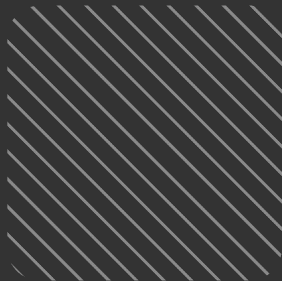


Introducción a Java

//Parte 1

IT BOARDING

BOOTCAMP





Pablo Sznajdleder

DOCENTE

Ingeniero en Sistemas de Información.

Profesor Universitario.

Autor de Java a Fondo y Algoritmos a Fondo, entre otras obras.

Índice



01

El lenguaje de programación
Java, algo de historia y
versionado

03

Sintaxis, variables y
tipos de dato

02

IntelliJ IDEA (IDE) y
HolaMundo.java

04

Estructuras de
control

IT BOARDING

BOOTCAMP

// El lenguaje de programación Java

IT BOARDING

BOOTCAMP

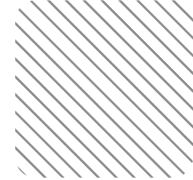
*¿Por qué **Java**? ¿Cómo surge,
evoluciona, y qué es **Java hoy en día**?*





Versiones de lenguaje de programación Java

- Desde el inicio y hasta 1998, se llamaron **JDK**. La primera versión fue **JDK1.0.2**
- En 1998 surge Java2, y se separó en dos partes:
 - El lenguaje de programación propiamente dicho: **J2SE**
 - Una biblioteca o plataforma de desarrollo del backend: **J2EE**
- En 2004, Java5 trae dos aportes: Generics y Annotations
- En 2014, Java8 agrega: Expresiones lambda y Streams



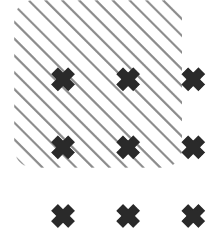
Java como lenguaje y plataforma del backend

- Java es un **lenguaje** de programación de **propósitos generales**
- **Spring Boot**, programado en Java, es una plataforma de ejecución del *backend* de las **aplicaciones empresariales**

¿Aplicaciones Empresariales?

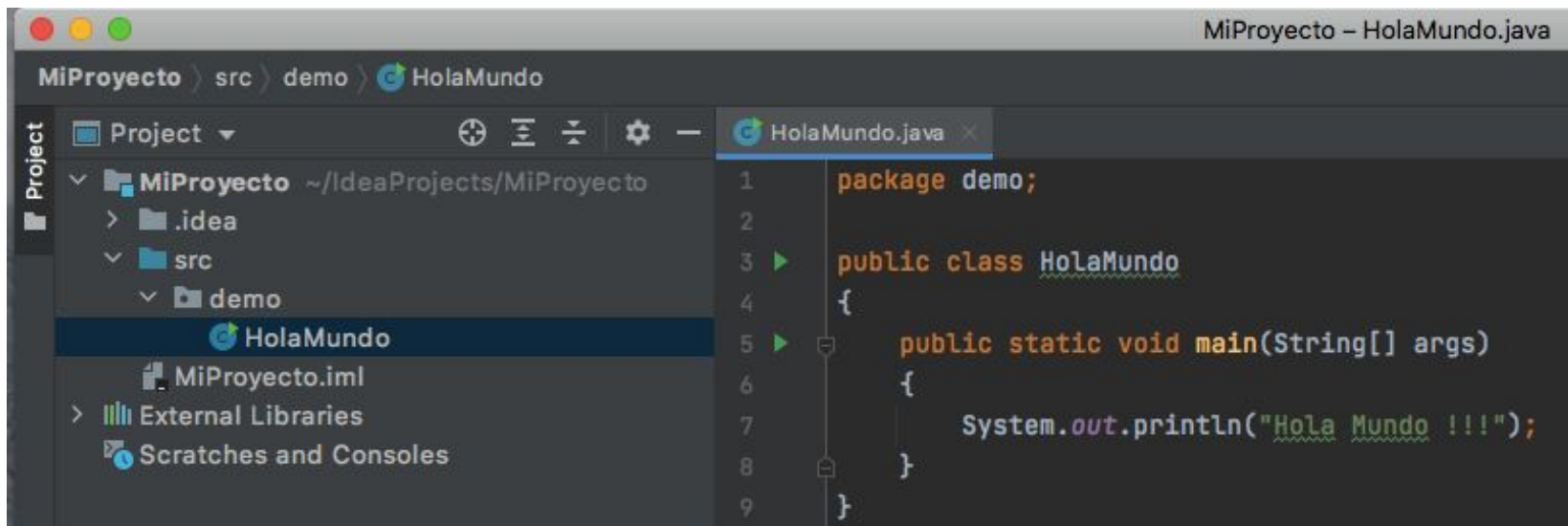


Comenzando a programar: Hello World



IntelliJ, la herramienta de desarrollo

Para programar con Java necesitamos disponer de una herramienta que nos asista durante todo el proceso de desarrollo. Estas herramientas se llaman IDE



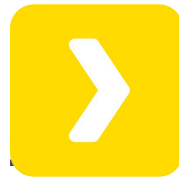
Sintaxis del lenguaje

Java tiene una sintaxis similar a la de C y C++:

- Es **Case sensitive**
- Las líneas finalizan con ; (**punto y coma**)
- Los bloques de código se delimitan con { } (**llaves**)

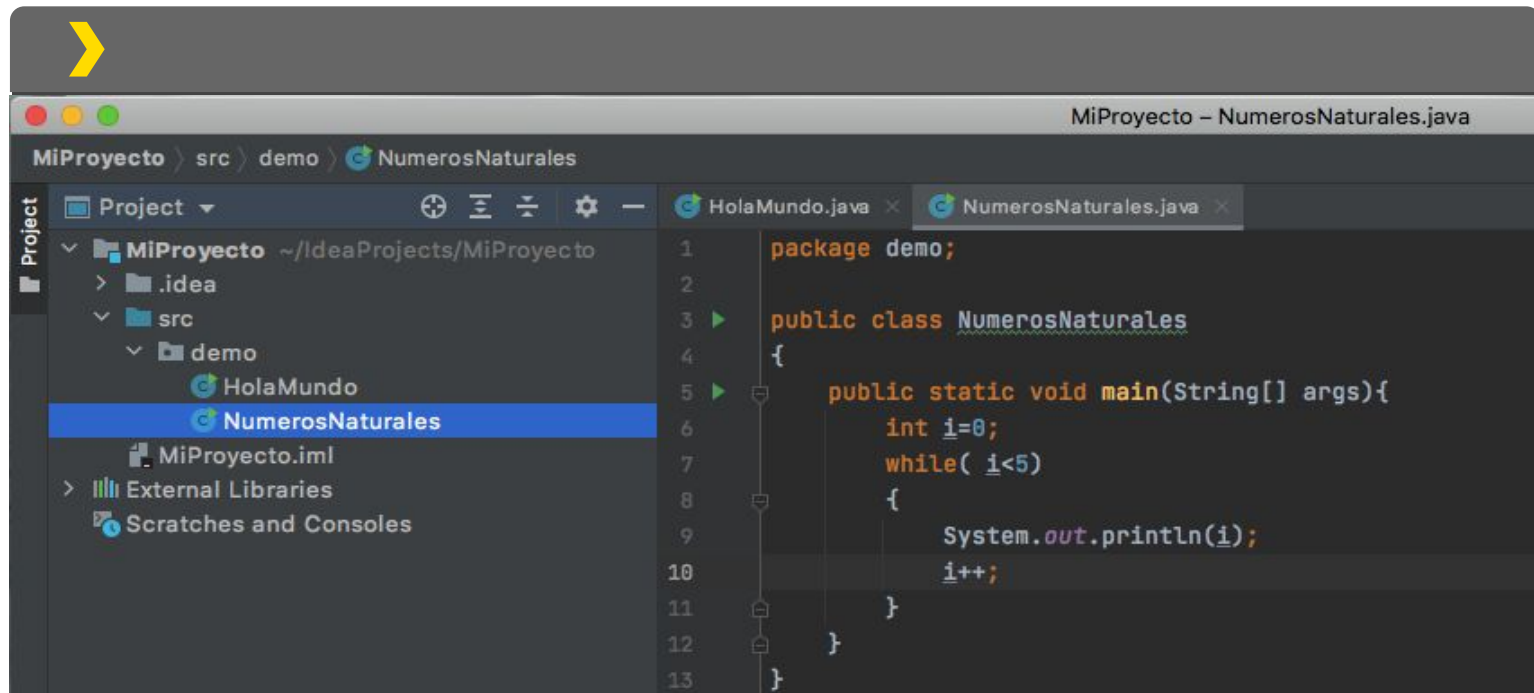
Además, por ser un lenguaje Orientado a Objetos:

- Cualquier cosa que hagamos estará dentro de una **clase**.
- Los programas de entrada son clases que tienen el método especial “**main**”.
- Las clases quedan en archivos con el mismo <nombre>.**java**



Otro ejemplo

El siguiente programa muestra los primeros 5 números naturales en la consola.



The screenshot shows an IDE window titled "MiProyecto - NumerosNaturales.java". The left sidebar displays the project structure: "MiProyecto" (root) contains ".idea", "src", and "MiProyecto.iml". The "src" directory contains a "demo" package with two classes: "HolaMundo" and "NumerosNaturales" (selected). The main editor shows the code for "NumerosNaturales.java":

```
1 package demo;
2
3 public class NumerosNaturales
4 {
5     public static void main(String[] args){
6         int i=0;
7         while( i<5)
8         {
9             System.out.println(i);
10            i++;
11        }
12    }
13 }
```

Tipos de dato

Java tiene los mismos tipos de dato que C y C++:

- **char** (entero, 2 bytes, sin bit de signo)
- **short** (entero, 2 bytes)
- **int** (entero, 4 bytes)
- **long** (entero, 8 bytes)

- **float** (flotante, 4 bytes)
- **double** (flotante, 8 bytes)

- **boolean** (lógico)
- **String** (no es un tipo primitivo, **es una clase**)

- **void** (tipo de dato nulo, aplicable a funciones que no retornan nada)

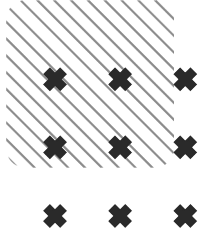




Virtual Machine (JVM) y Runtime Enviroment (JRE)

- Los programas Java corren dentro de una Máquina Virtual
- Linux, Windows, Mac, AS-400, existe una JRE para cada plataforma
- Sea cual fuere la máquina real, los programas corren en la JVM
- En la JVM, los tipos de dato tienen las longitudes mencionadas

Multiplataforma, write once, run anywhere



Variables

Así declaramos variables y les asignamos valores literales.

```
int a;  
double d;  
char c = 'A';  
boolean b = false;
```

```
String s = "Hola";  
s = s+", todo bien?";  
  
System.out.println(s);
```



Convención de nombres

Las variables deben comenzar con minúscula. Si se trata de un nombre compuesto por dos o más palabras, cada inicial (excepto la primera) debe colocarse en mayúscula.

```
int edad;  
double alturaPromedio;  
char tipoDocumento;  
boolean fechaNacimiento;
```

```
int iEdad;  
double dAlturaPromedio;  
char cTipoDocumento;  
boolean bFechaNacimiento;
```

Estructuras de control

En Java, las estructuras de control son exactamente las mismas que C y C++.

```
if( expresión )  
{  
    // ...  
}  
else  
{  
    // ...  
}
```

```
while( expresión )  
{  
    // ...  
}
```

```
for( var;expresión;incr )  
{  
    // ...  
}
```



//Operadores aritméticos, lógicos y relacionales

Otra similitud con C/C++. Java mantiene exactamente los mismos operadores.

```
int a = 3;
int b = a+1;
int c = a*b;
int d = c/a;

double t=(double)a/5;
int r = a%5;
int s = a/5;
```

```
if( a>b && b>c )
{
    // ...
}

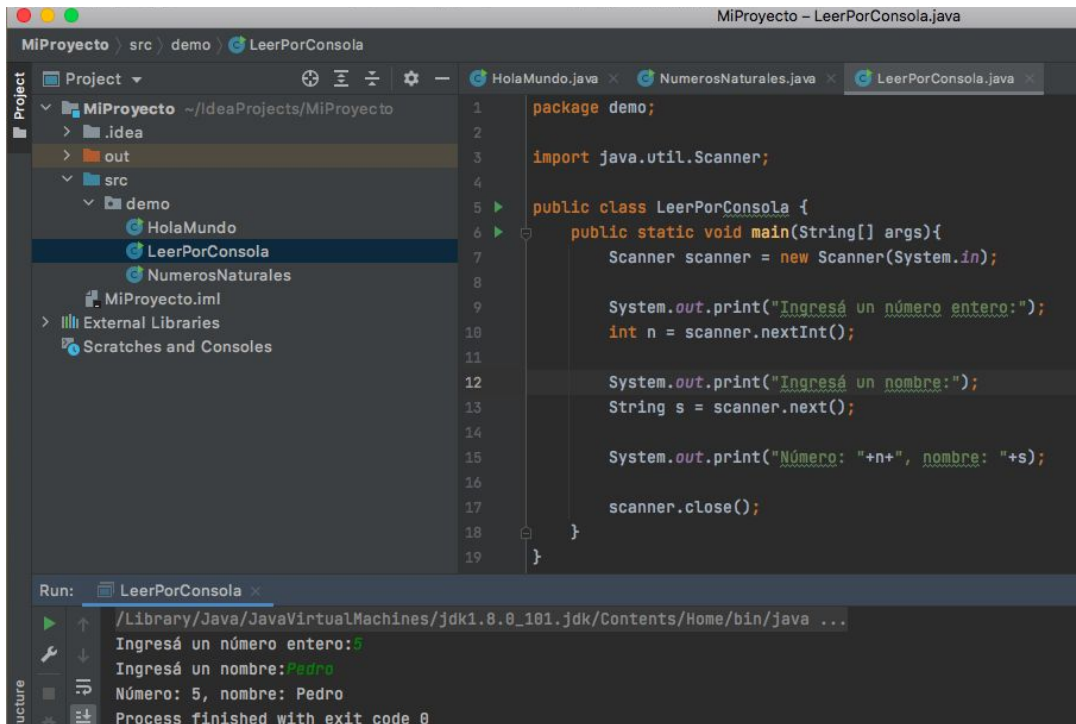
if( a>b || b>c )
{
    // ...
}
```

```
bool x = !(a>b);
bool y = a>=b;
bool z = a!=b;
```



Leer valores a través de la consola

con la clase **Scanner** podemos leer datos a través de la consola.

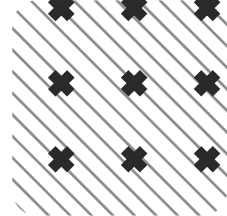


The screenshot shows an IDE window titled "MiProyecto - LeerPorConsola.java". The left sidebar displays the project structure: "MiProyecto" (containing ".idea", "out", and "src" folders) and "src" (containing "demo" folder with files "HolaMundo", "LeerPorConsola", and "NumerosNaturales", and a "MiProyecto.iml" file). The main editor shows the code for "LeerPorConsola.java":

```
1 package demo;
2
3 import java.util.Scanner;
4
5 public class LeerPorConsola {
6     public static void main(String[] args){
7         Scanner scanner = new Scanner(System.in);
8
9         System.out.print("Ingresá un número entero:");
10        int n = scanner.nextInt();
11
12        System.out.print("Ingresá un nombre:");
13        String s = scanner.next();
14
15        System.out.print("Número: "+n+", nombre: "+s);
16
17        scanner.close();
18    }
19 }
```

The bottom panel shows the "Run" output for "LeerPorConsola":

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_101.jdk/Contents/Home/bin/java ...
Ingresá un número entero:5
Ingresá un nombre:Pedro
Número: 5, nombre: Pedro
Process finished with exit code 0
```



Métodos o funciones estáticas

Aunque en POO tienen otra utilidad, las funciones estáticas (static) nos permiten crear funciones. Sí, al estilo de la programación estructurada.

```
public static boolean esPar(int n)
{
    return n%2==0;
}

public static void main(String []args)
{
    Scanner scanner = new Scanner(System.in);
    int n = scanner.nextInt();
    if( esPar(n) )
    {
        // ...
    }
}
```



¿Dudas? ¿Preguntas?



Gracias.

IT BOARDING

BOOTCAMP

