaEntera = Double.parseDouble(alturaDouble);  
  
double numeroDouble = 9.2;  
int numeroEntero = (int) numeroDouble;  
  
int numeroEntero = 9;  
double numeroDouble = (double) numeroEntero;
```

A continuación, un resume de todo lo referente a variables y tipos de datos de una forma más gráfica y ordenada:

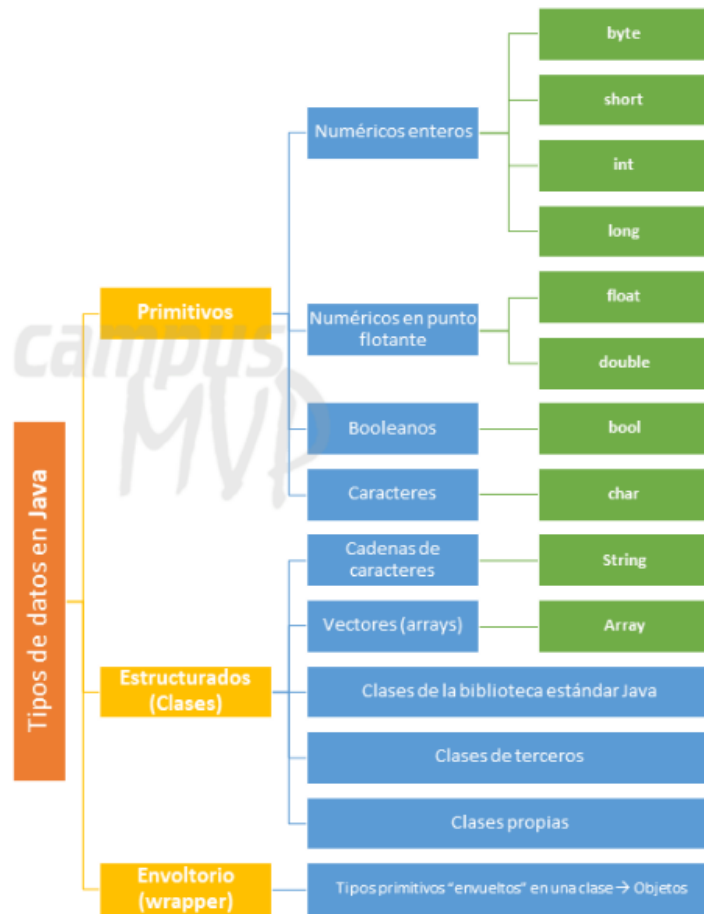


Ilustración 22: Variables y tipos de datos en Java.

Fuente: Campusvpm.

Constantes

Una constante desde el punto de vista de la programación es un dato cuyo valor no cambia durante la ejecución del programa, en otras palabras, una vez que a una constante se le asigna un valor, este no podrá ser modificado y permanecerá así durante toda la ejecución del programa.

Las constantes son útiles para datos o atributos para los cuales el valor no tiene por qué cambiar. Con esto se puede evitar modificaciones en nuestro sistema que puedan causar problemas durante la ejecución del mismo.

La palabra clave que java ha reservado para definir constantes es la palabra "*final*". En java es muy simple definir constantes, solo basta con adicionar el modificador "*final*" antes de la declaración del tipo.

```
final int documento = 921474159;
final char letraA = 'A', letraB = 'B', letraC = 'C';
final String acronimo = "CC";
final float pi;
pi = 3.1415F;
```

Hay características que se deben tener en cuenta con las constantes:

- Se pueden declarar N cantidad de constantes, siempre y cuando sean necesarias.
- **SIEMPRE** debe estar presente la palabra final.
- Se puede o No inicializar la constante en su creación.
- El valor que se asigne no puede ser modificado en ejecución.

Ejercicio

¿Deseas profundizar en la temática de Tipos de Datos y Variables?. Entonces, te sugerimos realizar los siguientes ejercicios que pondrán a prueba los conocimientos adquiridos. ¡Inténtalo! 👍 (MÓDULO 1 – EJERCICIOS CON VARIABLES)



TEMA 3 Operadores

Operadores

Un operador lleva a cabo operaciones sobre uno (*operador unario*), dos (*operador binario*) o tres (*operador ternario*) datos u operandos de tipo primitivo devolviendo un valor determinado también de un tipo primitivo. El tipo de valor devuelto tras la evaluación depende del operador y del tipo de los operandos.

Por ejemplo, los operadores aritméticos trabajan con operandos numéricos, llevan a cabo operaciones aritméticas básicas y devuelven el valor numérico correspondiente.

Los operadores se pueden clasificar en distintos grupos según se muestra en los siguientes apartados. (Arkaitzgarro, 2018).

Operadores de asignación

El operador asignación (=), es un operador binario que asigna el valor del término de la derecha al operando de la izquierda. El operando de la izquierda suele ser el identificador de una variable. El término de la derecha es, en general, una expresión de un tipo de dato compatible; en particular puede ser una constante u otra variable. Como caso particular, y a diferencia de los demás operadores, este operador no se evalúa devolviendo un determinado valor.

```
int numero1 = 8;
```

Se crea una variable (**numero1**) entera (**int**) y se asigna el valor 8.

Operador	Símbolo	Ejemplo	Resultado
Asignación	=	numero1 = 8	numero1 vale 8

Tabla 1: Operadores de asignación.

Fuente: Arkaitzgarro.

```
int numero1 = 8;  
int numero2 = numero1;
```

Se crea una variable (**numero2**) entera (**int**) y se asigna el valor de la variable (**numero1**).

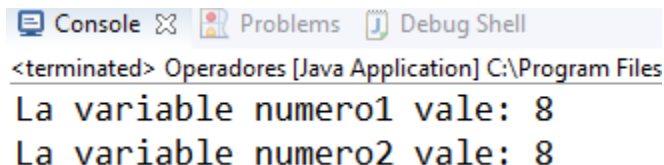
Operador	Símbolo	Ejemplo	Resultado
----------	---------	---------	-----------

Asignación	=	numero2 = numero1	Numero2 vale 8
------------	---	-------------------	-------------------

Tabla 2: Operadores de asignación.

Fuente: Arkaitzgarro.

```
System.out.println("La variable numero1 vale: "+numero1);
System.out.println("La variable numero2 vale: "+numero2);
```



Console Problems Debug Shell
<terminated> Operadores [Java Application] C:\Program Files
La variable numero1 vale: 8
La variable numero2 vale: 8

Se imprime el valor de ambas variables concatenando éstas con el "+".

Operadores aritméticos

El lenguaje de programación Java tiene varios operadores aritméticos para los datos numéricos enteros y reales. En la siguiente tabla se resumen los diferentes operadores de esta categoría.

Operador	Descripción
-	Operador unario de cambio de signo
+	Suma
-	Resta
*	Producto
/	División
%	Módulo

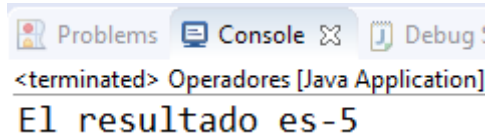
Tabla 3: Operadores aritméticos.

Fuente: Arkaitzgarro.

El resultado exacto depende de los tipos de operando involucrados. Es conveniente tener en cuenta las siguientes peculiaridades:

- El resultado es de tipo long si, al menos, uno de los operandos es de tipo long y ninguno es real (float o double).

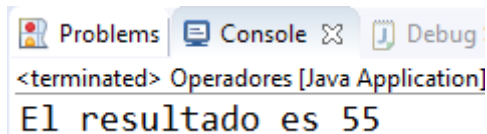
```
long numero1 = 8;  
int numero2 = 13;  
long resta = numero1-numero2;  
System.out.println("El resultado es "+resta);
```



Problems Console Debug
<terminated> Operadores [Java Application]
El resultado es-5

- El resultado es de tipo int si ninguno de los operandos es de tipo long y tampoco es real (float o double).

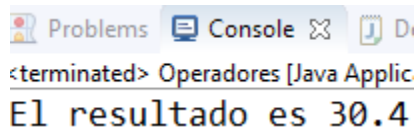
```
int numero1 = 3;  
int numero2 = 52;  
int suma = numero1+numero2;  
System.out.println("El resultado es "+suma);
```



Problems Console Debug
<terminated> Operadores [Java Application]
El resultado es 55

- El resultado es de tipo double si, al menos, uno de los operandos es de tipo double.

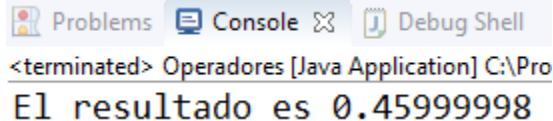
```
double numero1 = 7.6;  
int numero2 = 4;  
double multiplicacion = numero1*numero2;  
System.out.println("El resultado es "+multiplicacion);
```



Problems Console Debug
<terminated> Operadores [Java Application]
El resultado es 30.4

- El resultado es de tipo float si, al menos, uno de los operandos es de tipo float y ninguno es double.

```
float numero1 = 2.3F;  
int numero2 = 5;  
float division = numero1/numero2;  
System.out.println("El resultado es "+division);
```



Problems Console Debug Shell
<terminated> Operadores [Java Application] C:\Pro
El resultado es 0.45999998

- El formato empleado para la representación de datos enteros es el complemento a dos. En la aritmética entera no se producen nunca desbordamientos (*overflow*) aunque el resultado sobrepase el intervalo de representación (int o long).
- La división entera o módulo se trunca hacia 0. La división o el resto de dividir por cero es una operación válida que genera una excepción `ArithmeticException` que puede dar lugar a un error de ejecución y la consiguiente interrupción de la ejecución del programa.

```
double division = 15/0;  
System.out.println("El resultado es "+division);
```

Exception in thread "main" [java.lang.ArithmeticException](#): / by zero
at Operadores.Operadores.main([Operadores.java:12](#))

- La aritmética real (en coma flotante) puede desbordar al infinito (demasiado grande, *overflow*) o hacia cero (demasiado pequeño, *underflow*).

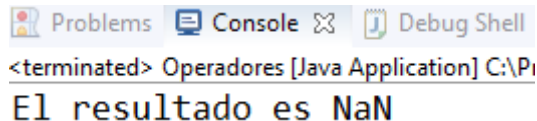
```
int numero1 = 248154379456;  
double numero2 = 2e308;  
System.out.println("El numero1 es "+numero1);  
System.out.println("El numero2 es "+numero2);
```

```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:  
The literal 248154379456 of type int is out of range  
The literal 2e308 of type double is out of range
```

```
at Operadores.Operadores.main(Operadores.java:7)
```

- El resultado de una expresión inválida, por ejemplo, dividir infinito por infinito, no genera una excepción ni un error de ejecución: es un valor NaN (Not a Number).

```
double modulo = 2.0%0;  
System.out.println("El resultado es "+modulo);
```



```
<terminated> Operadores [Java Application] C:\P...  
El resultado es NaN
```

Ejercicio práctico

1. En base a dos números enteros, realice las 5 operaciones básicas vistas hasta el momento y muestre el resultado.
2. Cree una operación utilizando números enteros y los símbolos aritméticos.

```
int numero1 = 12;  
int numero2 = 2;  
double division;  
int suma, resta, multiplicacion;  
double modulo;  
double operacion;  
  
operacion = 9/3*5-3+8*2;  
System.out.println("El resultado de la operación es: "+operacion);  
division = numero1/numero2;  
System.out.println("El resultado de la división es: "+division);  
suma = numero1+numero2;  
System.out.println("El resultado de la suma es: "+suma);  
resta = numero1-numero2;  
System.out.println("El resultado de la resta es: "+resta);  
multiplicacion = numero1*numero2;  
System.out.println("El resultado de la multiplicación es: "+multiplicacion);  
modulo = numero1%numero2;  
System.out.println("El resultado del módulo es: "+modulo);
```

```

Problems Console Debug Shell
<terminated> Operadores [Java Application] C:\Program Files\Java\jdk1.8.0_
El resultado de la operación es: 28.0
El resultado de la división es: 6.0
El resultado de la suma es: 14
El resultado de la resta es: 10
El resultado de la multiplicación es: 24
El resultado del módulo es: 0.0

```

Operadores Aritméticos incrementales


Los operadores aritméticos incrementales son operadores unarios (un único operando). El operando puede ser numérico o de tipo char y el resultado es del mismo tipo que el operando. Estos operadores pueden emplearse de dos formas dependiendo de su posición con respecto al operando.

Operador	Descripción	Ejemplo	Resultado
++A	Incrementa el valor y luego se utiliza la variable	A = 5;	A = 6;
		B = ++A;	B = 6;
A++	Utiliza la variable y luego incrementa el valor	A = 5;	A = 6;
		B = A++;	B = 5;
--A	Decrementa el valor y luego se utiliza la variable	A = 5;	A = 4;
		B = --A;	B = 4;
A--	Utiliza la variable y luego decrementa el valor	A = 5;	A = 4;
		B = A--;	B = 5;

Tabla 4: Operadores aritméticos incrementales.


Fuente: Arkaitzgarro.

```
int a = 5;
int b = ++a;
```

Problems Console 
<terminated> Operadores [Java]


```
System.out.println(a); 6
System.out.println(b); 6
```

```
int a = 5;
int b = a++;
```

Problems Console 
<terminated> Operadores [Java]


```
System.out.println(a); 6
System.out.println(b); 5
```

```
int a = 5;
int b = --a;
```

Problems Console 
<terminated> Operadores [Java]

```
System.out.println(a); 4
System.out.println(b); 4
```

```
int a = 5;
int b = a--;
```

Problems Console 
<terminated> Operadores [Java]

```
System.out.println(a); 4
System.out.println(b); 5
```

Operadores Aritméticos Combinados

Combinan un operador aritmético con el operador asignación. Como en el caso de los operadores aritméticos pueden tener operandos numéricos enteros o reales y el tipo específico de resultado numérico dependerá del tipo de éstos. En la siguiente tabla se resumen los diferentes operadores de esta categoría.

Operador	Descripción	Ejemplo	Resultado
+=	Suma combinada	a += b	a = a + b
-=	Resta combinada	a -= b	a = a - b
*=	Multiplicación combinada	a *= b	a = a * b

/=	División combinada	a /= b	a = a / b
%=	Módulo combinado	a %= b	a = a % b

Tabla 5: Operadores aritméticos combinados.

Fuente: Arkaitzgarro.

```
int a = 5;
int b = 2;
b += a;

System.out.println(b); 7
```

```
int a = 5;
int b = 2;
b -= a;

System.out.println(b); -3
```

```
int a = 5;
int b = 2;
b *= a;

System.out.println(b); 10
```

```
int a = 5;
double b = 2.0;
b /= a;

System.out.println(b); 0.4
```

```
int a = 5;
int b = 2;
b %= a;

System.out.println(b); 2
```

Operadores de relación

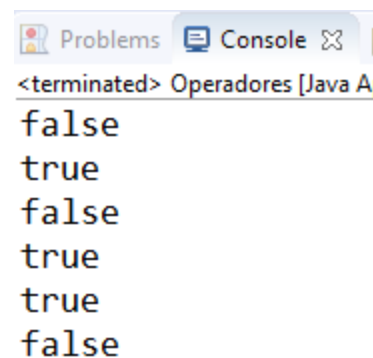
Realizan comparaciones entre datos compatibles de tipos primitivos (numéricos, carácter y booleanos) teniendo siempre un resultado booleano. Los operandos booleanos sólo pueden emplear los operadores de igualdad y desigualdad.

Operador	Descripción	Ejemplo	Resultado
==	Igual qué	5 == 4	False
!=	Diferente qué	4 != 2	True
<	Menor qué	5 < 2	False
>	Mayor qué	5 > -5	True
<=	Menor o igual qué	2 <= 3	True
>=	Mayor o igual qué	3 >= 9	False

Tabla 6: Operadores de relación.

Fuente: Arkaitzgarro.

```
boolean igualQue = 5 == 4;
System.out.println(igualQue);
boolean distintoQue = 4 != 2;
System.out.println(distintoQue);
boolean menorQue = 5 < 3;
System.out.println(menorQue);
boolean mayorQue = 5 > -5;
System.out.println(mayorQue);
boolean menorIgualQue = 2 <= 3;
System.out.println(menorIgualQue);
boolean mayorIgualQue = 3 >= 9;
System.out.println(mayorIgualQue);
```



```
<terminated> Operadores [Java A
false
true
false
true
true
false
```

Operadores lógicos

Realizan operaciones sobre datos booleanos y tienen como resultado un valor booleano. En la siguiente tabla se resumen los diferentes operadores de esta categoría.

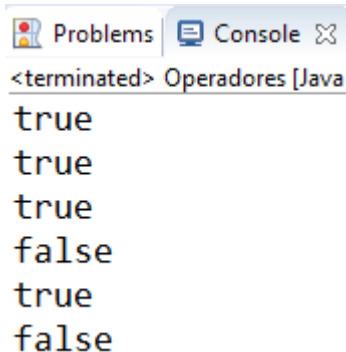
Operador	Descripción	Ejemplo	Resultado
!	Negación (Unario)	!False	True
		!(5 == 5)	False
	Suma Lógica (Binario)	True False	True

		$(5 == 5) \mid (5 < 4)$	True
^	Suma Lógica Exclusiva	$\text{True} \wedge \text{False}$	True
		$(5 == 5) \wedge (5 < 4)$	True
&	Producto Lógico (Binario)	$\text{True} \& \text{False}$	False
		$(5 == 5) \& (5 < 4)$	False
	Suma Lógica Cortocircuito	$\text{True} \mid\mid \text{False}$	True
		$(5 == 5) \mid\mid (5 < 4)$	True
&&	Producto Lógico Cortocircuito	$\text{True} \&\& \text{False}$	False
		$(5 == 5) \&\& (5 < 4)$	False

Tabla 7: Operadores lógicos.

Fuente: Arkaitzgarro.

```
boolean negacion = !false;
System.out.println(negacion);
boolean sumaLogica = true | false;
System.out.println(sumaLogica);
boolean sumaLogicaExclusiva = (5 == 5) ^ (5 < 4);
System.out.println(sumaLogicaExclusiva);
boolean productoLogico = (5 == 5) & (5 < 4);
System.out.println(productoLogico);
boolean sumaLogicaCortocircuito = true || false;
System.out.println(sumaLogicaCortocircuito);
boolean productoLogicoCortocircuito = (5 == 5) && (5 < 4);
System.out.println(productoLogicoCortocircuito);
```



Problems Console

<terminated> Operadores [Java]

```
true
true
true
false
true
false
```

Operador condicional

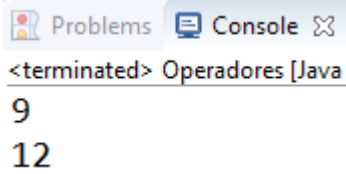
Este operador ternario permite devolver valores en función de una expresión lógica.

Operador	Descripción	Ejemplo	Resultado
?:	Operador Condicional	a = 4;	b = 9;
		b = a == 4 ? a + 5 : 6 - a;	
		b = a > 4 ? a * 7 : a + 8;	b = 12;

Tabla 8: Operador condicional.

Fuente: Arkaitzgarro.

```
int a = 4;
int b = a == 4 ? a + 5 : 6 - a;
System.out.println(b);
b = a > 4 ? a * 7 : a + 8;
System.out.println(b);
```



<terminated> Operadores [Java]
9
12

Operador de concatenación de cadenas

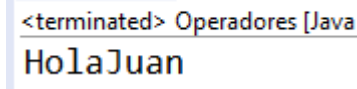
El operador concatenación `+`, es un operador binario que devuelve una cadena resultado de concatenar las dos cadenas que actúan como operandos. Si sólo uno de los operandos es de tipo cadena, el otro operando se convierte implícitamente en tipo cadena.

Operador	Descripción	Ejemplo	Resultado
+	Operador de concatenación	"Hola" + "Juan"	"HolaJuan"

Tabla 9: Operador de concatenación de cadenas.

Fuente: Arkaitzgarro.

```
String saludo = "Hola" + "Juan";
System.out.println(saludo);
```



<terminated> Operadores [Java]
HolaJuan

Operadores de separación

Existen algunos caracteres que tienen un significado especial en el lenguaje Java. En la siguiente tabla se resumen los diferentes separadores que pueden encontrarse en el código fuente de un programa.

Separador	Descripción
()	Permite modificar la prioridad de una expresión, contener expresiones para el control del flujo y realizar conversiones de tipo. Por otro lado pueden contener la lista de parámetros o argumentos, tanto en la definición de un método como en la llamada al mismo.
{ }	Permite definir bloques de código y ámbitos y contener los valores iniciales de un array.
[]	Permite declarar bloques de array (Vectores o matrices) y referenciar sus elementos
;	Permite separar sentencias
,	Permite separar identificadores consecutivos en la declaración de variables y en las listas de parámetros. También se emplea para encadenar sentencias dentro de un ciclo for.
.	Permite separar el nombre de un atributo o método de su instancia de referencia. También separa el identificador de un paquete de los subpaquetes y clases.

Tabla 10: Operadores de separación.

Fuente: Arkaitzgarro.

```
int suma = (5+9)*2;
int arreglo[] = {2, 9};
int a, b, c;
double euler = Math.E;
```

Prioridad entre operadores

Si dos operadores se encuentran en la misma expresión, el orden en el que se evalúan puede determinar el valor de la expresión. En la siguiente tabla se muestra el orden o prioridad en el que se ejecutan los operadores que se encuentren en la misma sentencia. Los operadores de la misma prioridad se evalúan de izquierda a derecha dentro de la expresión.

Prioridad	Operador	Tipo	Operación
1	++	Aritmético	Incremento previo o posterior
	--	Aritmético	Decremento previo o posterior
	+, -	Aritmético	Suma, resta
	~	Integral	Cambio de bits
	!	Booleano	Negación
2	Tipo	Cualquiera	NA
3	*, /, %	Aritmético	Multiplicación, división y residuo
4	+, -	Aritmético	Suma, Resta
	+	Cadena	Concatenación de cadenas
5	<<	Integral	Desplazamiento de bits a la izquierda
	>>	Integral	Desplazamiento de bits a la derecha con inclusión de signo
	>>>	Integral	Desplazamiento de bits a la derecha con inclusión del cero

6	<, <=	Aritmético	Menor qué, menor o igual qué
	>, >=	Aritmético	Mayor qué, mayor o igual qué
	instanceof	Objeto, tipo	Comparación de tipos
7	==	Primitivo	Igual
	!=	Primitivo	Desigual
	==	Objeto	Igual
	!=	Objeto	Desigual
8	&	Integral	Cambio de bits AND
	&	Booleano	Producto booleano
9	^	Integral	Cambio de bits XOR
	^	Booleano	Suma exclusiva booleana
10		Integral	Cambio de bits OR
		Booleano	Suma booleana
11	&&	Booleano	AND condicional
12		Booleano	OR condicional
13	? :	Booleano	Operador condicional ternario
14	=	Variable	Asignación
	*=, /=, %=		Asignación con operación
	+=, -=		
	<<=, >>=		
	>>>=		
	&=, ^=, =		

Tabla 11: Prioridad entre operadores.

Fuente: Arkaitzgarro.

Recursos disponibles para el aprendizaje



¿Muchos operadores dentro del lenguaje? ¿Has escuchado sobre la notación infija, prefija y sufija?: Te invito a que consultes un poco sobre este tema.

Ejercicio

¿Deseas profundizar en la temática de Operadores?. Entonces, te sugerimos realizar los siguientes ejercicios que pondrán a prueba los conocimientos adquiridos. ¡Inténtalo! 👍 (MÓDULO 1 – EJERCICIOS CON OPERADORES)



TEMA 4

Math

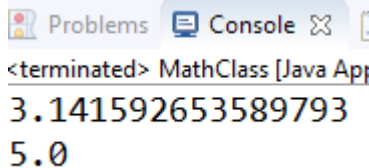
La clase Math representa la librería matemática de Java. Las funciones que contiene son las de todos los lenguajes, parece que se han metido en una clase solamente a propósito de agrupación, por eso se encapsulan en Math, y lo mismo sucede con las demás clases que

corresponden a objetos que tienen un tipo equivalente (Character, Float, etc.).

```
public class MathClass {

    public static void main(String[] args)
    {
        double valorPi = Math.PI;
        System.out.println(valorPi);

        double raiz = Math.sqrt(25);
        System.out.println(raiz);
    }
}
```



Problems Console [X]
<terminated> MathClass [Java Applet]
3.141592653589793
5.0

Método	Descripción
Math.abs(x)	Devuelve el valor absoluto de un número.
Math.acos(x)	Devuelve el arco coseno de un número.
Math.acosh(x)	Devuelve el arco coseno hiperbólico de un número.
Math.asin(x)	Devuelve el arco seno de un número.
Math.atan(x)	Devuelve el arco tangente de un número.

Math.atan2(y, x)	Devuelve el arco tangente del cociente de sus argumentos.
Math.cbrt(x)	Devuelve la raíz cúbica de un número.
Math.ceil(x)	Devuelve el entero más pequeño mayor o igual que un número.
Math.cos(x)	Devuelve el coseno de un número.
Math.cosh(x)	Devuelve el coseno hiperbólico de un número.
Math.exp(x)	Devuelve E^x , donde x es el argumento, y E es la constante de Euler (2.718...), la base de los logaritmos naturales.
Math.expm1(x)	Devuelve $e^x - 1$.
Math.floor(x)	Devuelve el mayor entero menor que o igual a un número.
Math.hypot(x, y)	Devuelve la raíz cuadrada de la suma de los cuadrados de sus argumentos.
Math.log(x)	Devuelve el logaritmo natural (log, también ln) de un número.
Math.max(a, b)	Devuelve el mayor de cero o más números.
Math.min(a, b)	Devuelve el más pequeño de cero o más números.

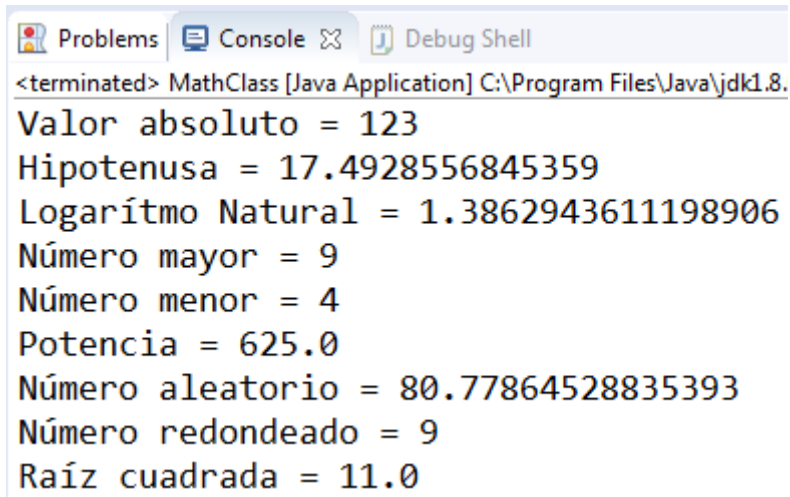
Math.pow(x, y)	Las devoluciones de base a la potencia de exponente, que es, base exponent.
Math.random()	Devuelve un número pseudo-aleatorio entre 0 y 1.
Math.round(x)	Devuelve el valor de un número redondeado al número entero más cercano.
Math.sin(x)	Devuelve el seno de un número.
Math.sinh(x)	Devuelve el seno hiperbólico de un número.
Math.sqrt(x)	Devuelve la raíz cuadrada positiva de un número.
Math.tan(x)	Devuelve la tangente de un número.
Math.tanh(x)	Devuelve la tangente hiperbólica de un número.
Math.E	Devuelve el valor de E
Math.PI	Devuelve el valor de PI

Nota: Estos son algunos de los métodos, la lista completa la podrás encontrar en: (Oracle, 2018).

Tabla 12: Metodos de la clase Math.

Fuente: Arkaitzgarro.

```
public static void main(String[] args)
{
    System.out.println("Valor absoluto = " + Math.abs(-123));
    System.out.println("Hipotenusa = " + Math.hypot(15, 9));
    System.out.println("Logaritmo Natural = " + Math.log(4));
    System.out.println("Número mayor = " + Math.max(4, 9));
    System.out.println("Número menor = " + Math.min(4, 9));
    System.out.println("Potencia = " + Math.pow(5, 4));
    System.out.println("Número aleatorio = " + Math.random()*100);
    System.out.println("Número redondeado = " + Math.round(Math.random()*10));
    System.out.println("Raíz cuadrada = " + Math.sqrt(121));
}
```



Problems Console Debug Shell
<terminated> MathClass [Java Application] C:\Program Files\Java\jdk1.8.
Valor absoluto = 123
Hipotenusa = 17.4928556845359
Logaritmo Natural = 1.3862943611198906
Número mayor = 9
Número menor = 4
Potencia = 625.0
Número aleatorio = 80.77864528835393
Número redondeado = 9
Raíz cuadrada = 11.0

Ejercicio

¿Deseas profundizar en la temática de la clase Math?. Entonces, te sugerimos realizar los siguientes ejercicios que pondrán a prueba los conocimientos adquiridos. ¡Inténtalo! 👍 (MÓDULO 1 – EJERCICIOS CLASE MATH)



Palabras Reservadas del Lenguaje

Las palabras reservadas, como su nombre lo indican, son palabras las cuales el lenguaje de programación ya ha reservado para realizar ciertas tareas, por lo que no pueden ser usadas para otras. En Java actualmente existen 50 palabras reservadas.

Palabras Reservadas				
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Algo importante a tener en cuenta es que las palabras reservadas no pueden ser utilizadas como nombres de variables, clases o métodos. En adición a este listado se puede agregar true, false y null.

Aunque const y goto son palabras reservadas, estas no son utilizadas en la actualidad.

Recursos disponibles para el aprendizaje



Para desarrollar las habilidades y destrezas necesarias en cada competencia, es muy importante que tengas acceso a los recursos didácticos adecuados.

Entonces, si necesitas reforzar esta información, te sugerimos revisar nuevamente los **Vídeos de Apoyos**, disponibles en el campus virtual, indicados en las lecturas anteriores. Además, recuerda que puede consultar las **Fuentes Documentales** que aparecen en esta guía, particularmente, en el apartado de Referencias Bibliográficas.

MATERIAL COMPLEMENTARIO

Es importante continuar adquiriendo conocimiento y no frenar el proceso de aprendizaje, por esto es importante complementar lo aprendido en esta guía con nuevos conceptos, definiciones y característica, sugerimos revisar el siguiente material:

Eres especial, ¿No lo sabías?, son pocas las personas que se involucran directamente con el mundo de la tecnología, conoce ¿Por qué todo el mundo debería saber programar?. Disponible en: <https://www.youtube.com/watch?v=X5Wkp1gsNik>

¿Qué es Java y cómo funciona?. Es fundamental consolidar una excelente y completa definición, por eso es importante conocer a fondo las características y funciones que lo complementan como lenguaje. Disponible en: <https://www.youtube.com/watch?v=L4pJhLXyOw>

¿Qué es un algoritmo?. En todo lenguaje de programación y prácticamente en cualquier actividad relacionada con computadores es importante conocer ¿Qué es un algoritmo? Esto nos ayudará a entender cómo funcionan todas las tareas dentro de un computador. Disponible en: <https://www.youtube.com/watch?v=U3CGMyjzlvM>

ASPECTOS CLAVES

Recuerda tener muy presente los conceptos visto en esta guía número 1 dado que en el transcurso del diplomado se tendrán en cuenta continuamente para su implementación.

Recuerda algunos aspectos abordados en el módulo:

- Tener muy presente los tipos de datos al momento de declarar todas las variables.
- Java es muy sensible, ten en cuenta las mayúsculas, números y símbolos especiales en todo momento, en tus proyectos, clases, variables y demás.
- Las palabras reservadas son de vital importancia en el lenguaje, recuerda no utilizarlas para declarar variables y clases.
- Los operadores tienen una prelación que hay que tener en cuenta a la hora de realizar operaciones.
- Recuerda que todas las instrucciones deben terminar en punto y coma ";".
- No podemos modificar variables constantes.

- Java es OpenSource.
- Puedes programar hasta desde el block de notas.
- Hay que saber diferenciar entre errores y excepciones.

¡Felicidades! 👍 Has concluido con la lectura de la Guía Didáctica N°1. Así que ya puedes realizar la Evaluación 1.

REFERENCIAS BIBLIOGRÁFICAS

Aboutespanol. (27 de 09 de 2018). *About Español*. Obtenido de About Español: <https://www.aboutespanol.com/que-es-java-157854>

Alegsa. (01 de 10 de 2009). *Alegsa*. Recuperado el 01 de 10 de 2018, de Alegsa: <http://www.alegsa.com.ar/Dic/dato.php>

Arkaitzgarro. (12 de 10 de 2018). *Arkaitzgarro*. Recuperado el 12 de 10 de 2018, de Arkaitzgarro: <https://www.arkaitzgarro.com/java/capitulo-4.html>

Campusmvp. (17 de 07 de 2018). *Campusmvp*. Recuperado el 06 de 10 de 2018, de Campusmvp: <https://www.campusmvp.es/recursos/post/variables-y-tipos-de-datos-en-java-tipos-simples-clases-y-tipos-envoltorio-o-wrapper.aspx>

Campusmvp. (17 de 07 de 2018). *Campusmvp*. Recuperado el 08 de 10 de 2018, de Campusmvp: <https://www.campusmvp.es/recursos/post/variables-y-tipos-de-datos-en-java-tipos-simples-clases-y-tipos-envoltorio-o-wrapper.aspx>

Oracle. (24 de 10 de 2018). *Oracle Docs*. Recuperado el 24 de 10 de 2018, de Oracle Docs:

<https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html>

Significados. (16 de 02 de 2017). *Significados*. Recuperado el 01 de 10 de 2018, de Significados:

<https://www.significados.com/informacion/>

Significados. (16 de 02 de 2017). *Significados*. Recuperado el 01 de 10 de 2018, de Significados:

<https://www.significados.com/informacion/>

Sistemas. (01 de 10 de 2018). *Sistemas*. Recuperado el 01 de 10 de 2018, de Sistemas: <https://sistemas.com/variable.php>

Wikibooks. (16 de 09 de 2018). *Wikibooks Wikipedia*. Recuperado el 01 de 10 de 2018, de Wikibooks:

https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_Java/Variables

Wikipedia. (27 de 09 de 2018). *Wikipedia*. Obtenido de Wikipedia:

[https://es.wikipedia.org/wiki/Java_\(lenguaje_de_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n))

Esta guía fue elaborada para ser utilizada con fines didácticos como material de consulta de los participantes en el Diplomado Virtual en Programación en Java del Politécnico de Colombia, especialmente, a los técnicos, tecnólogos y profesionales de carreras afines, estudiantes de todas las carreras, empíricos, y público en general con conocimientos básicos en informática que intentan entrar en el mundo de la programación, que se desempeñen o no en las áreas de TIC de cualquier tipo de organización y que deseen obtener las competencias y

habilidades necesarias para conocer los fundamentos prácticos del lenguaje de programación Java para la aplicación y desarrollo de algoritmos y aplicaciones, y solo podrá ser reproducida con esos fines. Por lo tanto, se agradece a los usuarios referirla en los escritos donde se utilice la información que aquí se presenta.

Derechos reservados - POLITÉCNICO DE COLOMBIA, 2018
Medellín, Colombia