

Java SE

Básico



Presentación :)

Ventajas de aprender Java



Ventajas de aprender Java

- Empezar rápidamente

`Similar a C y C++`

- Escribir menos código

`P00 - Reutilización`

- Escribir mejor código

`Buenas prácticas de codificación`



Ventajas de aprender Java

Es uno de los lenguajes más
utilizados a nivel mundial



Ventajas de aprender Java

Es uno de los lenguajes más
utilizados a nivel mundial

Google

Amazon

Uber

Empresas financieras

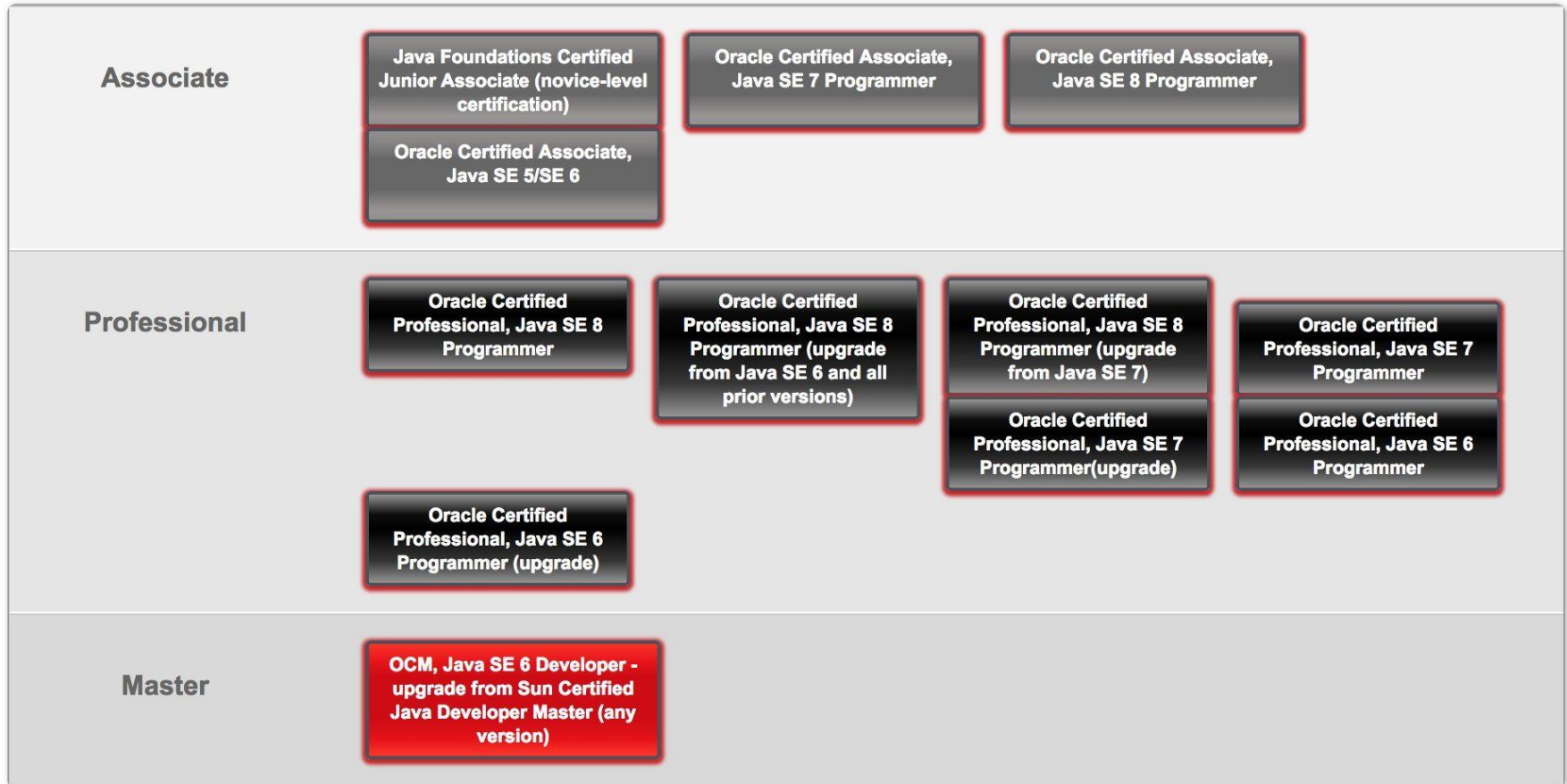


Ventajas de aprender Java

- Incrementa tus posibilidades de conseguir trabajo
- Aumentará tus aspiraciones profesionales



Ventajas de aprender Java



Ventajas de aprender Java

Especialista Certificado en Java



Ventajas de aprender Java

Especialista Certificado en Java

\$\$



Ventajas de aprender Java



¿Qué vamos a construir?



¿Qué vamos a construir?

AmazonViewer



¿Qué vamos a construir?

AmazonViewer



Introducción

Hola Mundo de Java

```
public class HolaMundo {  
    public static void main(String[] args) {  
        System.out.println("Hola mundo");  
    }  
}
```



¿Qué es **Java**?

Java es un lenguaje de
programación de
alto nivel



Java

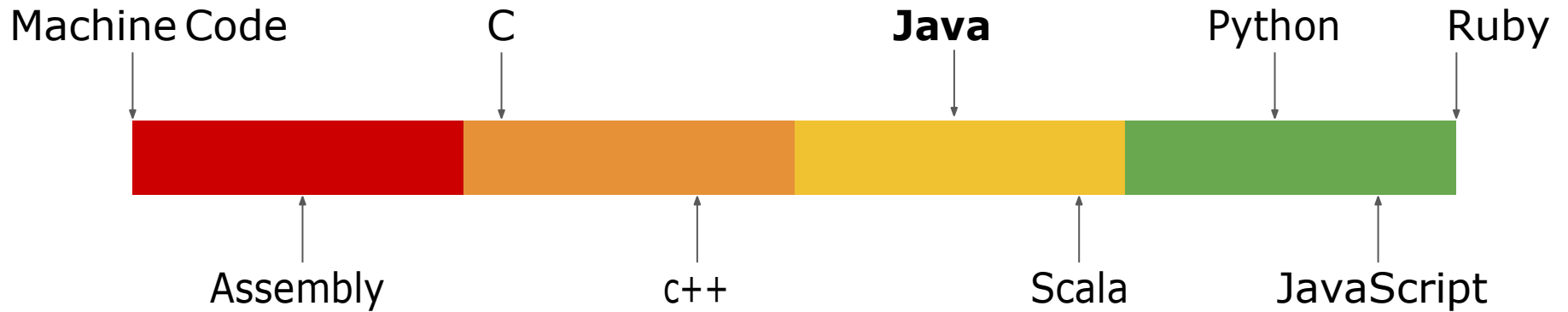
Bajo Nivel



Alto Nivel



Java



Write **once**
Run **Anywhere**

WORA

¿Qué es **Java**?

- Simple
- Orientado a Objetos
- Distribuido
- Multihilo
- Arquitectura Neutral
- Portable
- Alto desempeño
- Seguro



¿Qué es **Java**?

- **Simple**
- Orientado a Objetos
- Distribuido
- Multihilo
- Arquitectura Neutral
- Portable
- Alto desempeño
- Seguro



¿Qué es **Java**?

- Simple
- **Orientado a Objetos**
- Distribuido
- Multihilo
- Arquitectura Neutral
- Portable
- Alto desempeño
- Seguro



¿Qué es Java?

- Simple
- Orientado a Objetos
- **Distribuido**
- Multihilo
- Arquitectura Neutral
- Portable
- Alto desempeño
- Seguro



¿Qué es Java?

- Simple
- Orientado a Objetos
- Distribuido
- **Multihilo**
- Arquitectura Neutral
- Portable
- Alto desempeño
- Seguro



¿Qué es **Java**?

- Simple
- Orientado a Objetos
- Distribuido
- Multihilo
- **Arquitectura Neutral**
- Portable
- Alto desempeño
- Seguro



¿Qué es **Java**?

- Simple
- Orientado a Objetos
- Distribuido
- Multihilo
- Arquitectura Neutral
- **Portable**
- Alto desempeño
- Seguro



¿Qué es **Java**?

- Simple
- Orientado a Objetos
- Distribuido
- Multihilo
- Arquitectura Neutral
- Portable
- **Alto desempeño**
- Seguro



¿Qué es **Java**?

- Simple
- Orientado a Objetos
- Distribuido
- Multihilo
- Arquitectura Neutral
- Portable
- Alto desempeño
- **Seguro**



Origen de Java



Origen de Java

1991



Origen de Java

Gosling



Origen de Java



Ryan Gosling



Origen de Java



James Gosling



Origen de Java



Origen de Java



Origen de Java

2009



Origen de Java

ORACLE®



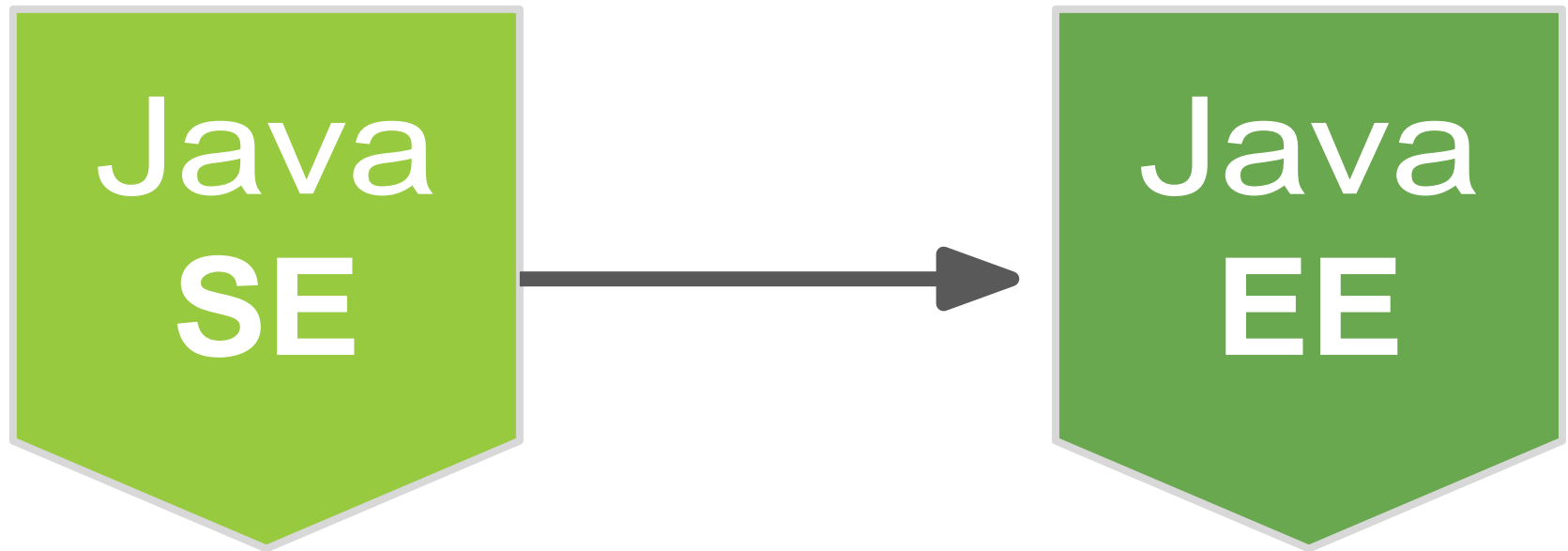
Categorías de Java

Java
SE

Java
EE



Categorías de Java



Categorías de Java

The logo for Java SE is a green shield-shaped icon with a thin grey border. Inside the shield, the words "Java" and "SE" are written in white, stacked vertically.

Java
SE

La base y sintaxis
del lenguaje para
desarrollar
Aplicaciones



Categorías de Java

The logo consists of a green shield with a white border. Inside the shield, the words "Java" and "EE" are written in white, stacked vertically.

Java
EE

Las empresas
trabajen
aplicaciones web
de última
generación



Categorías de Java



Java **Standard Edition**



Herramientas de Desarrollo

Componentes

JDK

Java Development
Kit

JRE

Java Runtime
Environment



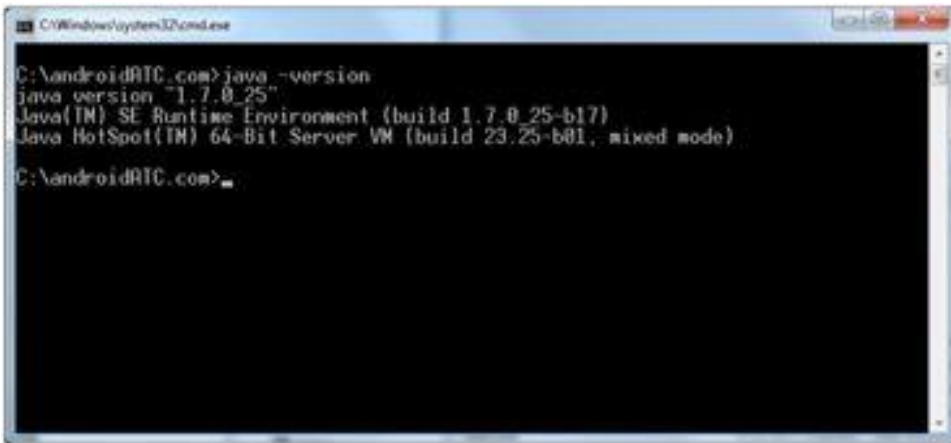
Java Virtual Machine

JRE

Java Runtime
Environment

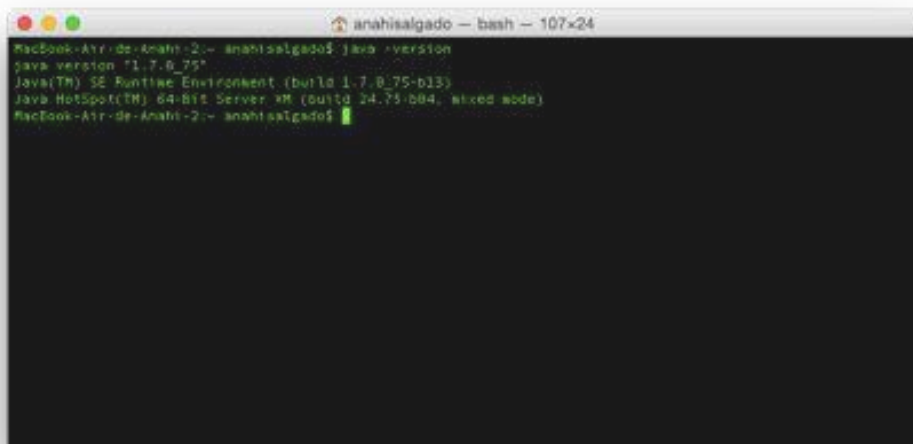


Instalación Java



```
C:\Windows\system32\cmd.exe
C:\androidRTC.com>java -version
java version "1.7.0_25"
Java(TM) SE Runtime Environment (build 1.7.0_25-b17)
Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)
C:\androidRTC.com>
```

Terminal



```
MacBook-Air-de-Anahi-2:~ anahisalgado$ java -version
java version "1.7.0_75"
Java(TM) SE Runtime Environment (build 1.7.0_75-b13)
Java HotSpot(TM) 64-Bit Server VM (build 24.75-b04, mixed mode)
MacBook-Air-de-Anahi-2:~ anahisalgado$
```



Instalación Java

```
C:\Windows\system32\cmd.exe
C:\androidRTC.com>java -version
java version "1.7.0_25"
Java(TM) SE Runtime Environment (build 1.7.0_25-b17)
Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)
C:\androidRTC.com>
```

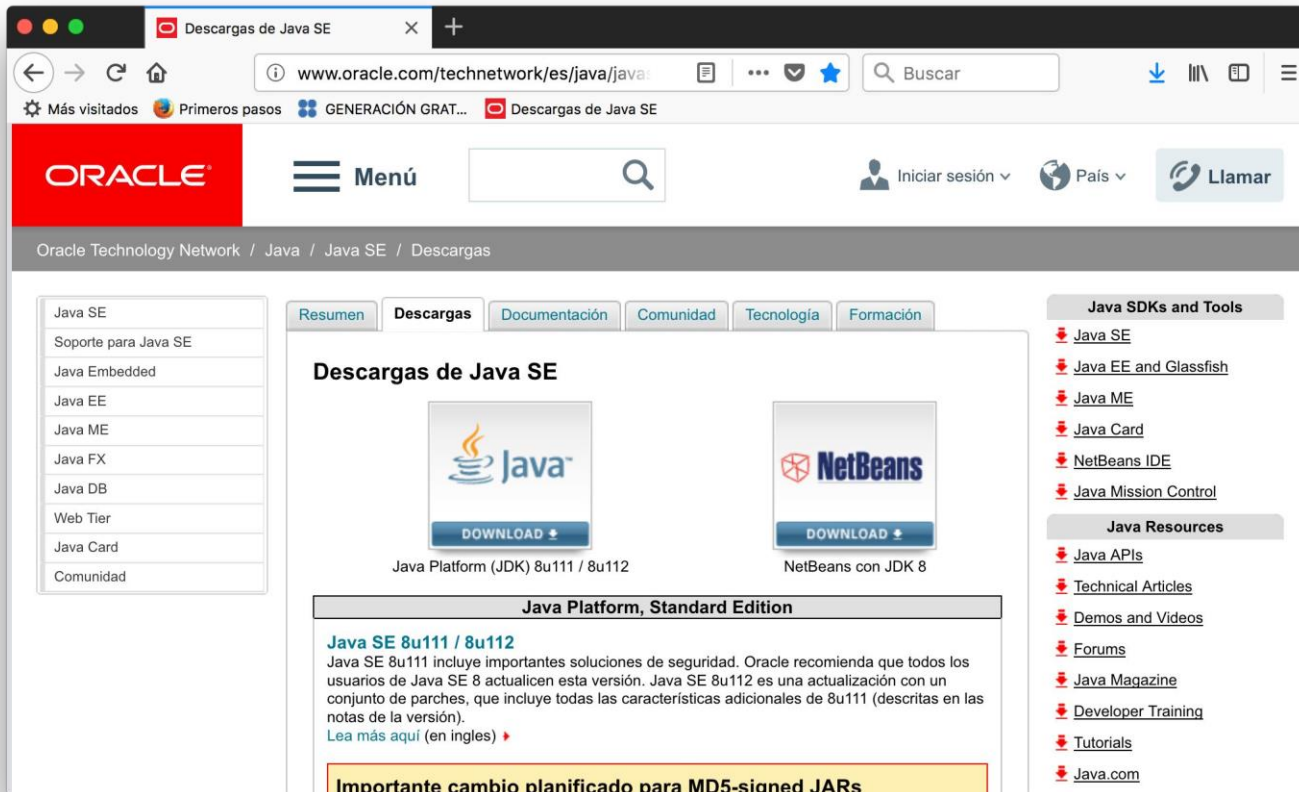
\$ java -version

\$ javac

```
MacBook-Air-de-Anahi-2:~ anahisalgado$ java -version
java version "1.7.0_75"
Java(TM) SE Runtime Environment (build 1.7.0_75-b13)
Java HotSpot(TM) 64-Bit Server VM (build 24.75-b04, mixed mode)
MacBook-Air-de-Anahi-2:~ anahisalgado$
```



Instalación Java



The screenshot shows the Oracle Java SE Downloads page in Spanish. The browser address bar displays www.oracle.com/technetwork/es/java/java.... The page features the Oracle logo and a navigation menu. The main content area is titled "Descargas de Java SE" and includes two download buttons: "Java Platform (JDK) 8u111 / 8u112" and "NetBeans con JDK 8". Below these, there is a section for "Java Platform, Standard Edition" with a link to "Java SE 8u111 / 8u112" and a note about security updates. A yellow banner at the bottom highlights an "Importante cambio planificado para MD5-signed JARs". The right sidebar lists various Java SDKs and tools, including Java SE, Java EE and Glassfish, Java ME, Java Card, NetBeans IDE, and Java Mission Control. The left sidebar provides a navigation menu for Java SE, Java Embedded, Java EE, Java ME, Java FX, Java DB, Web Tier, Java Card, and Comunidad.


Descargas de Java SE

Oracle Technology Network / Java / Java SE / Descargas


Java SE
Soporte para Java SE
Java Embedded
Java EE
Java ME
Java FX
Java DB
Web Tier
Java Card
Comunidad

Resumen Descargas Documentación Comunidad Tecnología Formación

Descargas de Java SE

 **DOWNLOAD**

Java Platform (JDK) 8u111 / 8u112

 **DOWNLOAD**

NetBeans con JDK 8

Java Platform, Standard Edition

Java SE 8u111 / 8u112
Java SE 8u111 incluye importantes soluciones de seguridad. Oracle recomienda que todos los usuarios de Java SE 8 actualicen esta versión. Java SE 8u112 es una actualización con un conjunto de parches, que incluye todas las características adicionales de 8u111 (descritas en las notas de la versión).
[Lea más aquí](#) (en inglés) ▶

Importante cambio planificado para MD5-signed JARs

Java SDKs and Tools

- [Java SE](#)
- [Java EE and Glassfish](#)
- [Java ME](#)
- [Java Card](#)
- [NetBeans IDE](#)
- [Java Mission Control](#)

Java Resources














- [Java APIs](#)
- [Technical Articles](#)
- [Demos and Videos](#)
- [Forums](#)
- [Java Magazine](#)
- [Developer Training](#)
- [Tutorials](#)
- [Java.com](#)

Instalación Java

Java SE Development Kit 8u151

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	77.9 MB	 jdk-8u151-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.85 MB	 jdk-8u151-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.95 MB	 jdk-8u151-linux-i586.rpm
Linux x86	183.73 MB	 jdk-8u151-linux-i586.tar.gz
Linux x64	166.1 MB	 jdk-8u151-linux-x64.rpm
Linux x64	180.95 MB	 jdk-8u151-linux-x64.tar.gz
macOS	247.06 MB	 jdk-8u151-macosx-x64.dmg
Solaris SPARC 64-bit	140.06 MB	 jdk-8u151-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.32 MB	 jdk-8u151-solaris-sparcv9.tar.gz
Solaris x64	140.65 MB	 jdk-8u151-solaris-x64.tar.Z
Solaris x64	97 MB	 jdk-8u151-solaris-x64.tar.gz
Windows x86	198.04 MB	 jdk-8u151-windows-i586.exe
Windows x64	205.95 MB	 jdk-8u151-windows-x64.exe



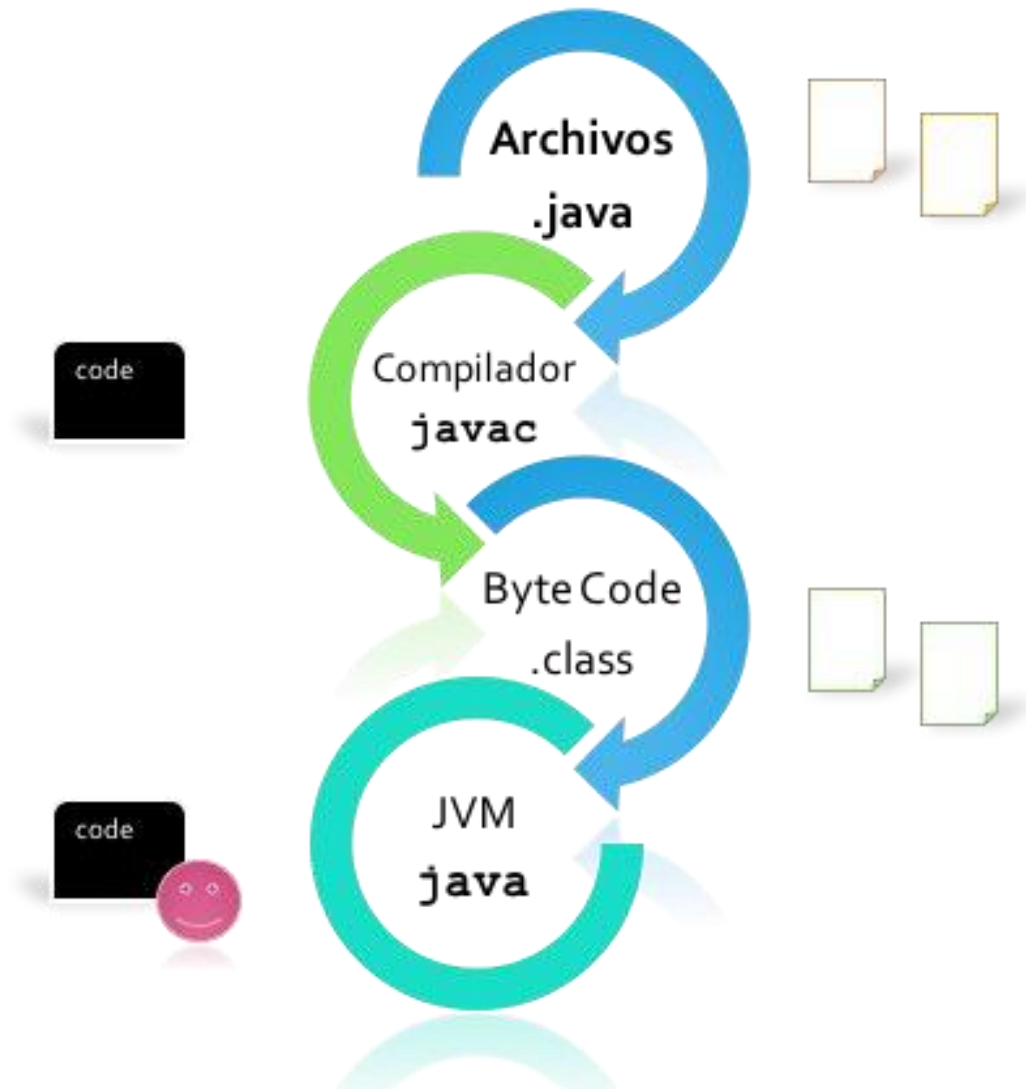
HolaMundo.java

Hola Mundo en Java

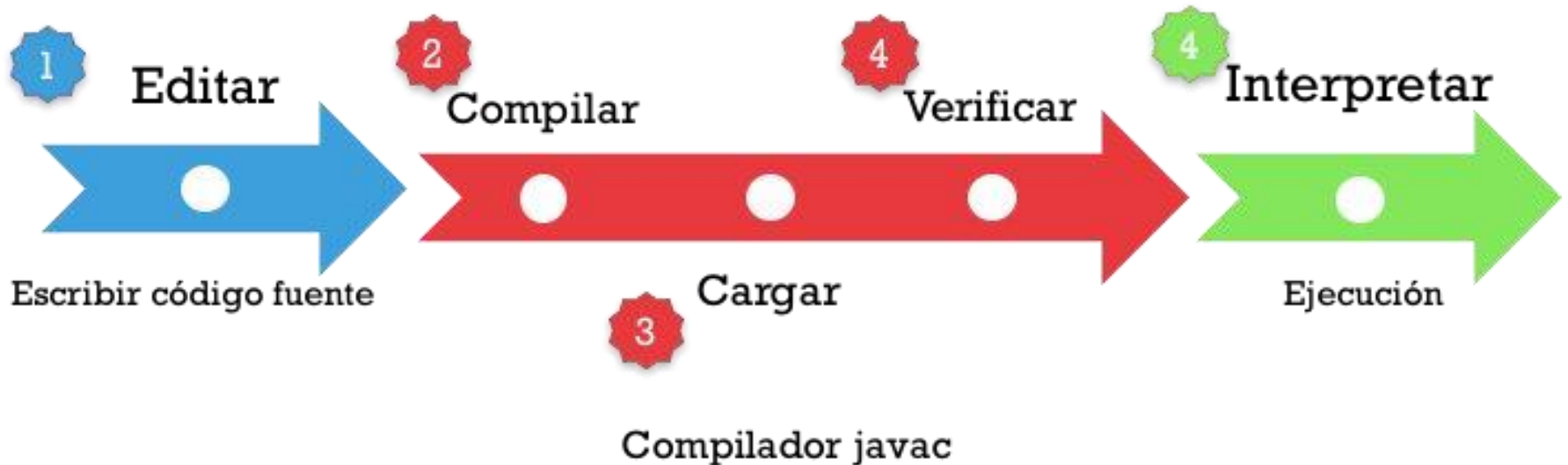
- Editor de código ✓
- Consola de Comando ✓



Fases de la programación



Fases de la programación



Por lo tanto

Java es **Compilado** e
Interpretado



Método **Main**



Método **Main**

Es el **punto de entrada** de una aplicación Java.



Método **Main**

Declara todas las **acciones** realizadas por tu aplicación

```
public static void main (String[] args) {  
    // acciones  
}
```



Método **Main**

Sin él, la **aplicación no se ejecutará**
, regresando el siguiente error:

In class NoMain: void main\$tring
args[]) no está definido.



Hola Mundo en Java

- IDE 

Entorno de Desarrollo Integrado



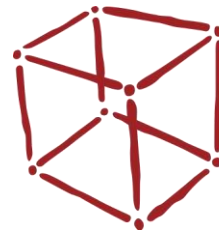
IDE's para Java

Es un entorno de programación que ha sido **empaquetado** como un programa de aplicación.

- Editor de código ✓
- Compilador ✓
- Depurador ✓
- Constructor de interfaz Gráfica ✓



IDE's para Java



NetBeans



eclipse



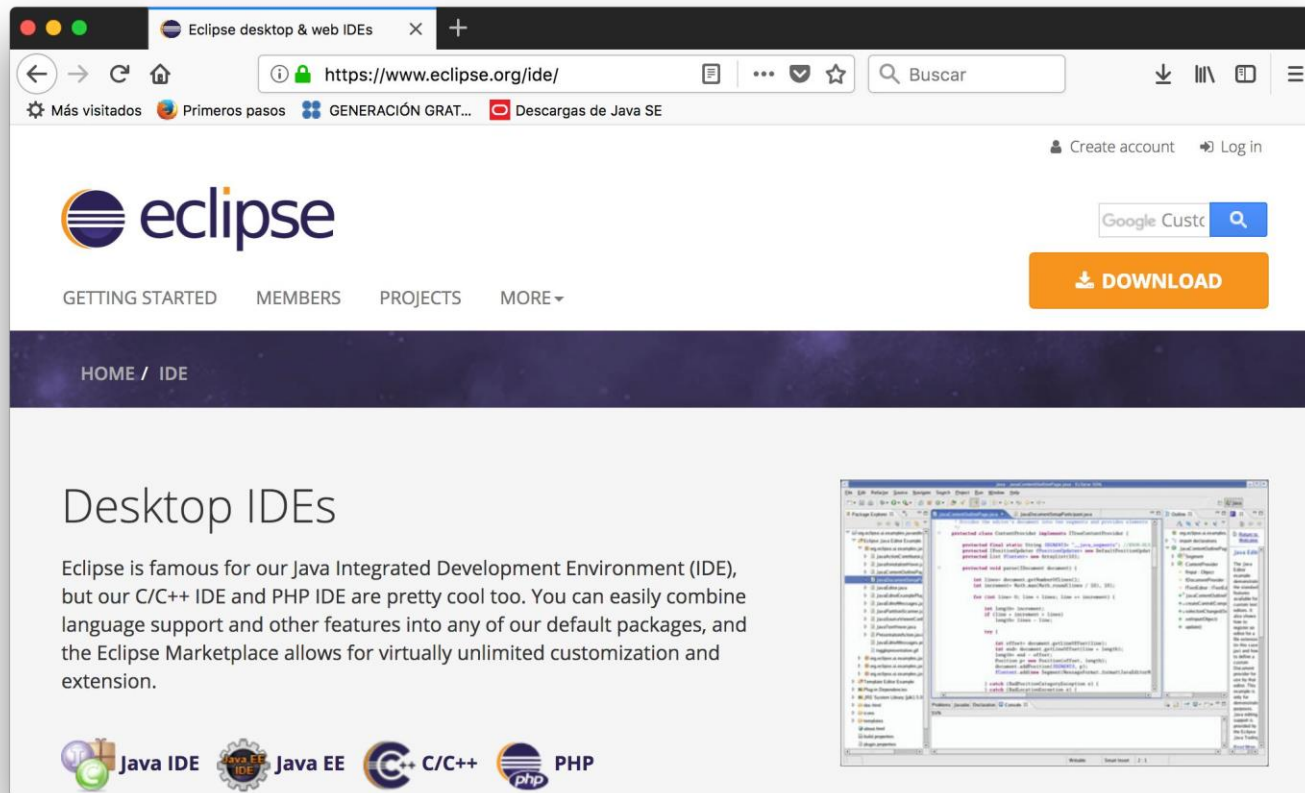
IntelliJ IDEA



IDE's para Java



Hola Mundo en Java



The screenshot shows the Eclipse website with the URL <https://www.eclipse.org/ide/>. The page features the Eclipse logo, navigation links for GETTING STARTED, MEMBERS, PROJECTS, and MORE, and a prominent orange DOWNLOAD button. Below the navigation bar, the section "HOME / IDE" is visible. The main content area is titled "Desktop IDEs" and contains the following text: "Eclipse is famous for our Java Integrated Development Environment (IDE), but our C/C++ IDE and PHP IDE are pretty cool too. You can easily combine language support and other features into any of our default packages, and the Eclipse Marketplace allows for virtually unlimited customization and extension." At the bottom of this section, there are icons for Java IDE, Java EE, C/C++, and PHP. To the right of the text, there is a screenshot of the Eclipse IDE interface showing a code editor with Java code, a project explorer, and a console window.

Desktop IDEs

Eclipse is famous for our Java Integrated Development Environment (IDE), but our C/C++ IDE and PHP IDE are pretty cool too. You can easily combine language support and other features into any of our default packages, and the Eclipse Marketplace allows for virtually unlimited customization and extension.

Java IDE Java EE C/C++ PHP



Tipos de Datos

Variable

Un **espacio de memoria** al que le **asignamos un contenido**, puede ser un valor numérico, de tipo carácter o cadena de caracteres.



Tipos de Datos en Java

**Tipo
Primitivo**

**Tipo
Objeto**



Tipos de Datos en Java

**Tipo
Primitivo**



Enteros

byte

Rango
-128 a 127

1 byte

short

Rango
-32,768 a 32,767

2 bytes

int

Rango
-2,147,483,648 a
2,147,483,647

4 bytes

long

Rango
-9,223,372,036,854,775,808
a
+9,223,372,036,854,775,807

8 bytes



Punto Flotante

float

Rango

1.40129846432481707e-45

a

3.40282346638528860e+38

4 bytes

double

Rango

4.94065645841246544e-324d

a

1.79769313486231570e+308d

8 bytes



Texto

char

Rango
Unicode

2 bytes



Lógicos

boolean

Rango
true o false

2 bytes



Nombres en Java



Nombres en Java

- Sensible a **mayúsculas y minúsculas** ✓
- **Comenzar** con letra, \$ o “_” ✓



Nombres en Java

- Sensible a **mayúsculas y minúsculas** ✓
- **Comenzar** con letra, \$ o “_” ✓



Nombres en Java

- Letras posteriores pueden ser letras, números, \$ y "_" ✓
- Las **constantes** se escriben en **mayúsculas** y contienen "_". ✓



Nombres en Java

- **Letras posteriores** pueden ser letras, números, \$ y "_" ✓
- Las **constantes** se escriben en **mayúsculas** y contienen "_". ✓



Nombres en Java

- Por convención se debe usar la técnica **“camello”**



Nombres en Java



Upper Camel Case
Lower Camel Case



Cast



Cast

En la programación hay
situaciones donde se necesita
cambiar el tipo de dato



Cast

Genera un tipo de dato diferente al original



Cast

- Tipos de datos primitivos



Cast

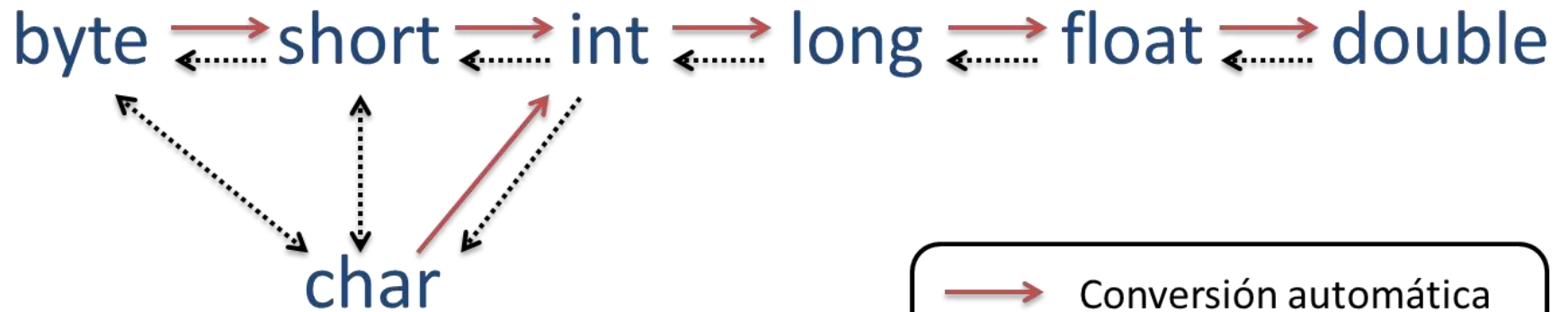
- Tipos de datos primitivos
- Tipos de objetos primitivos.



```
double d = 86.45;  
int i = (int) d;
```



Cast



→ Conversión automática
← Requiere utilizar un cast



Cast

- Se puede realizar el cast para todos los tipos de datos primitivos, con excepción de boolean.



Arreglos

Arrays

Los arreglos se pueden definir como **objetos** en los que podemos guardar **más de una variable**



Arrays



Arrays



Arrays



Arrays



Arrays



1 dimensión



Arrays



Arrays



Arrays



1

2

3

1

2

2 dimensiones



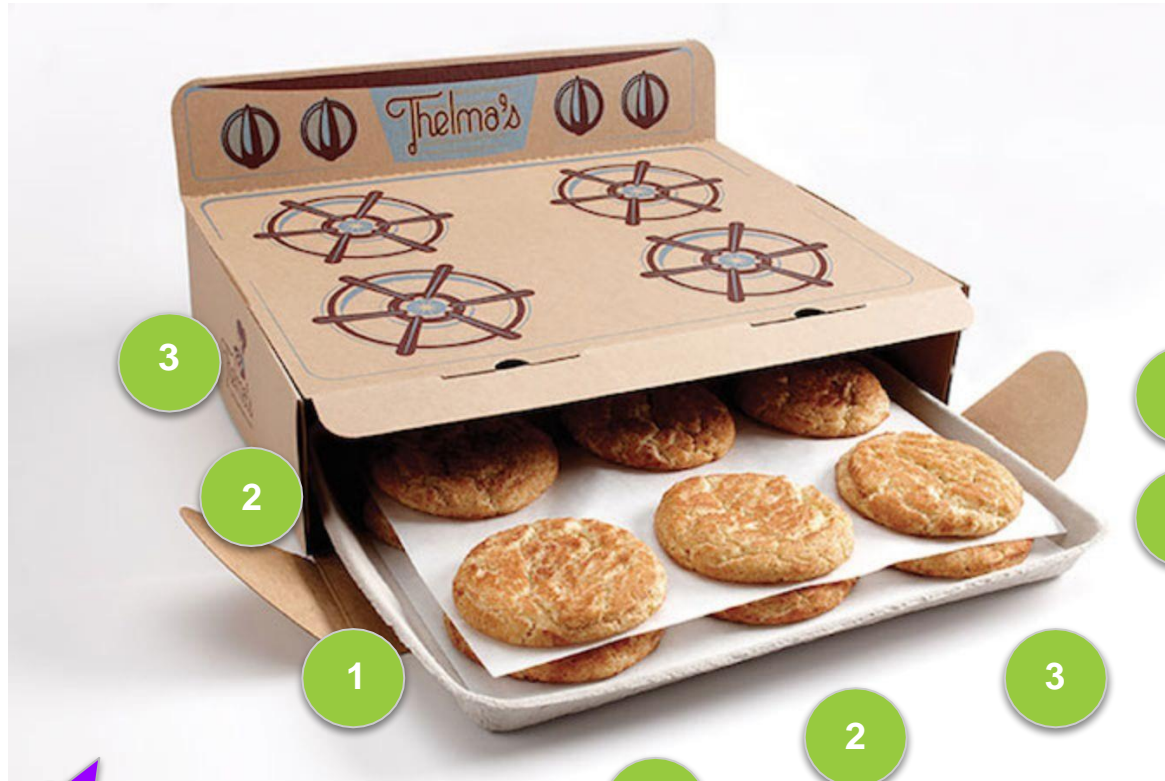
Arrays



3 dimensiones



Arrays



3 dimensiones



Arrays

Más dimensiones

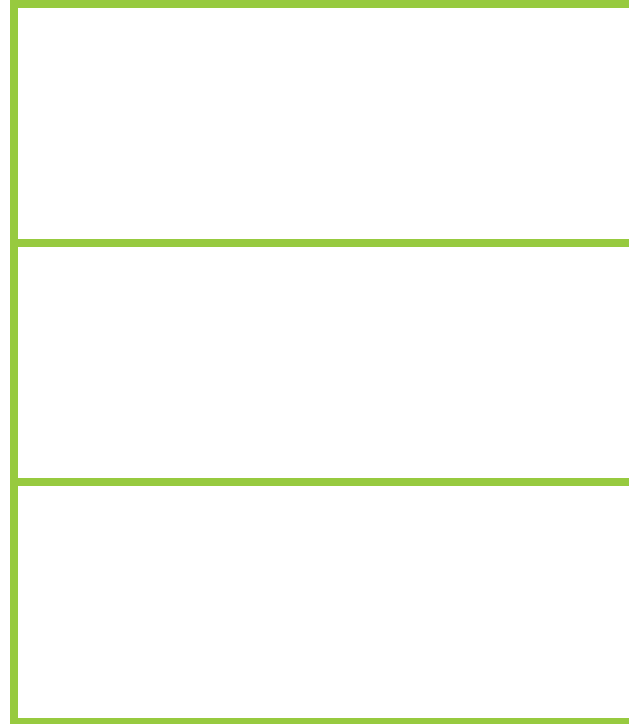


1 dimensión

3

2

1



1



2 dimensiones

3		
2		
1		
	1	2

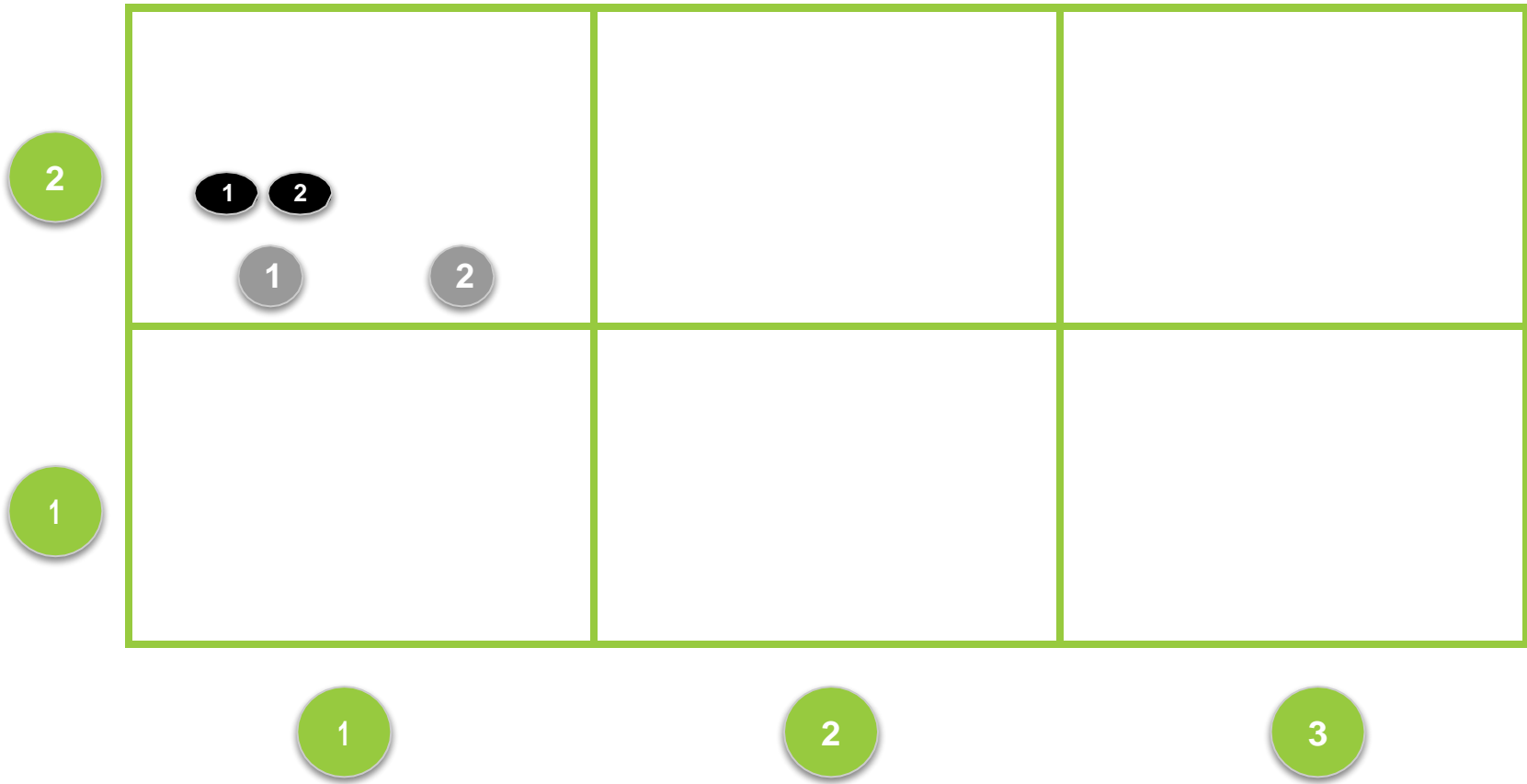


3 dimensiones

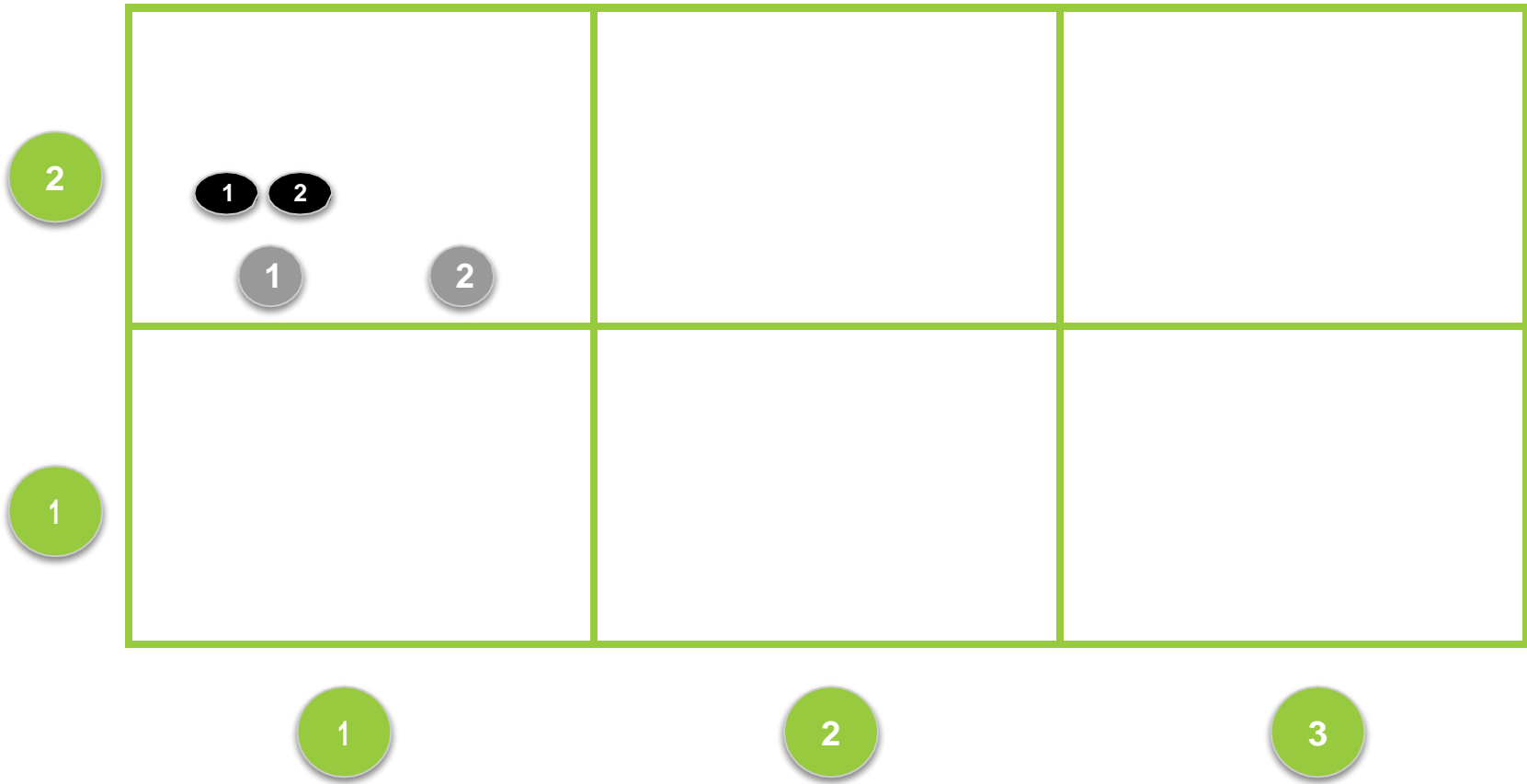
3	1	2	
2			
1			
	1		2



n dimensiones



Arrays



Declarar Arrays

```
TipoDato [ ] nombreVariable;
```



Declarar Arrays

TipoDato nombreVariable[] ;



Definir tamaño Arrays

```
nombreVariable = new TipoDato[capacidad];
```

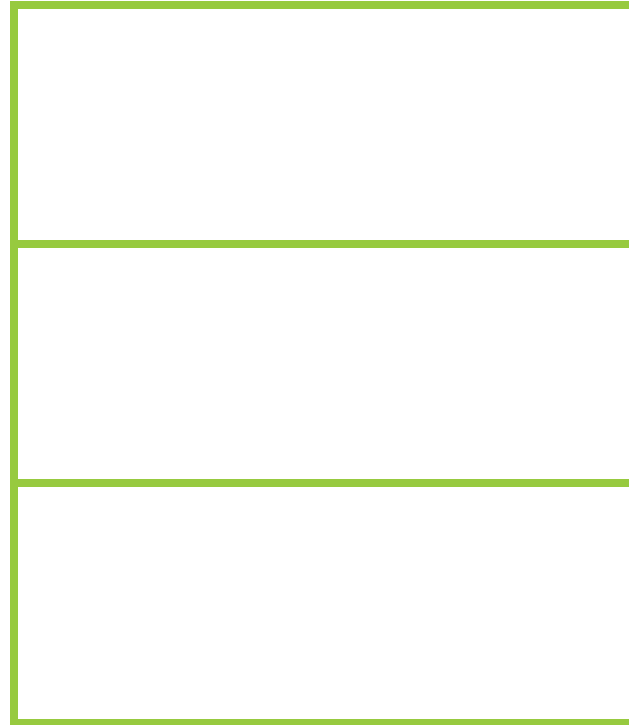


1 dimensión

2

1

0



0



2 dimensiones

2		
1		
0		
	0	1



3 dimensiones

2	0	1	
1			
0			
	0		1



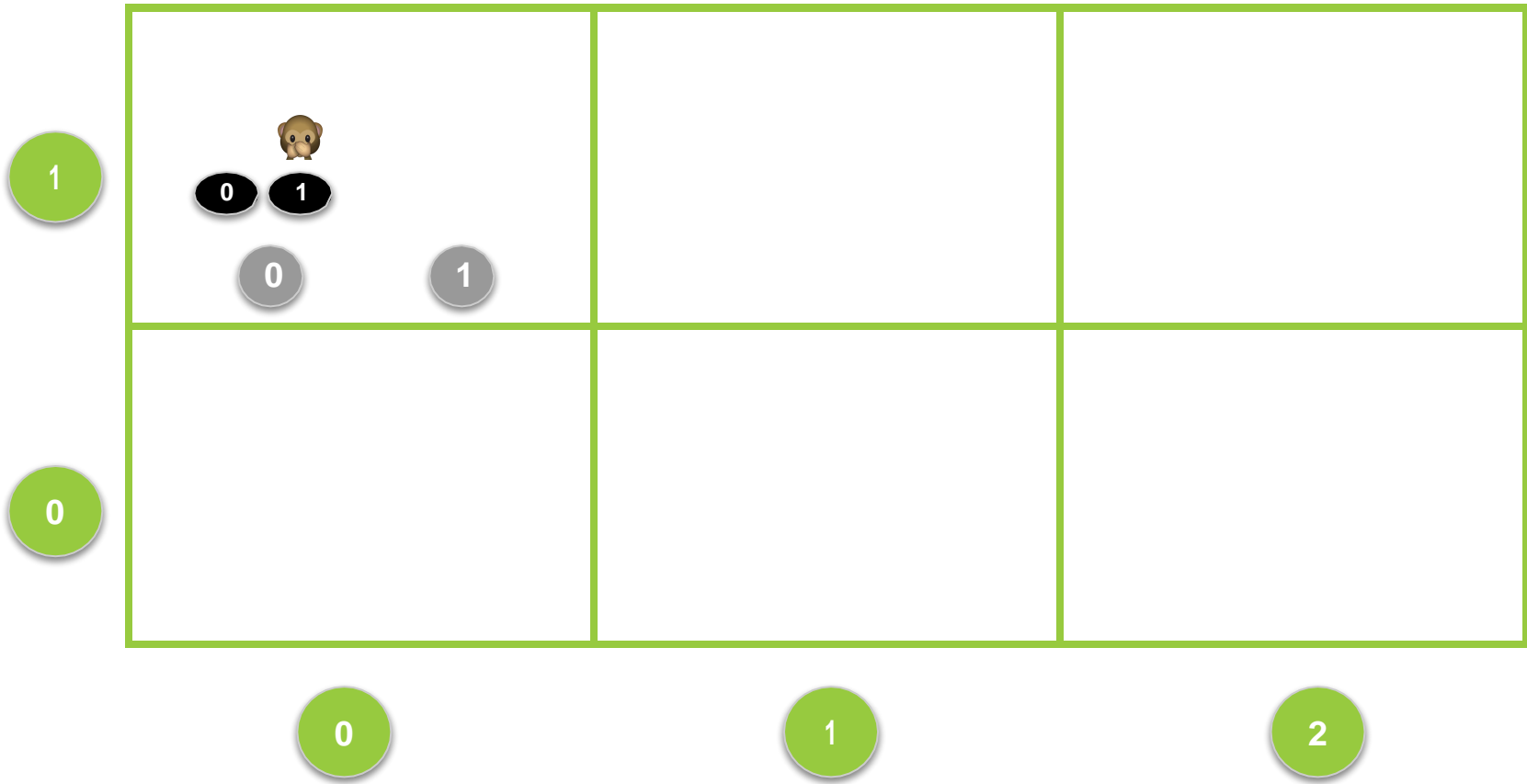
Declarar Arrays



?



Arrays



Arrays



[1][0][0][1]



Asignar valores Arrays

```
nombreVariable[indice] = valor;
```



Operadores



Operadores

Una vez que tenemos variables,
las podemos usar para crear y
formar **expresiones** que **regresen**
valores.



Operadores aritméticos

Operador	Nombre	Ejemplo
+	Adición	$a+b$
-	Substracción	$a-b$
*	Multiplicación	$a*b$
/	División	a/b
%	Módulo	$a\%b$



Operador +

El operador + puede usarse
para agregar o **concatenar**
cadenas



Operadores asignación

Operador	Aplicación	Desglose
<code>+=</code>	<code>a += b</code>	<code>a = a + b</code>
<code>-=</code>	<code>a -= b</code>	<code>a = a - b</code>
<code>*=</code>	<code>a *= b</code>	<code>a = a * b</code>
<code>/=</code>	<code>a /= b</code>	<code>a = a / b</code>
<code>%=</code>	<code>a %= b</code>	<code>a = a % b</code>



Operador incremento decremento

Operador	Nombre	Ejemplo	Desglose
++	incremento	i++	$i = i + 1$
--	decremento o	i--	$i = i - 1$



Prefijo y Postfijo

Prefijo

++

Posfijo

i+



Operador equidad

Operador	Nombre	Ejemplo
==	igualdad	a == b
!=	desigualdad	a != b



Operadores relacionales

Operador	Nombre	Ejemplo
$<$	Menor que	$a < b$
$>$	Mayor que	$a > b$
\leq	Menor o igual que	$a \leq b$
\geq	Mayor o igual que	$a \geq b$



Operadores lógicos

Operador	Nombre	Ejemplo
&&	AND	a && b
	OR	a b
!	NOT	!a



Operadores lógicos

a	b	a && b
f	f	f
f	v	f
v	f	f
v	v	v

AND

a	b	a b
f	f	f
f	v	v
v	f	v
v	v	v

OR

a	!a
f	v
v	f

NOT



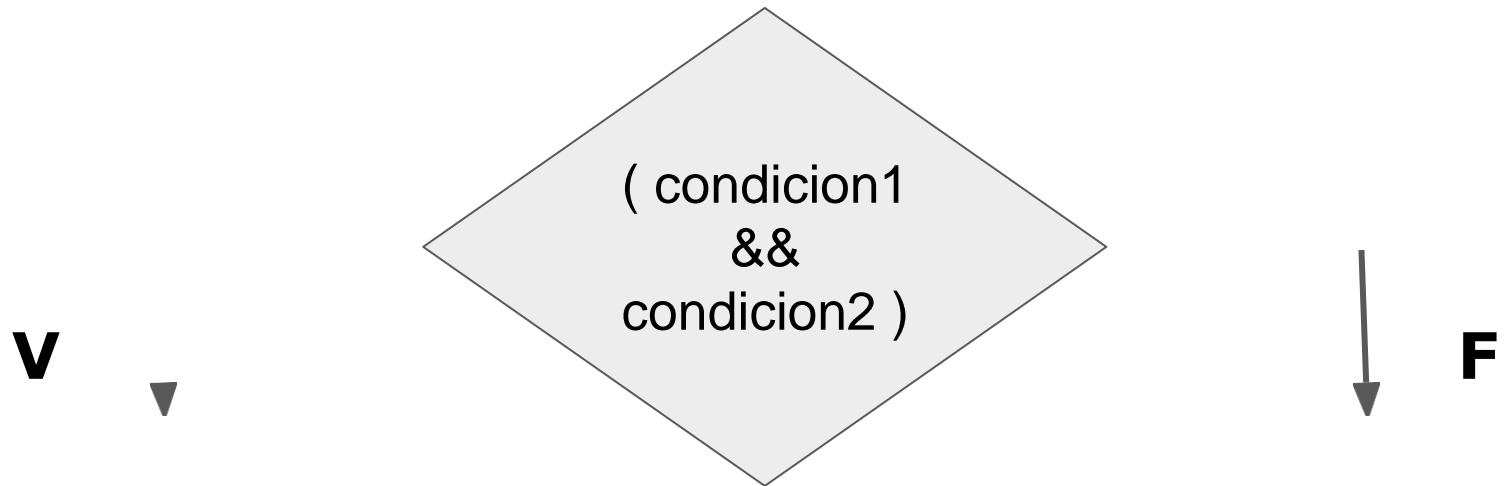
Control de Flujo

Control de Flujo

Podemos controlar el flujo usando sentencias condicionales, ciclos, etc.



If / Else

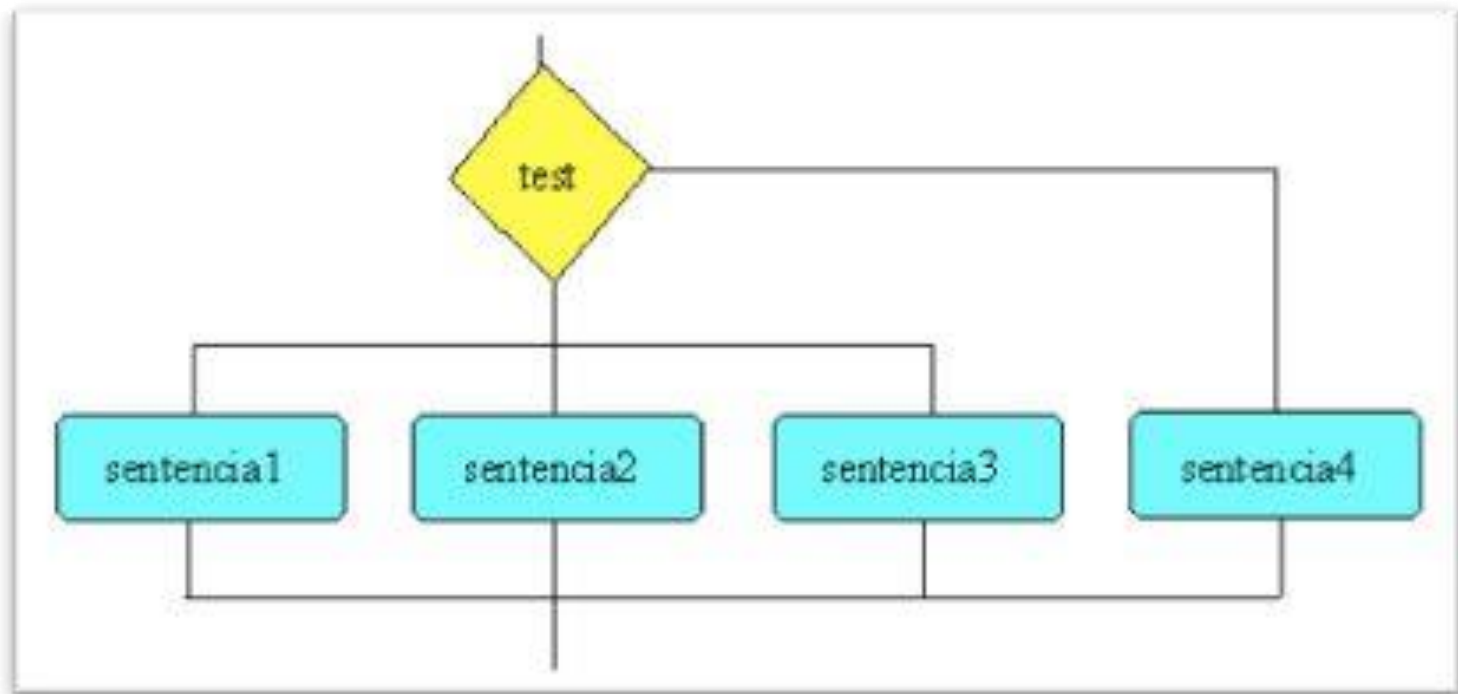


If / Else

```
if (condición) {  
    instrucciones  
} else {  
    instrucciones  
}
```



Switch

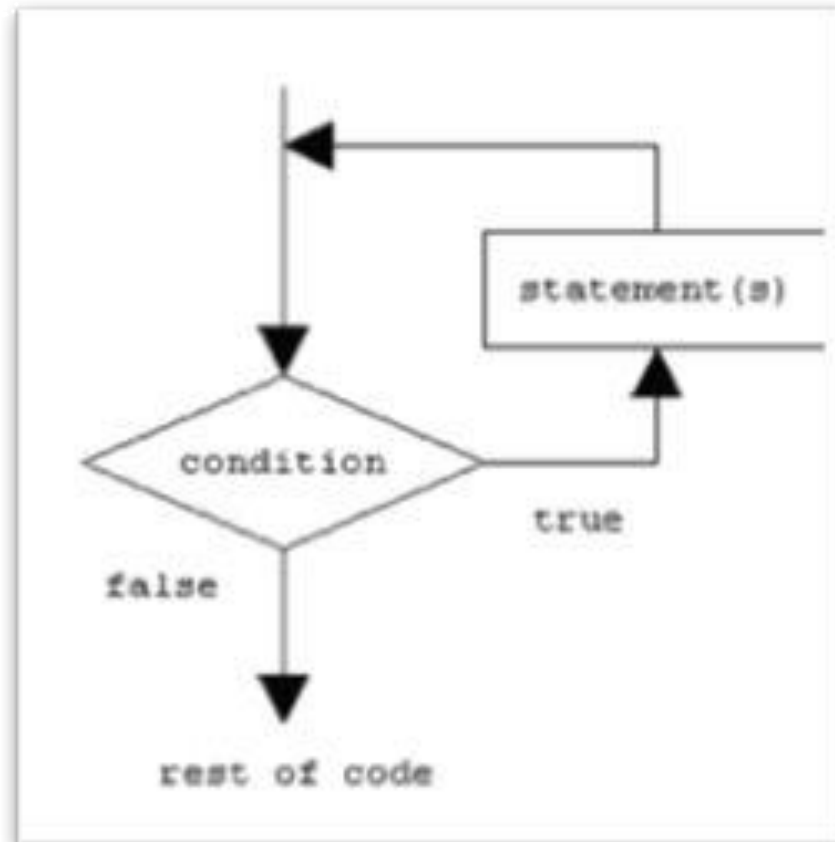


Switch

```
switch(a){  
    case valor1:  
        break;  
    case valor2:  
        break;  
    default:  
        Break;  
}
```



While



While

```
while (condicion){  
  
    //instrucciones  
  
}
```



Do While

```
do {
```

```
//instrucciones
```

```
} while (condicion);
```



For

```
for (inicializacion; condicion; incremento) {  
    //instrucciones  
}
```



Foreach

```
for ( TipoDato elemento : coleccion ) {  
  
    //Instrucciones  
  
}
```



Break, Continue y Return



Programación Orientada a Objetos POO

Programación Orientada a Objetos

Una nueva **forma de pensar**



Programación Orientada a Objetos

Se trata de **descomponer un problema** en subproblemas y más **subproblemas**



Programación Orientada a Objetos

Definir un Dominio del Problema
PROBLEM DOMAIN



Programación Orientada a Objetos

Recopilación de requisitos del cliente y tener por escrito un alcance

¿Qué queremos lograr?



Programación Orientada a Objetos

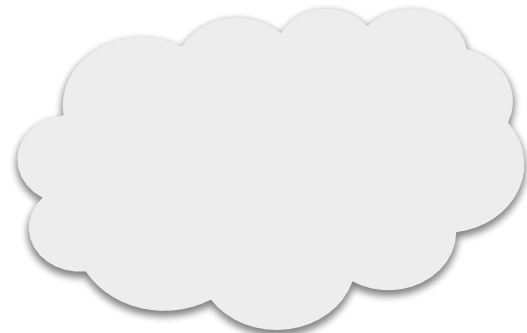


Ver un problema como un **escenario** del problema y tratar de **simularlo con objetos**



Identificando Objetos

Pueden ser **Físicos** o **Conceptuales**



Identificando Objetos

Tienen **atributos** (características)

- tamaño
- nombre
- forma
- representan el estado del objeto



Identificando Objetos

Los **nombres** y **atributos** de los
objetos por lo general son
sustantivos

user, session



Identificando Objetos

Las operaciones suelen ser verbos o
sustantivo y verbo

login, makeReport



Objeto



Objeto

- Auto



Atributos:

- matricula
- marca
- modelo



Objeto

- Auto



Atributos:

- matricula
- marca
- modelo

Comportamientos:

- arrancar
- frenar
- reversa



Objeto



POO

- Auto



Atributos:

- matricula
- marca
- modelo
- precio
- vendido

Comportamientos:

- mostrarDatos
- esVendido



Clase

Una Clase es la forma en cómo **defines tu objeto** para generar más objetos



Clase

Las Clases son descriptivas son
plantillas



Clase

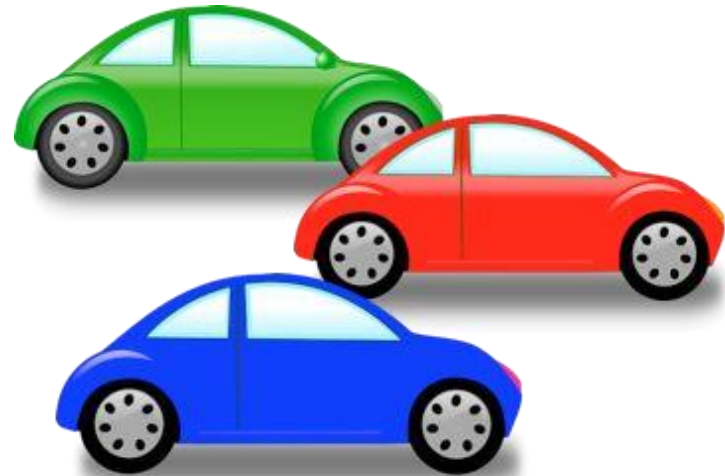
- Auto

atributos

- id
- matricula
- marca
- año
- precio
- vendido

comportamientos

- mostrarDatos
- esVendido



Estructura de una Clase

```
public class Auto {
```

atributos

```
int id;  
String matricula  
String marca  
String modelo  
double precio  
boolean vendido
```

comportamientos

```
public void mostrarDatos() {  
}  
public boolean esVendido() {  
}
```

```
}
```



Classes

Movie
+ id: int + title: String + genre: String + creator: String + duration: int + year: short + viewed: boolean + timeViewed: int
+ see() + getters(): type + setters(type)

Serie
+ id: int + title: String + genre: String + creator: String + duration: int + year: short + viewed: boolean + sessionQuantity: int + chapters: Chapter
+ getters(): type + setters(type)

Chapter
+ id: int + title: String + duration: int + year: short + viewed: boolean + timeViewed: int +sessionNumber: int
+ see() + getters(): type + setters(type)



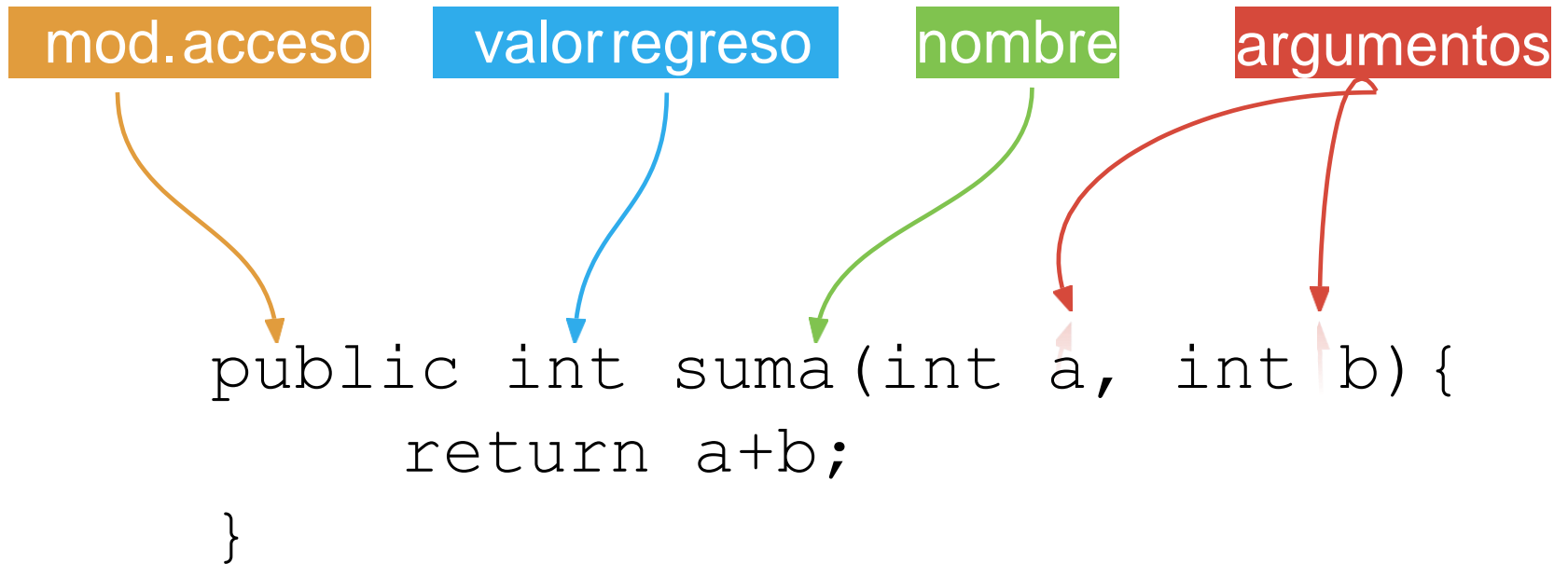
Classes

Book
+ id: int + title: String + editionDate: Date + editorial: String + autores: String[] + isbn: String + readed: boolean + timeReaded: int
+ getters + setters + read

Magazine
+ id: int + title: String + editionDate: Date + editorial: String + autores: String[]
+ getters + setters



Métodos



Métodos

Tiene un **valor de regreso** explícitamente invocado en su cuerpo usando la palabra reservada `return`.





Métodos

No regresa ningún valor si es declarado **void**.

No puede declararse **dentro de otro método**.



Declarando un objeto

Auto	miAuto;
	
Tipo de Objeto	Nombre del Objeto



Instanciando un objeto

```
miAuto = new Auto ( ) ;
```

Nombre del Objeto

Creando el objeto



Declarando e instanciando un objeto

```
Auto miAuto = new Auto ( ) ;
```


Declarando el objeto

Instanciando/Creando
el objeto



Método Constructor

```
miAuto = new Auto ( ) ;
```


Método Constructor



Método Constructor

Crea nuevas **instancias** de una clase.

Tiene el **mismo nombre** que la clase que inicializa.

Usa la palabra reservada **new** para invocarlo.



Método Constructor

Usa **cero o más argumentos** contenidos dentro de los paréntesis que siguen al nombre.

No regresa un valor.



Utilizando el objeto

```
Auto miAuto = new Auto ();  
miAuto.marca = "Ferrari";  
miAuto.mostrarDatos ();
```



Static

Final

Sobrecarga

Modificadores de Acceso

Static

Accesando a Métodos

```
Auto miAuto = new Auto();  
miAuto.mostrarDatos();
```

```
Math.random();
```

```
Math.sqrt(25);
```

```
Math.PI;
```



Métodos static

Se puede usar en toda la clase

Está definido por la palabra reservada
static.



Métodos static

Puede ser **accesado indicando el nombre de la clase**, la notación punto y el nombre del método.



Métodos static

Se invoca en una clase que **no tiene instancias de la clase.**



Métodos static

```
public class Calculadora {
```

```
    public static int suma(int a, int b) {  
        return a+b;  
    }
```

```
}
```

```
Calculadora.suma(5,2);
```



Métodos static

Puede ser invocado en una clase
que **no tiene instancias de la
clase.**



Miembros static

```
public class Calculadora{
```

```
    public static final double PI = 3.1415926
```

```
    public static int valor = 0;
```

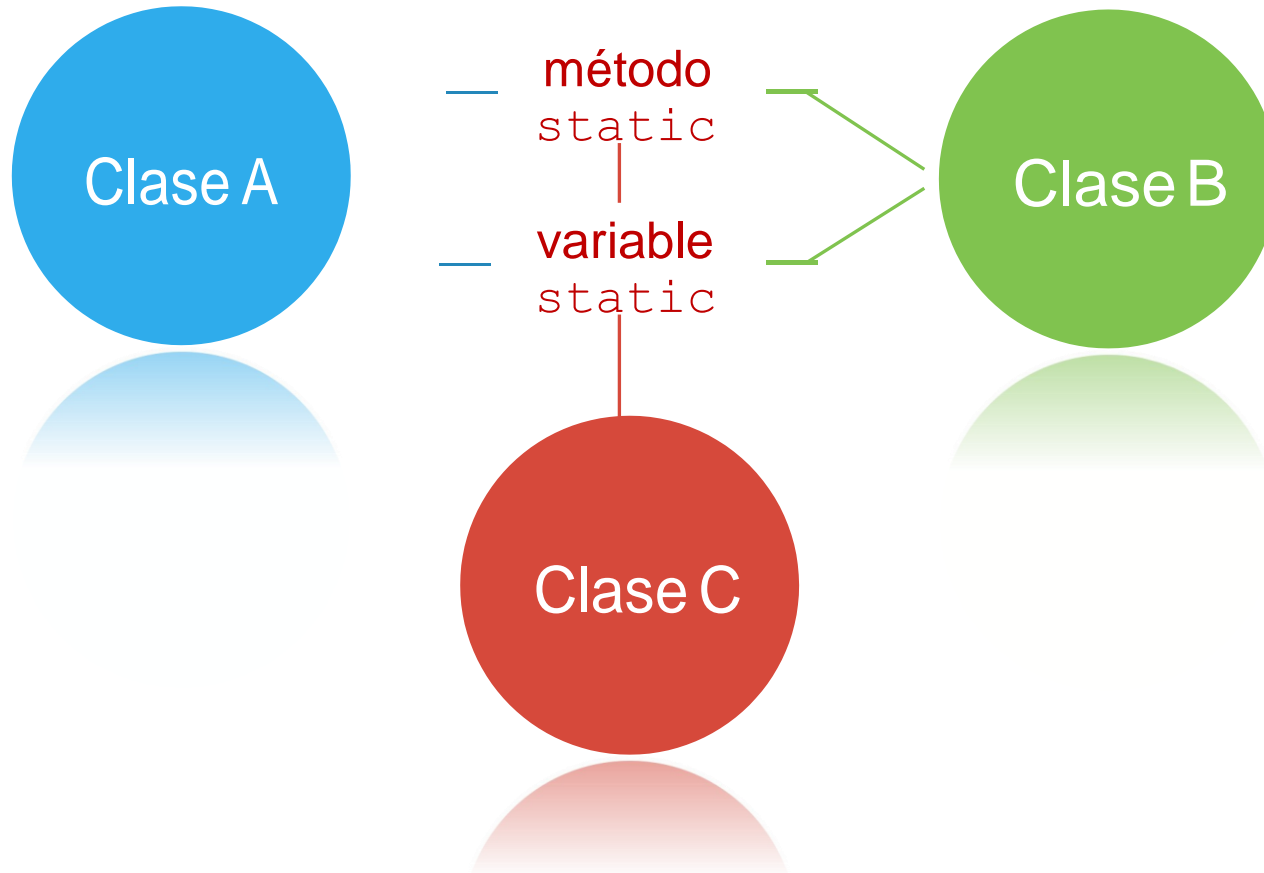
```
}
```

```
Calculadora.PI;
```

```
Calculadora.valor;
```



Miembros static



Miembros static

```
import static com.anncode.operaciones.Calculadora.*  
import static java.lang.Math.*;
```

```
public class Principal{
```

```
    public static void main(String[] args){  
        System.out.println(suma(3, 5));  
        System.out.println(PI);  
    }
```

```
}
```



Final

Miembros final

```
public class Calculadora{
```

```
    public static final double PI = 3.1415926
```

```
}
```

```
Calculadora.PI;
```



Sobrecarga

Sobrecarga

A veces necesitamos que dos o más métodos **tengan el mismo nombre pero con diferentes argumentos**



Sobrecarga

```
public class Calculadora{
```

```
    public int suma(int a, int b){  
        return a+b;  
    }
```

```
    public float suma(float a, float b){  
        return a+b;  
    }
```

```
    public float suma(int a, float b){  
        return a+b;  
    }
```

```
}
```



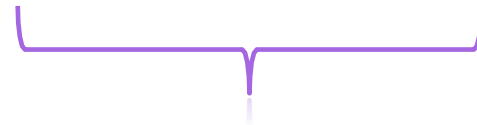
Sobrecarga de Constructores

La sobrecarga de constructores se
usa para **inicializar objetos**



Sobrecarga de Constructores

```
Auto miAuto = new Auto ( ) ;
```


Método Constructor



Sobrecarga de Constructores

```
public class Auto {  
    int id;  
    String matricula  
    String marca  
    String modelo  
    double precio  
    boolean vendido  
    public Auto() {  
  
    }  
  
    public Auto(int id, String matricula, String precio) {  
        this.id = id;  
        this.matricula = matricula;  
        this.precio = precio;  
    }  
}
```



Modificadores de Acceso

Sobrecarga de Constructores

Modificador	Clase	Package	Subclase	Otros
public	✓ <input type="checkbox"/>	✓ <input type="checkbox"/>	✓ <input type="checkbox"/>	✓ <input type="checkbox"/>
protected	✓ <input type="checkbox"/>	✓ <input type="checkbox"/>	✓ <input type="checkbox"/>	● <input type="checkbox"/>
default	✓ <input type="checkbox"/>	✓ <input type="checkbox"/>	● <input type="checkbox"/>	● <input type="checkbox"/>
private	✓ <input type="checkbox"/>	● <input type="checkbox"/>	● <input type="checkbox"/>	● <input type="checkbox"/>



Getters y Setters

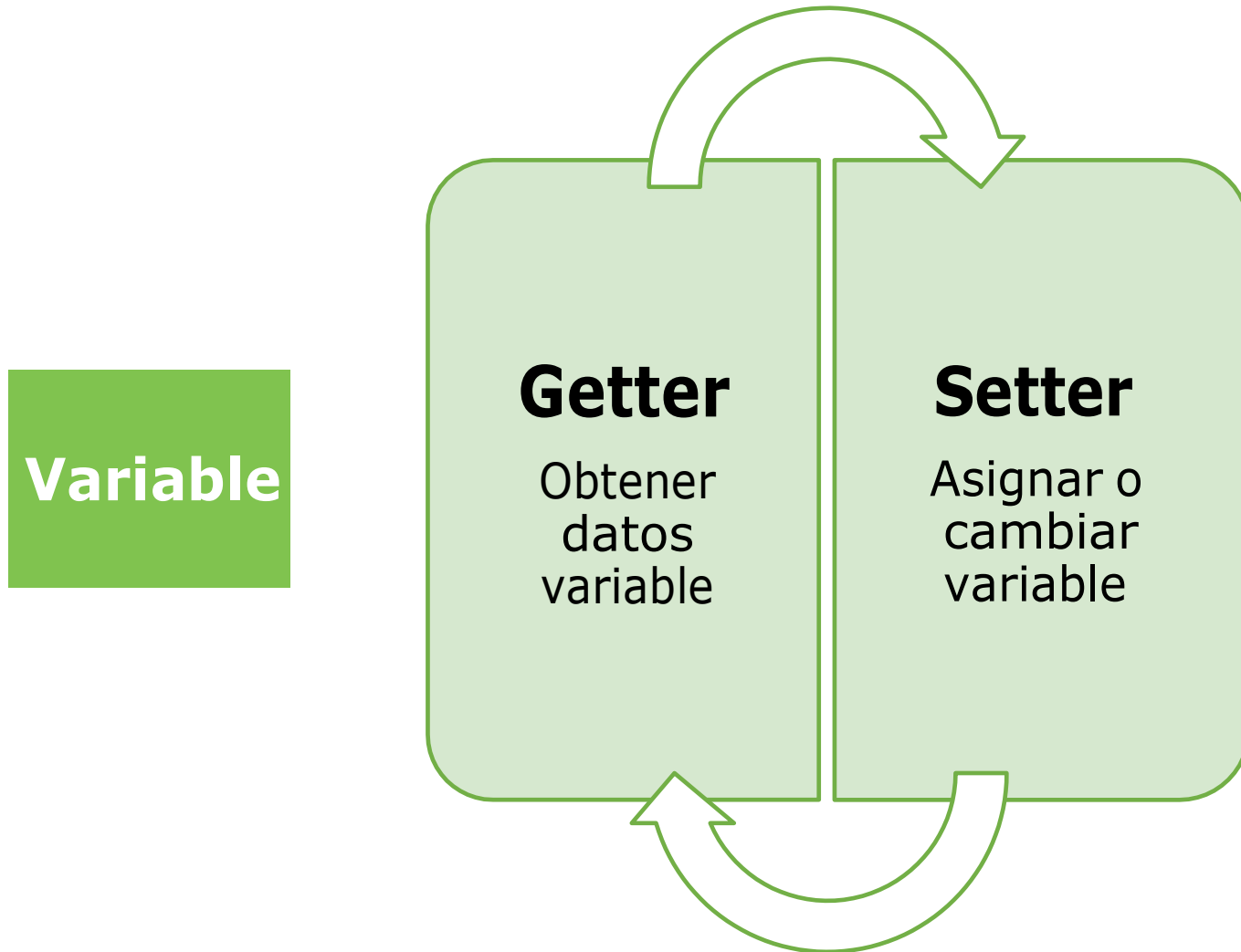
Getters y Setters

Leer/Escribir

específicamente los valores
de las **variables** miembro.



Getters y Setters



Variables \neq Objetos

Classes Wrapper / Objeto primitivo

Byte

Short

Integer

Long

Float

Double

Characer

Boolean

String



Variables \neq Objetos

Variables son entidades elementales (**muy sencillas**)

Un número

Un carácter

Un valor verdadero
falso

Objetos son entidades **complejas** que pueden estar formadas por la **agrupación de muchas variables y métodos.**

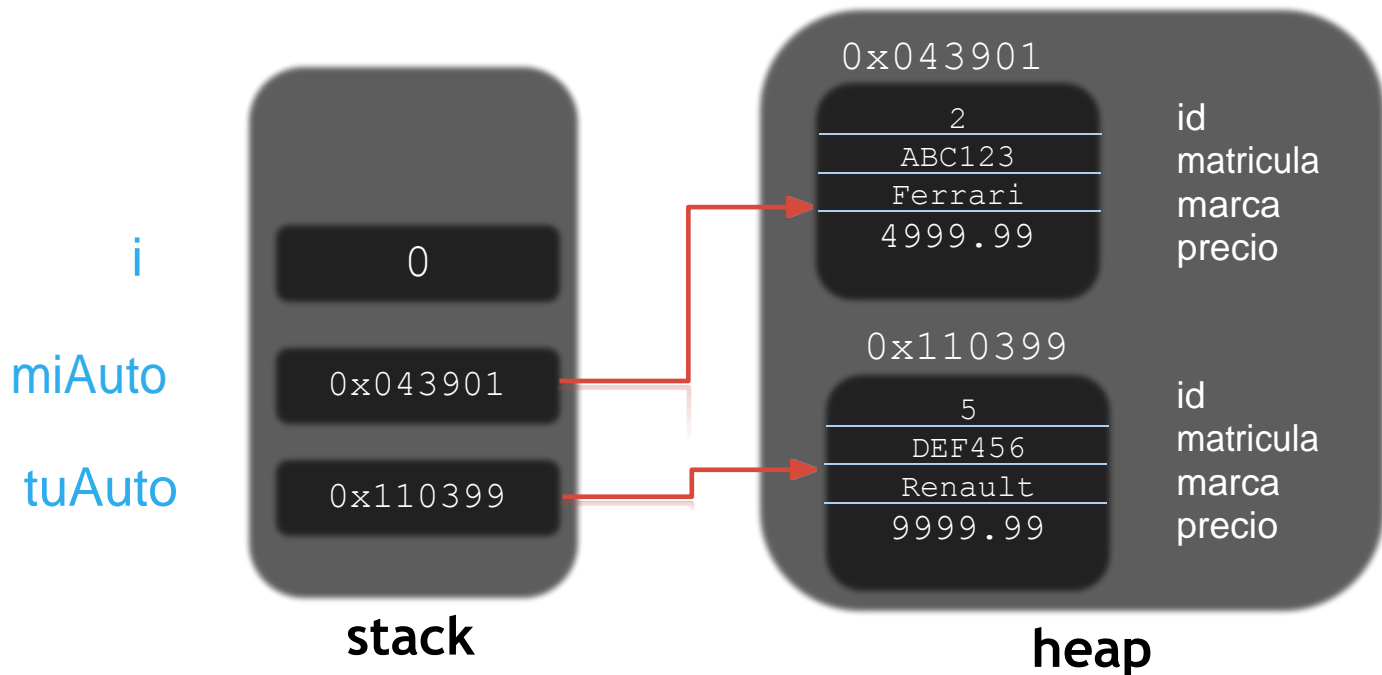


Variables \neq Objetos

```
int i = 0;
```

```
Auto miAuto = new Auto();
```

```
Auto tuAuto = new Auto();
```



Herencia
Polimorfismo
Interfaces

Herencia

Herencia

En algunas circunstancias, es necesario **utilizar el estado y comportamiento** de una clase en conjunto con **otras clases**.



Herencia



Herencia

Se establece una **relación padre-hijo** entre dos objetos diferentes.

La idea de la herencia es permitir la creación de **nuevas clases** basadas en **clases existentes**.



Herencia

Movie
+ id: int + title: String + genre: String + creator: String + duration: int + year: short + viewed: boolean + timeViewed: int
+ see() + getters(): type + setters(type)

Serie
+ id: int + title: String + genre: String + creator: String + duration: int + year: short + viewed: boolean + sessionQuantity: int + chapters: Chapter
+ getters(): type + setters(type)

Chapter
+ id: int + title: String + duration: int + year: short + viewed: boolean + timeViewed: int +sessionNumber: int
+ see() + getters(): type + setters(type)



Herencia

Book
<ul style="list-style-type: none">+ id: int+ title: String+ editionDate: Date+ editorial: String+ autores: String[] <ul style="list-style-type: none">+ isbn: String+ readed: boolean+ timeReaded: int
<ul style="list-style-type: none">+ getters+ setters+ read

Magazine
<ul style="list-style-type: none">+ id: int+ title: String+ editionDate: Date+ editorial: String+ autores: String[]
<ul style="list-style-type: none">+ getters+ setters



Herencia

```
public class Film {
```

SúperClase

```
}
```

```
public class Movie extends Film {
```

Subclase

```
}
```



Herencia

Una **subclase** hereda todos los **miembros de su súper clase** que están declarados como **public** o **protected**.



super y this

super

Indica que una variable o un método es de la clase Padre (superclase)

this

Permite especificar que la variable que esta señalando (`this.nombreVariable`) es de la misma clase en la que se usa.



Sobreescritura

Sobreescritura

Cuando una clase hereda de otra, y en esta **clase hija se redefine un método** con una implementación distinta a la de la clase padre



Sobreescritura

Los métodos marcados como final o static no se pueden sobrescribir.



Sobreescritura de Constructores

Un constructor en una subclase usando los miembros heredados de la superclase, **con argumentos diferentes.**



Sobreescritura de Constructores

```
/** constructor de una subclase */  
public Subclase(parámetros...) {  
    // invoca el constructor de la superclase  
    super(parámetros para la superclase);  
    // inicializa sus atributos  
    ...  
}
```



Polimorfismo

Polimorfismo

Posibilidad de construir **varios métodos con el mismo nombre**, pero con relación a la clase a la que pertenece cada uno, con **comportamientos diferentes**.



Interfaces

Interfaces

Es un tipo de referencia similar a una clase que podría contener solo **constantes**, **definiciones de métodos**, métodos con modificador de acceso default.



Interfaces

Se establece la forma de una clase (nombres de métodos, listas de argumentos y tipos de retorno, pero **no bloques de código**).



Interfaces

```
public interface IVisualizable {
```

```
    Date startTo See(Date dateI);
```

```
    void stopToSee(Date dateI, Date dateF);
```

```
}
```

```
public class Movie extends Film
```

```
implements IVisualizable {
```

```
}
```



Colecciones de Datos

Colecciones

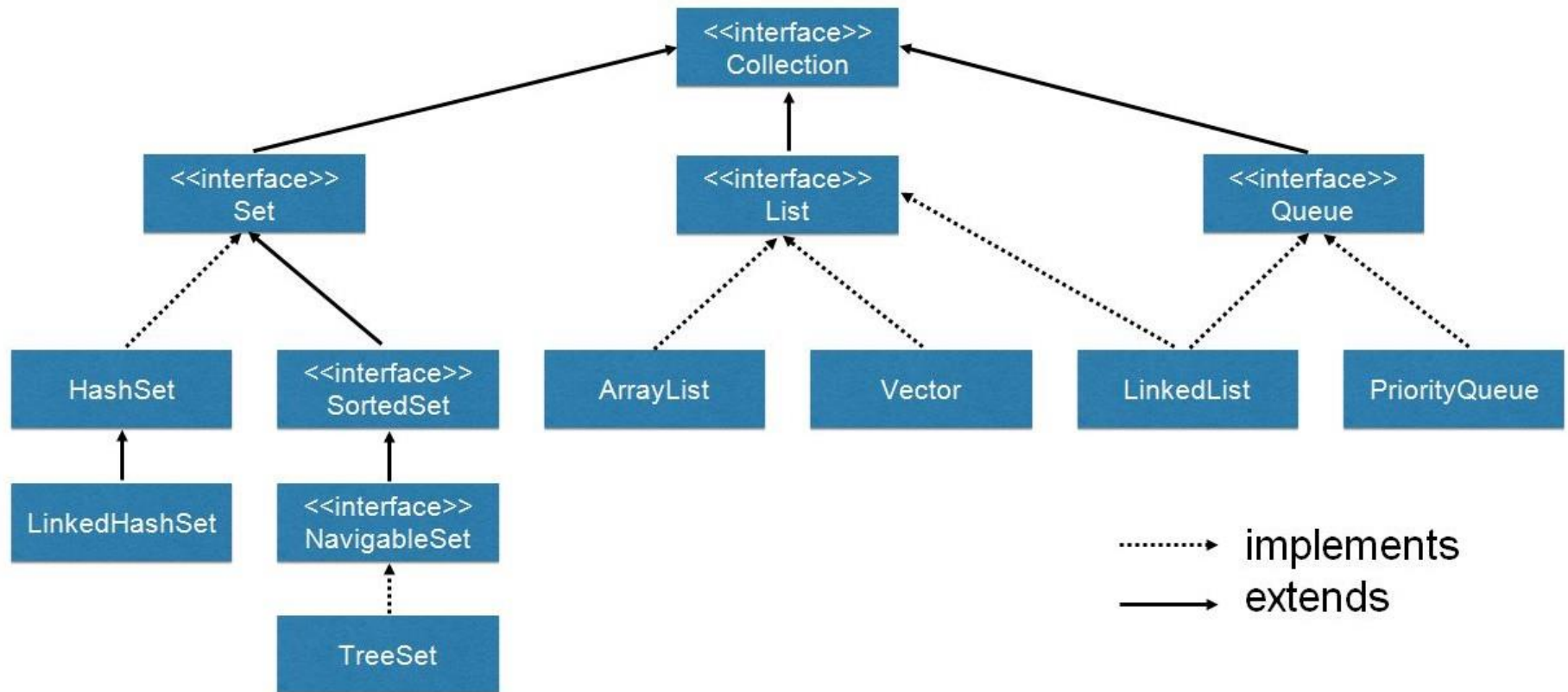
Se **diferencian de los arrays** en que **su tamaño no es fijo**

son dinámicos

Se pueden realizar operaciones de añadir, eliminar, obtener, encontrar o recorrer una colección.



Colecciones



List

Es una interfaz que extiende de la interfaz Collection, se utiliza para almacenar colecciones de objetos, proviene del paquete **java.util**



List

Colecciones **ordenadas** y
con **elementos repetidos**



List

add(Object o): añade un objeto al final de la lista.

add(int indice, Object o): añade un objeto a la lista en la posición indicada.



List

get(int indice): devuelve el objeto de la lista de la posición indicada.

remove(int indice): elimina el objeto de la lista pasado por parámetro.



List

clear(): elimina todos los elementos de la lista.

indexOf(Object o): devuelve la posición de la primera vez que un elemento coincida con el objeto pasado por parámetro. Si el elemento no se encuentra devuelve -1.



List

lastIndexOf(Object o):

devuelve la posición de la última vez que un elemento coincida con el objeto pasado por parámetro. Si el elemento no se encuentra devuelve -1.

size(): devuelve el número de elementos de la lista.



List

contains(Object o)

devuelve verdadero si en la lista aparece el objeto pasado por parámetro, para lo cual utiliza intrínsecamente el método equals()

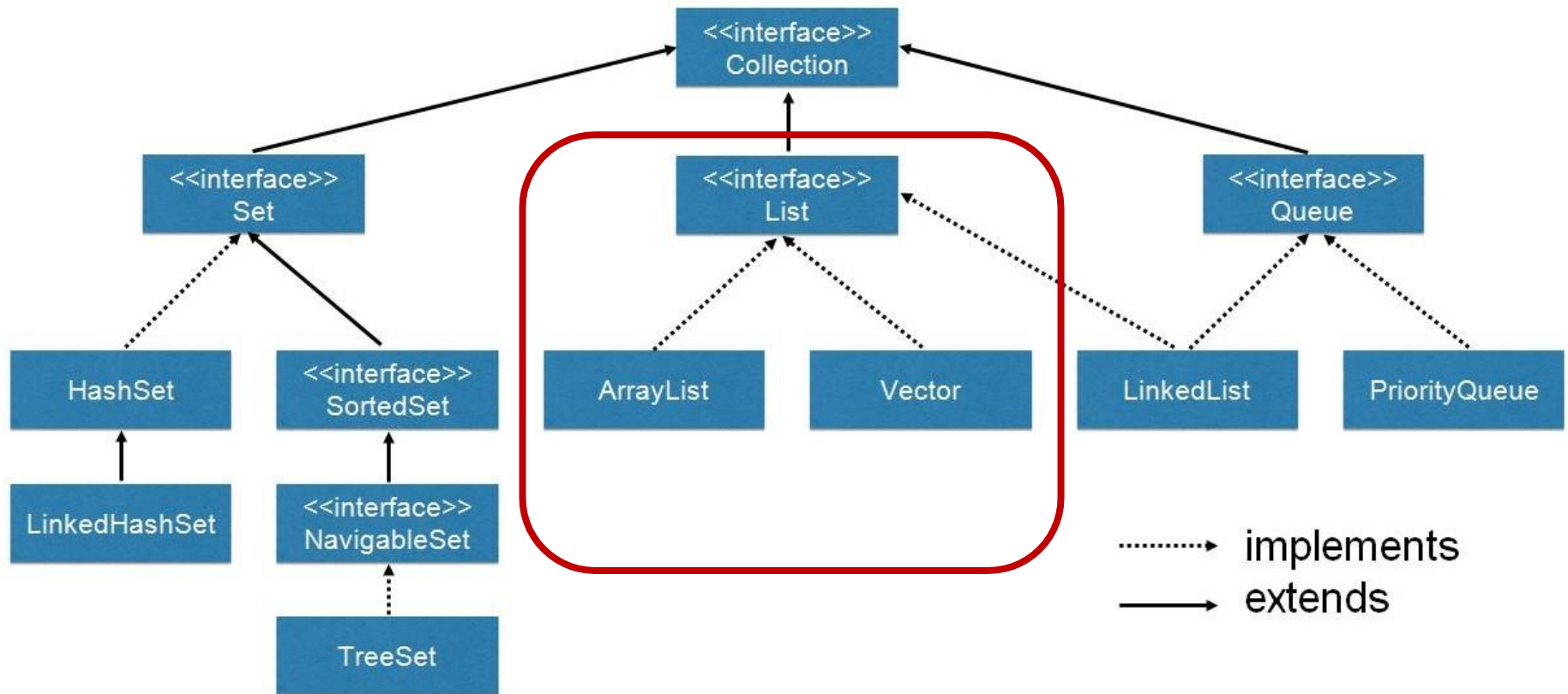


Set

Puede estar o no
ordenada no tiene
elementos repetidos



Colecciones



ArrayList

Almacena un arreglo de objetos que **puede cambiar de tamaño**, tiene una capacidad que crece automáticamente.



ArrayList

```
ArrayList<String> androids = new ArrayList<String>();
```

```
androids.add("Jelly Bean");
```

```
androids.add("Kit Kat");
```

```
androids.add("Lollipop");
```



Vector

Un vector es similar a un array, la diferencia estriba en que un vector usa **hilos** y **está sincronizado** y un array no



Java I/O

Java SE

Básico



Modularización

Java 9

Modularización

Proyecto **Jigsaw** 5años~



Modularización

Encapsulación



Modularización

Encapsulación

Detalles ocultos que provocan una
interfaz más linda



Modularización

Capas de Software



Modularización

El CLASSPATH generalmente se mantiene como una variable de entorno dentro de Windows, indicando el directorio en donde tenemos nuestros archivos JAR



Modularización

- Agrupación de código y recursos como los **JARs**.
- **Un descriptor** que restringe el acceso a sus paquetes, y describe sus dependencias.

