

1. Algoritmos y su Importancia

Un algoritmo es un conjunto ordenado y finito de pasos o instrucciones que debemos seguir para realizar una tarea o resolver un problema.

El algoritmo tiene datos de entrada (input), los cuáles son necesarios para que el algoritmo se ejecute, así como también produce datos de salida (output), los cuáles son el resultado final del algoritmo.

En nuestra vida diaria están presentes muchos algoritmos, como cuando queremos sumar dos números, sabemos que primero sumamos las unidades, luego las decenas y así sucesivamente, aunque no lo crean eso es un algoritmo, donde el input son los números que queremos sumar y el output es el resultado de la suma.

Otro ejemplo clásico de un algoritmo es una receta de cocina, donde el input son los ingredientes, el output es el plato de comida terminado y el algoritmo nos indica como pasar de un estado al otro. O cuando queremos matricularnos en la universidad, sabemos que cada universidad tiene definido sus propios pasos a seguir, es decir su algoritmo.

Hay muchos algoritmos que para nosotros funcionan como una “caja negra”, es decir sabemos cuál es su funcionalidad o propósito, pero no sabemos al detalle cómo trabaja. Por ejemplo, cuando entramos a Google, ingresamos una palabra y le damos buscar; nosotros sabemos que obtendremos todo los artículos o páginas relacionadas a esa palabra o texto, pero no sabemos internamente cómo se obtiene esa información, cómo Google la almacena, etc.

Aunque ahora sabemos que todo en nuestra vida diaria tiene su algoritmo y si lo planteáramos así todo sería mucho más fácil; la mayor importancia de los algoritmos se encuentra en la informática, ciencias de la computación, robótica, inteligencia artificial y disciplinas relacionadas a la tecnología ya que actualmente con el desarrollo tecnológico vienen siendo muy utilizados. Podemos resaltar algunos algoritmos utilizados en la inteligencia artificial como los algoritmos genéticos, algoritmos de búsqueda, las redes neuronales; en la seguridad informática podemos ver muchos algoritmos de encriptación como el RSA; asimismo en las ciencias de la computación podemos encontrar muchos algoritmos de grafos como Dijkstra, Floyd, entre otros.

2. Pseudocódigo, lenguaje de programación y java

2.1. Pseudocódigo

Es una manera de representar un algoritmo de tal forma que sea fácil de entender para las personas, es por ello que no existe una sintaxis estándar.

El pseudocódigo no puede ser ejecutado por una computadora, pero tiene como ventaja que nos permite planificar de una mejor manera un programa y sobre todo centrarnos en la lógica más que en los problemas propios del lenguaje de programación que se vaya a utilizar.

En general un pseudocódigo consta de las siguientes partes:

- Nombre del algoritmo
- Palabra clave **Inicio**, la cual no indicará que ahí empieza el algoritmo.
- Ahora siguen los pasos o instrucciones.
- Palabra clave **Fin**, la cual nos indicará el fin de nuestro algoritmo.

Por ejemplo, podemos hacer un algoritmo para tomar un vuelo internacional:

Algoritmo Vuelo Internacional

Inicio

*Ir al aeropuerto
Hacer check-in y dejar el equipaje
Pasar por control de seguridad
Pasar por migraciones
Ir a la puerta de embarque correspondiente
Subir al avión*

Fin

2.2. Lenguaje de Programación

Es un lenguaje formado por un conjunto de símbolos y reglas, diseñado para realizar procesos que puedan ser llevados a cabo por las computadoras. Así como existe el lenguaje humano, éste es un lenguaje que puede ser entendido por las computadoras.

Algunos de los lenguajes de programación más conocidos son Java, Python, C++, C#, PHP, Ruby, entre otros.

2.3. Programa

Es una secuencia de instrucciones realizadas en algún lenguaje de programación que serán ejecutadas por una computadora.

2.4. Java

Es un lenguaje de programación orientado a objetos (esta teoría no lo trataremos mucho en este curso, sólo lo necesario), comercializado por primera vez en 1995 por Sun Microsystems, que surgió a partir de la evolución de los lenguajes C y C++.

Actualmente es uno de los lenguajes más usados, permitiéndonos desarrollar aplicaciones de escritorio, web y móviles. Además, Java resalta mucho por ser un lenguaje multiplataforma, es decir el mismo código funciona en cualquier sistema operativo.

2.4.1. Java Runtime Enviroment (JRE)

Es el entorno de ejecución Java, el cual nos brinda lo necesario para que un programa escrito en este lenguaje pueda ser ejecutado.

2.4.2. Java Development Kit (JDK)

Es la plataforma de desarrollo Java, el cual incluye el JRE más otras herramientas de desarrollo como un compilador, debugger y otras necesarias para desarrollar una aplicación en java.

2.5. Eclipse

Es un entorno de desarrollo integrado (IDE), principalmente utilizado para Java, el cual nos permite escribir y ejecutar nuestro código de una manera muy sencilla.

Además de Eclipse, para Java existen otros IDE's como NetBeans, JDeveloper e IntelliJ.

En este curso utilizaremos Eclipse, ya que es el IDE más robusto y usado de manera profesional, incluso soportando otros lenguajes como C++, PHP y Python.

2.6. Preparando nuestro ambiente de trabajo

- Instalamos el JDK

Como vamos a crear programas necesitamos el JDK, para ello vamos a la página principal de Oracle (www.oracle.com) y luego a la sección de Downloads, donde seleccionamos *Java for Developers*, o simplemente ir al siguiente enlace:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Seleccionamos *Java Platform (JDK) 8u91* (una de las versiones más recientes del JDK) y nos aparecerá la siguiente pantalla.

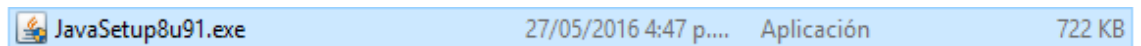
Java SE Development Kit 8u91

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 | 77.72 MB | jdk-8u91-linux-arm32-vfp-hflt.tar.gz |
| Linux ARM 64 Hard Float ABI | 74.69 MB | jdk-8u91-linux-arm64-vfp-hflt.tar.gz |
| Linux x86 | 154.74 MB | jdk-8u91-linux-i586.rpm |
| Linux x86 | 174.92 MB | jdk-8u91-linux-i586.tar.gz |
| Linux x64 | 152.74 MB | jdk-8u91-linux-x64.rpm |
| Linux x64 | 172.97 MB | jdk-8u91-linux-x64.tar.gz |
| Mac OS X | 227.29 MB | jdk-8u91-macosx-x64.dmg |
| Solaris SPARC 64-bit (SVR4 package) | 139.59 MB | jdk-8u91-solaris-sparcv9.tar.Z |
| Solaris SPARC 64-bit | 98.95 MB | jdk-8u91-solaris-sparcv9.tar.gz |
| Solaris x64 (SVR4 package) | 140.29 MB | jdk-8u91-solaris-x64.tar.Z |
| Solaris x64 | 96.78 MB | jdk-8u91-solaris-x64.tar.gz |
| Windows x86 | 182.29 MB | jdk-8u91-windows-i586.exe |
| Windows x64 | 187.4 MB | jdk-8u91-windows-x64.exe |

Damos click en aceptar licencia y luego podemos descargar el JDK. Es un ejecutable (.exe) y lo veremos algo así.



Luego lo instalamos como cualquier programa en Windows, con toda la configuración por defecto (simplemente dando next a todo).

- Descargamos Eclipse

Como ya lo habíamos mencionado, el IDE que usaremos será Eclipse. Este IDE no necesita instalación, sólo lo tenemos que descargar de la página principal de Eclipse en su sección Downloads: <https://eclipse.org/downloads/>

Seleccionamos la opción indicada en la imagen siguiente, que es la opción más ligera de Eclipse.

Eclipse IDE for Java EE Developers
 276 MB | 1,937,402 DOWNLOADS
 Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn...

Windows
 32 bit | 64 bit

Eclipse IDE for Java Developers
 167 MB | 1,000,484 DOWNLOADS
 The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven integration and WindowBuilder...

Windows
 32 bit | 64 bit

Eclipse IDE for C/C++ Developers
 177 MB | 279,823 DOWNLOADS
 An IDE for C/C++ developers with Make integration...

Windows
 32 bit | 64 bit

Luego seleccionamos la versión de nuestro IDE, para este curso usaremos Mars, una de la versiones recientes y estables.

RELEASES

Neon Packages

Mars Packages

Luna Packages

Kepler Packages


Juno Packages

Indigo Packages

Helios Packages

Galileo Packages

Ganymede Packages



Eclipse IDE for Java Developers

Package Description

The essential tools for any Java developer, including a Java IDE, a Git client, Mylyn, Maven integration and WindowBuilder

This package includes:

- Eclipse Git Team Provider
- Eclipse Java Development Tools

Luego seleccionamos la opción ligera para Java antes mencionada y escogemos que sea de 64-bit para Windows (si su sistema operativo es de 32-bit escoger la otra opción).

Eclipse Mars 2 Packages

**Eclipse IDE for Java EE Developers**
275 MB - Downloaded 1,938,121 Times

**Eclipse IDE for Java Developers**
167 MB - Downloaded 1,000,749 Times

Windows **32-bit 64-bit**
Mac Cocoa **64-bit**
Linux **32-bit 64-bit**

Windows **32-bit 64-bit**
Mac Cocoa **64-bit**
Linux **32-bit 64-bit**

Finalmente, sólo le damos click en Download y comenzará la descarga de nuestro Eclipse Mars

Download from: **Brazil - C3SL - Federal University of Parana (http)**

OR >


File: eclipse-java-mars-2-win32-x86_64.zip


Checksums: MD5 SHA1

SHA-512

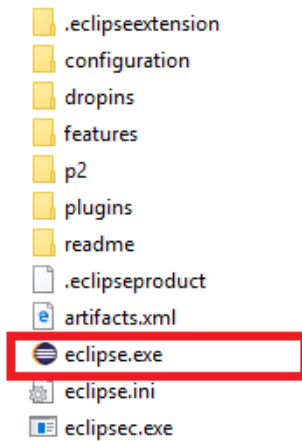
DOWNLOAD

Get It Faster from our Members

**IBM**
Blazingly fast downloads hosted by IBM Bluemix.
DOWNLOAD

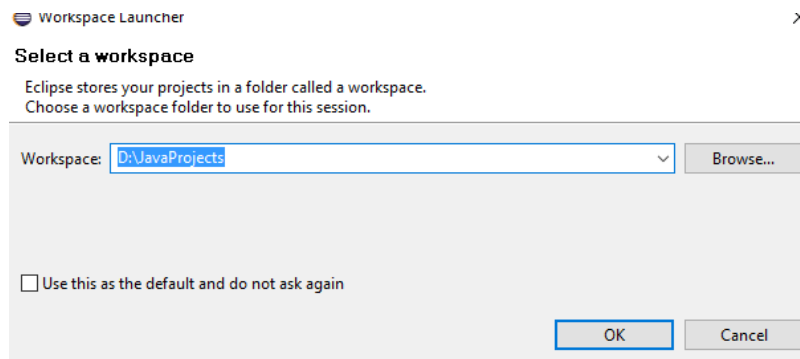
**Spring by Pivotal**
Rapid downloads of Eclipse packages. Free downloads of Spring Tool Suite for Spring, Spring Boot, Cloud Foundry, and AspectJ.
DOWNLOAD

Descomprimos la carpeta que descargamos y la guardamos en la ruta que creamos conveniente, ya que en esa carpeta se encuentra el ejecutable de Eclipse.



- Creamos un proyecto

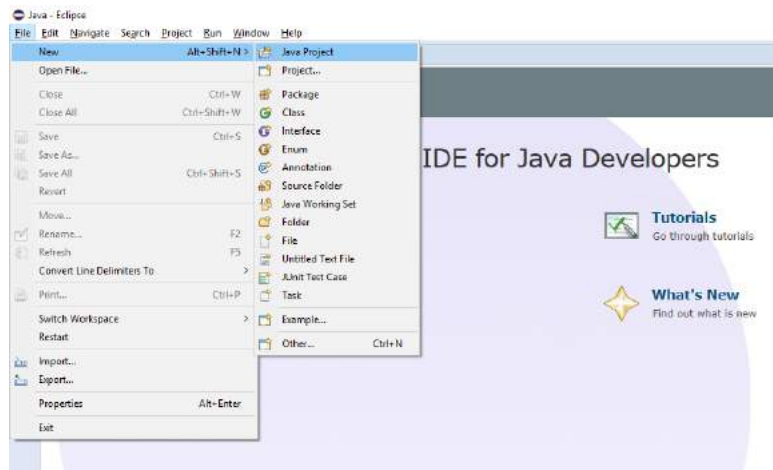
Damos doble click en el ícono de Eclipse de la imagen anterior, para iniciar el programa. Luego Eclipse nos preguntará la ruta de nuestro Workspace, que es donde guardará todos nuestros proyectos que creemos.



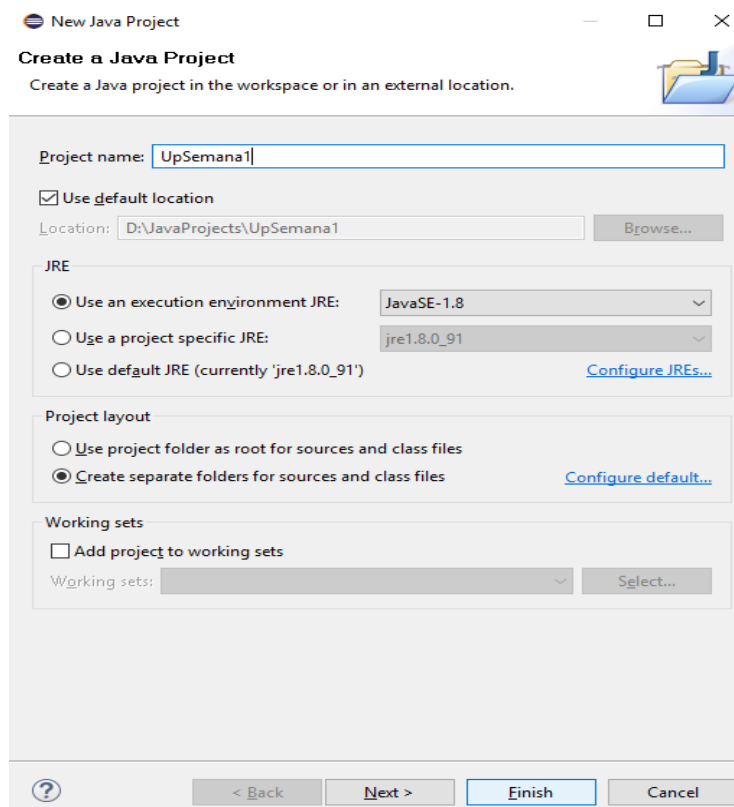
Escogemos la ruta que deseamos y le damos click en OK. Después nos aparecerá la imagen de bienvenida de Eclipse.



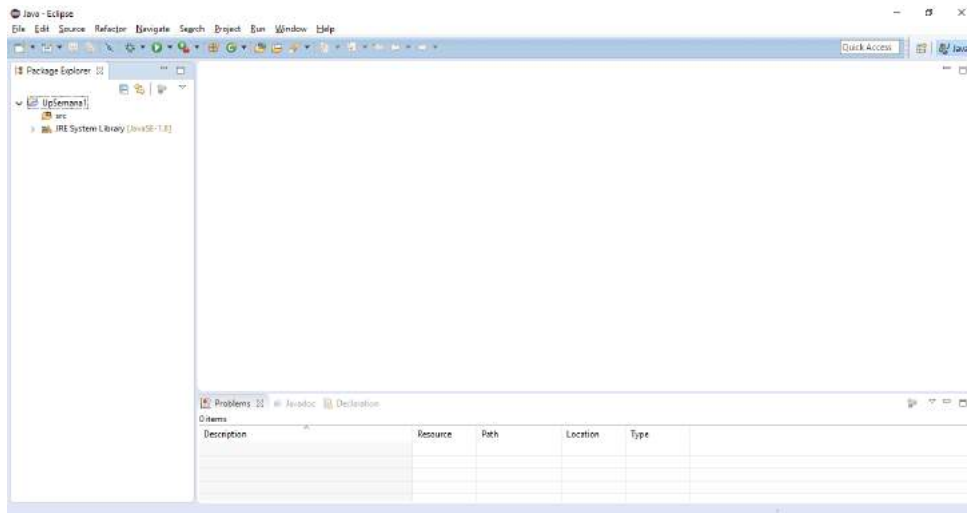
Ahora que ya estamos en Eclipse, crearemos un nuevo proyecto con nombre *UpSemana1*. Para esto vamos a File > New > Java Project



Colocamos el nombre del proyecto y le damos finish.



Si nos sigue apareciendo la pestaña de bienvenida, la cerramos y tendremos la siguiente pantalla, si sale algunas pestañas extras a la imagen, ciérruelas porque no las vamos a utilizar. Finalmente tenemos configurado nuestro ambiente de trabajo.



3. Variables, operadores, lectura y escritura

3.1. Variable

Es un espacio en la memoria de la computadora donde se va a almacenar información y posee un nombre identificador asociado. Su valor puede cambiar durante la ejecución del algoritmo, es por ello que recibe el nombre de variable.

- **Tipo de Dato**

Las variables poseen un tipo de dato, el cual nos define qué tipo valores podrá tomar. Existen 3 tipos de datos considerados como primitivos:

- Numérico: números, tanto enteros (int) como reales (double).
- Lógico o Booleano: solo puede tomar los valores de verdadero o falso.
- Carácter: puede ser una letra, un número, un signo de puntuación u otros especiales que tienen que ver más con el procesamiento de textos. Para ver la lista completa de caracteres se puede ver la codificación ASCII, la cual asigna un valor entero a cada carácter.

- **Asignación**

Permite guardar un valor en una variable. Se utiliza el signo igual (=) para esta acción.

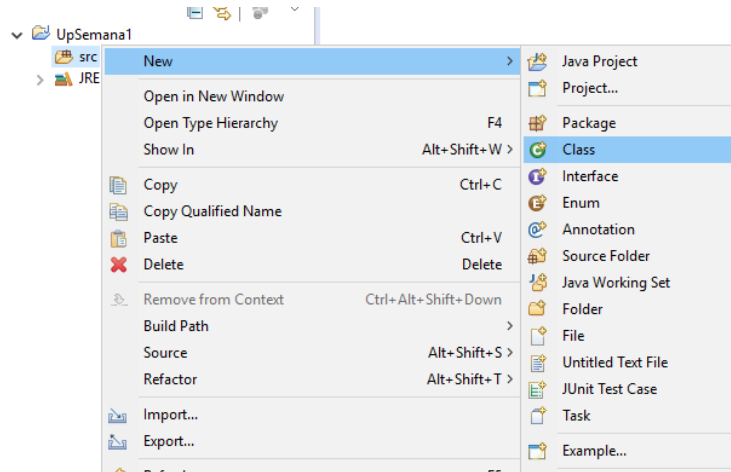
- **Lectura**

Permite recibir valores ingresados por teclado y guardarlos en variables.

- **Escritura**

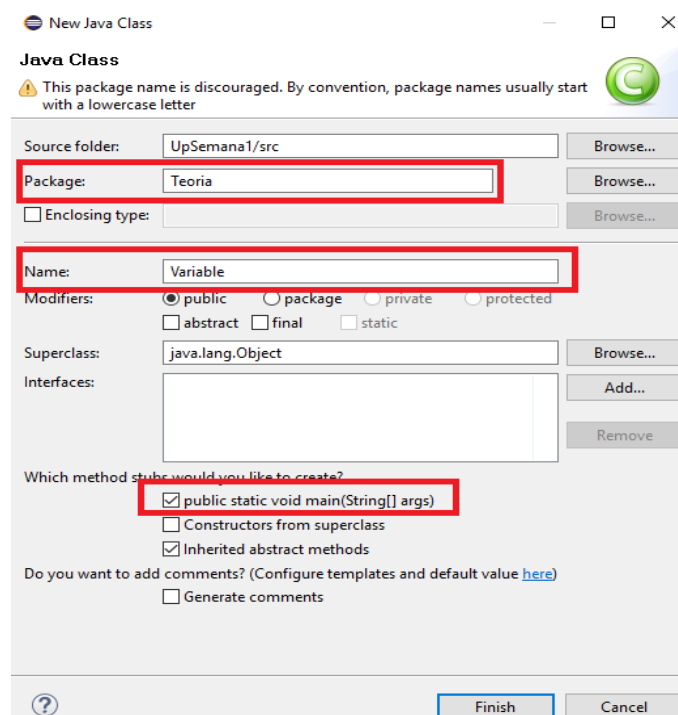
Permite mostrar en pantalla (consola) algún dato o valor de una variable.

A continuación, veremos un ejemplo para aprender a trabajar con las variables. Para eso creamos una nueva clase (archivo que nos permitirá escribir nuestro código) dando click derecho en el paquete `src` y seleccionamos `new > Class`.

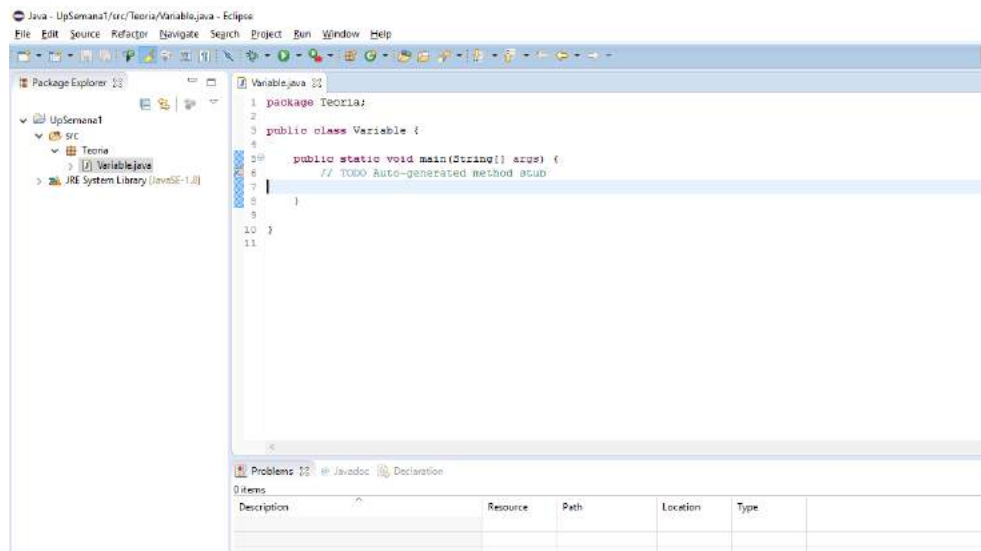


Ahora nos aparece la siguiente ventana donde debemos completar todo lo resaltado, antes de eso haremos una breve explicación: en primer lugar, debemos saber que el package o paquete, simplemente es un agrupador de clases (por ahora veámoslo como archivos), el cual nos permite ser más ordenados.

Luego nos pide el name o nombre que tendrá nuestra clase y finalmente debemos dar check para que incluya el método `main`, necesario para que se ejecute un programa Java y es lo primero que se ejecuta.

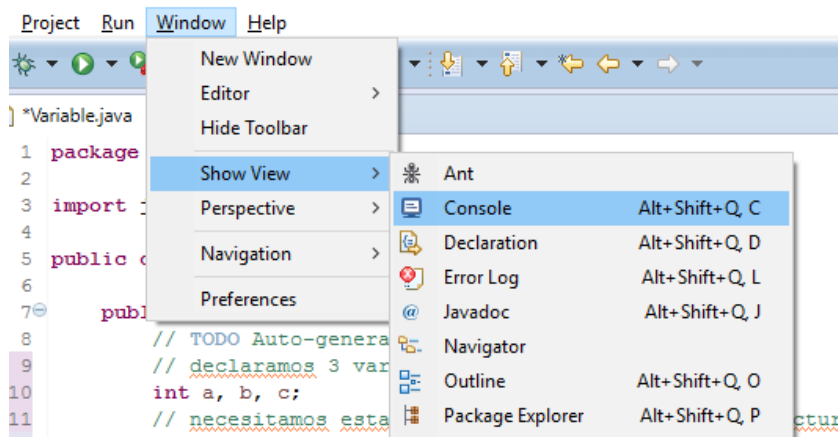


Finalmente tendremos la siguiente pantalla.



Bueno ahora que ya lo explicamos, siempre que queramos crear una nueva clase, haremos lo mismo, aunque ahora todo lo que creemos lo podemos poner en el paquete Teoría.

El siguiente ejemplo sólo realiza la lectura de 3 números enteros y los muestra en consola. Para poder ver la consola, guiarse de la imagen:



Ahora que la consola la veremos en la parte inferior de la pantalla. Ahora si podemos analizar el código, para esto se ha comentado cada línea de código mediante la sentencia `//`, todo lo que esta comentado (de verde) no será ejecutado por el programa.

```

package Teoria;

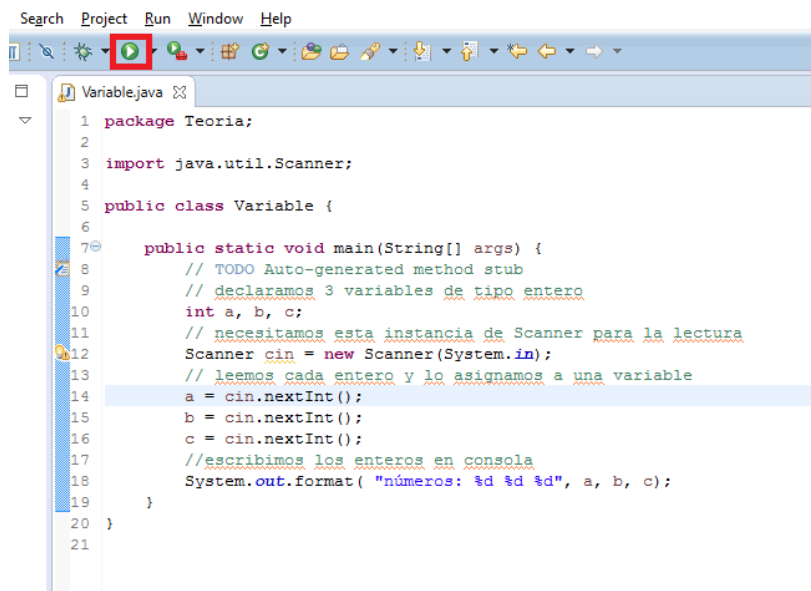
import java.util.Scanner;

public class Variable {

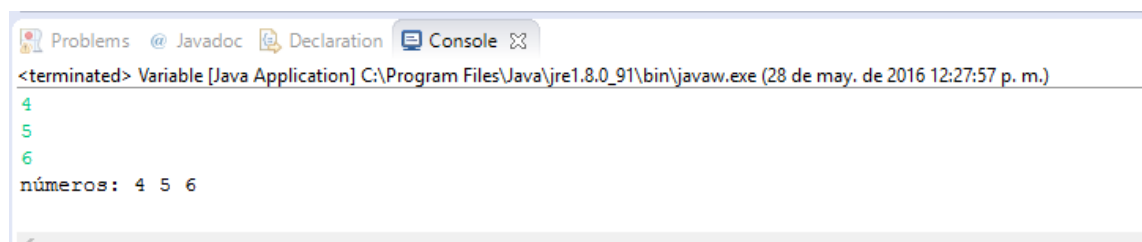
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        // declaramos 3 variables de tipo entero
        int a, b, c;
        // necesitamos esta instancia de Scanner para la lectura
        Scanner cin = new Scanner(System.in);
        // leemos cada entero y lo asignamos a una variable
        a = cin.nextInt();
        b = cin.nextInt();
        c = cin.nextInt();
        //escribimos los enteros en consola
        System.out.format( "números: %d %d %d", a, b, c);
    }
}

```

Para ejecutar el programa sólo debemos dar click en el ícono play.



Ahora todo pasa a la consola, donde podemos ingresar los 3 números separados por *enter* o por *espacio*. Cuando terminamos de ingresar los números presionamos *enter* y veremos el siguiente resultado:



Si queda alguna duda de la forma de escrita usada, es una escritura con formato, donde cada %d será reemplazado por la variable indicada. El primer %d será reemplazado por la variable a , el segundo por la variable b y el tercer por c.

Posteriormente veremos que para número reales tendremos que usar %c.

3.2. Operadores

3.2.1. Operadores Matemáticos

Son símbolos que nos permitirán realizar cálculos matemáticos, además de ser casi estándar en todos los lenguajes de programación.

| Operador | Significado | Ejemplo |
|----------|--------------------------------------|-------------------------------|
| + | Suma | $a = b + c$ |
| - | Resta | $a = b - c$ |
| * | Multiplicación | $a = b * c$ |
| / | División | $a = b / c$ |
| % | Módulo (resto de la división entera) | resto = $n \% \text{divisor}$ |

3.2.2. Operadores de Comparación

Son símbolos que nos permitirán comparar dos valores, si el resultado de la comparación es correcto entonces la expresión es considerada verdadera, caso contrario será falsa.

| Operador | Significado | Ejemplo |
|----------|-------------------|-----------|
| > | Mayor que | $10 > 8$ |
| < | Menor que | $4 < 7$ |
| >= | Mayor o igual que | $3 >= 3$ |
| <= | Menor o igual que | $2 <= 6$ |
| == | Igual a | $5 == 5$ |
| != | Distinto a | $3 != 10$ |

3.2.3. Operadores Lógicos

Estos operadores nos permitirán obtener un resultado verdadero o falso a partir de un conjunto de expresiones o condiciones.

| Operador | Significado | Ejemplo |
|----------|---|---------------------------------------|
| && | Conjunción: si ambas expresiones son verdaderas, el resultado es verdadero, caso contrario el resultado es falso. | (10>8) && (5=6) devuelve falso |
| | Disyunción: si ambas expresiones son falsas, el resultado es falso, caso contrario el resultado es verdadero. | (10>8) (5=6) devuelve verdadero |
| ! | Negación: si la expresión es verdadera, el resultado será falso, caso contrario el resultado será verdadero. | ! (3>=3) devuelve falso |

Ahora veremos un ejemplo usando operadores matemáticos, hallaremos la suma y multiplicación de dos números. El resto de operadores los usaremos cuando veamos la estructura de control condicional.

```
package Teoria;

import java.util.Scanner;

public class Operadores {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a, b;
        Scanner cin = new Scanner(System.in);
        // escritura sin formato y agrega salto de línea
        System.out.print("Ingrese a: ");
        a = cin.nextInt();
        System.out.print("Ingrese b: ");
        b = cin.nextInt();
        // \n genera un salto de línea
        System.out.format("La suma es %d.\n", a+b );
        System.out.format("El producto es %d.", a*b);
    }
}
```

4. Estructuras de control condicional: simple, doble y anidada

Nos permiten gobernar el flujo de ejecución de las instrucciones según se cumplan ciertas condiciones.

4.1. Condicional Simple

Se encarga de evaluar una condición, en caso sea verdadera realiza un conjunto de acciones, en caso contrario no hace nada. Es a partir de ahora en que nos ayudará mucho el pseudocódigo, la estructura es la siguiente:

```
Si <condicion> Entonces
    <instrucciones>
FinSi
```

Ahora veremos un ejemplo, para ver si una edad es mayor que otra:

Algoritmo CondicionalSimple

```
Inicio
    Definir miEdad, tuEdad Como Entero;
    Leer miEdad, tuEdad;
    Si ( miEdad > tuEdad ) Entonces
        Escribir "Soy mayor que tú";
    FinSi
Fin
```

Ahora lo pasaremos a Java, seguimos trabajando en el paquete Teoria, donde creamos una nueva clase con nombre CondicionalSimple:

```
package Teoria;

import java.util.Scanner;

public class CondicionalSimple {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int miEdad, tuEdad;
        Scanner cin = new Scanner(System.in);
        miEdad = cin.nextInt();
        tuEdad = cin.nextInt();
        if( miEdad > tuEdad){
            System.out.println("Soy mayor que tú");
        }
    }
}
```

4.2. Condicional Doble

Se encarga de evaluar una condición en caso sea verdadera realiza un conjunto de acciones, en caso contrario realiza otro grupo de acciones. La estructura es la siguiente:

Si <condicion> Entonces
 <instrucciones1>
Sino
 <instrucciones2>
FinSi

Ahora complementaremos el ejemplo anterior:

Algoritmo CondicionalDoble

Inicio
 Definir miEdad, tuEdad Como Entero;
 Leer miEdad, tuEdad;
 Si (miEdad > tuEdad) Entonces
 Escribir "Soy mayor que tú.";
 Sino
 Escribir "Soy menor que tú o tenemos la misma edad."
 FinSi
Fin

Ahora en Java:

```
package Teoria;

import java.util.Scanner;

public class CondicionalDoble {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int miEdad, tuEdad;
        Scanner cin = new Scanner(System.in);
        miEdad = cin.nextInt();
        tuEdad = cin.nextInt();
        if( miEdad > tuEdad){
            System.out.println("Soy mayor que tú");
        }
        else{
            System.out.println("Soy menor que tú o tenemos
la misma edad");
        }
    }
}
```

4.3. Condicional Anidada

Este tipo de condicional presenta condiciones a evaluar dentro de otras condiciones (combina la condicional simple y doble), éstas estructuras pueden seguir creciendo según como uno se las plantee. La estructura más básica es la siguiente:

```

Si <condicion> Entonces
    <instrucciones1>

    Si <condicion> Entonces
        <instrucciones2>
    FinSi
Sino
    <instrucciones3>
    Si <condicion> Entonces
        <instrucciones4>
    FinSi
FinSi

```

Ahora seguiremos con un ejemplo complementario a los anteriores:

Algoritmo CondicionalAnidada

```

Inicio
    Definir miEdad, tuEdad Como Entero;
    Leer miEdad, tuEdad;
    Si ( miEdad > tuEdad ) Entonces
        Escribir "Soy mayor que tú.";
    Sino
        Si ( miEdad == tuEdad )Entonces
            Escribir "Tenemos la misma edad.";
        Sino
            Escribir "Soy menor que tú";
        FinSi
    FinSi
Fin

```

Para terminar esta sesión veremos este ejemplo en Java:

```

package Teoria;
import java.util.Scanner;

public class CondicionAnidada {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int miEdad, tuEdad;
        Scanner cin = new Scanner(System.in);
        miEdad = cin.nextInt();
        tuEdad = cin.nextInt();
        if( miEdad > tuEdad ){
            System.out.println("Soy mayor que tú");
        }
        else{
            if( miEdad == tuEdad){
                System.out.println("Tenemos la misma edad");
            }
            else{
                System.out.println("Soy menor que tú");
            }
        }
    }
}

```