

Desarrollo Web Integrado

Sesión 2

“Características de Java”



Universidad
Tecnológica
del Perú

Comentemos sobre lo acontecido

¿Qué vimos la clase pasada?

Introducción

Arquitectura Java

- WORA:
Write Once Run Anywhere
- JDK
- JDR
- JVM
- JIT
- BYTCODE
- JAR

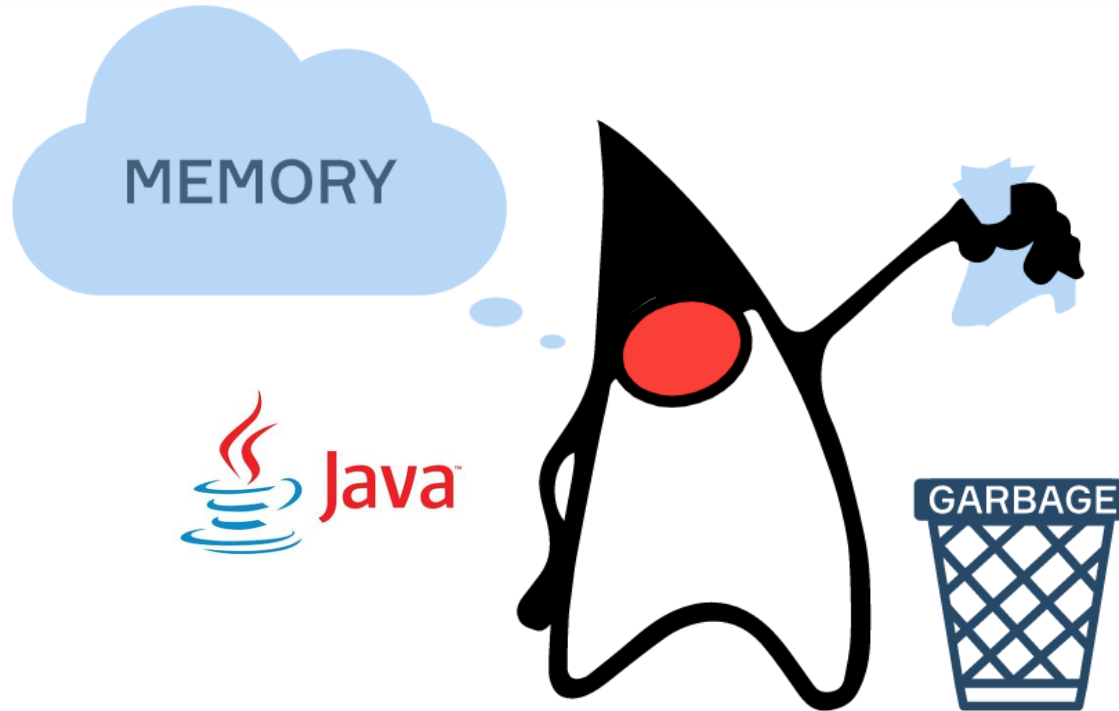


Levanta la mano para participar por audio o chat



Universidad
Tecnológica
del Perú

¿Qué idea les transmite la imagen sobre Java?



Fuente: Shutterstock



Levanta la mano para participar por audio o chat

Logro de la sesión:



Universidad
Tecnológica
del Perú

Al finalizar la sesión, el estudiante identifica las características de Java con énfasis en el manejo de memoria



Desaprende lo que te limita

Temario de la sesión



Universidad
Tecnológica
del Perú

- BYTECODE
- Manejo de memoria: STACK y HEAP
- Garbage Collector



Desaprende lo que te limita

¿Por qué crees que será importante aprender el tema el día de hoy?

Levanta la mano para participar por audio o chat

Tema: Características de Java

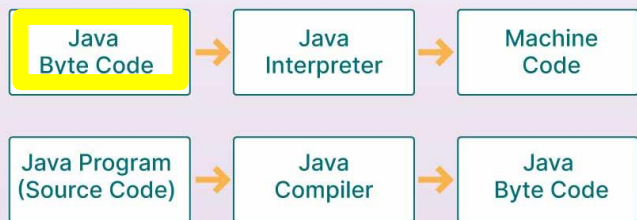
- BYTECODE
- Manejo de memoria: STACK y HEAP
- GARBAGE COLLECTOR

Veamos ...

¿En que consisten las aplicaciones Java?



Difference Between Machine Code and Byte Code



*Una aplicación de Java consiste de una
secuencia de instrucciones ByteCode*

ByteCode

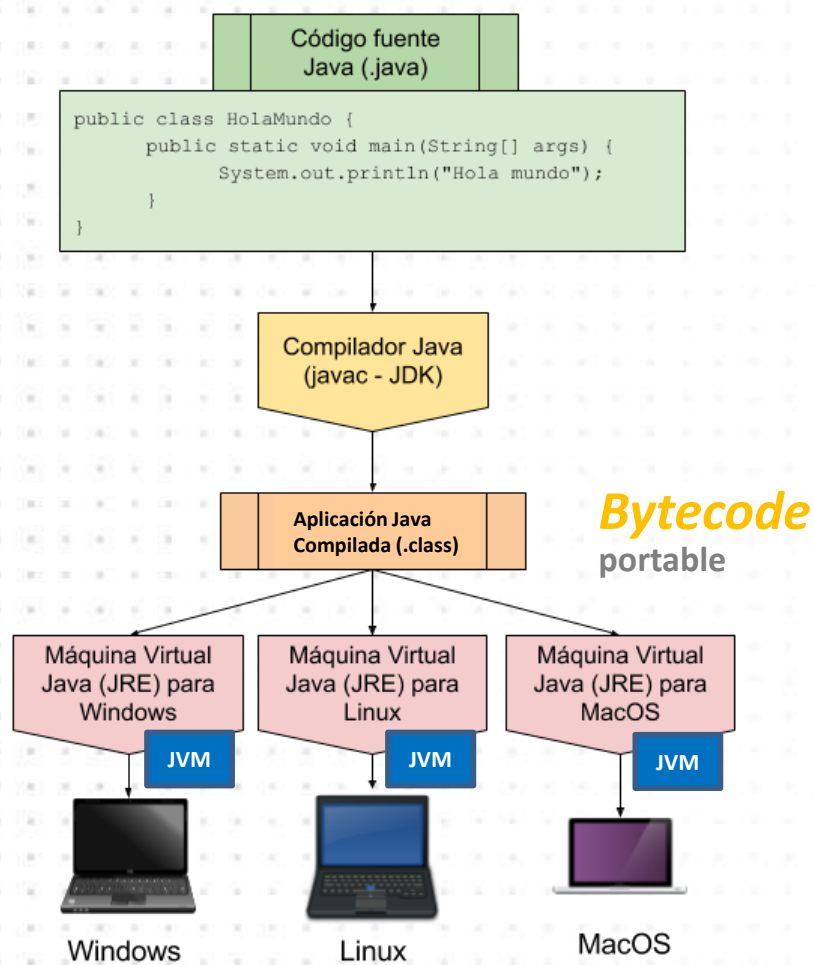
*Código
intermedio*

Java: .class

- Cada código de operación tiene una longitud de un byte
- Cada instrucción tiene un código de operación entre 0 y 255 seguido de parámetros tales como los registros o las direcciones de memoria

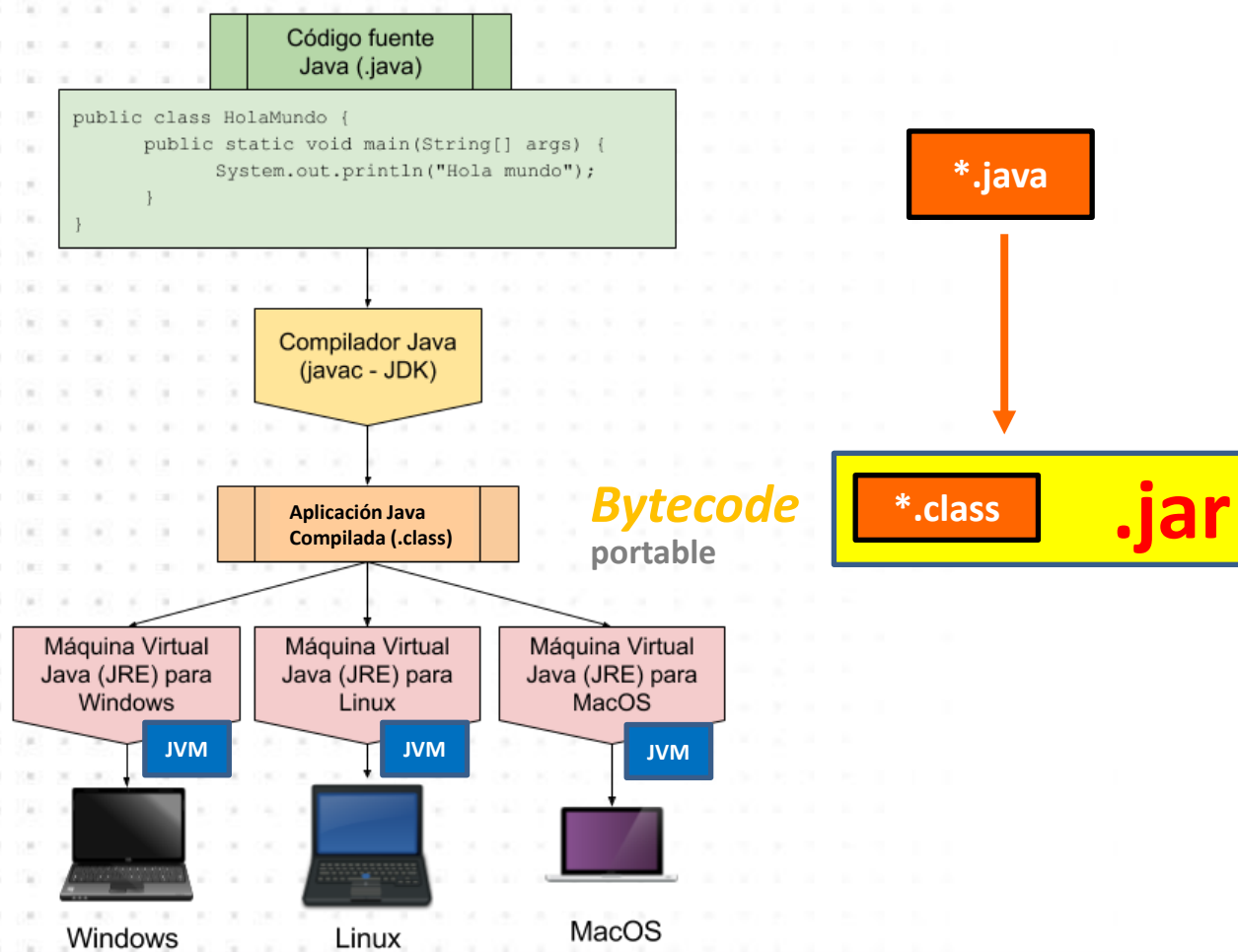
¿Cómo se construyen las instrucciones ByteCode para una aplicación Java?





***.java**

***.class**



MyApp.jar

+-- META-INF/

+-- MANIFEST.MF

+-- com/

+-- example/

+-- app/

+-- Main.class

+-- Utils.class

+-- config/

+-- app.properties

+-- images/

+-- logo.png

Java **AR**chive



Universidad
Tecnológica
del Perú

Un archivo JAR (Java ARchive) es un archivo comprimido que contiene todos los componentes necesarios para ejecutar una aplicación Java

Clases: Contienen el código compilado.

MANIFEST.MF: Metadatos sobre el JAR, incluyendo el punto de entrada.

Recursos: Archivos adicionales necesarios por la aplicación.

META-INF: Contiene `MANIFEST.MF` y posibles firmas digitales.

MANIFEST.MF

Manifest-Version: 1.0

Main-Class: com.example.app.Main

Indicaciones

Generar las instrucciones bytecode para un código fuente escrito en el lenguaje de programación Java



Practiquemos

Tiempo: 15 minutos

Terminado el tiempo, cada grupo regresa a la sala principal y expone

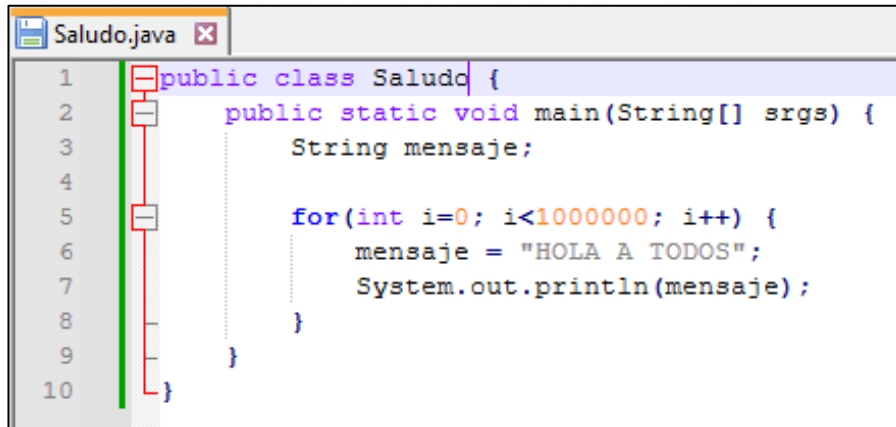
¿Cómo generamos la secuencia de instrucciones ByteCode en un archivo .class?

```
Saludo.java
1 public class Saludo {
2     public static void main(String[] srgs) {
3         String mensaje;
4
5         for(int i=0; i<1000000; i++) {
6             mensaje = "HOLA A TODOS";
7             System.out.println(mensaje);
8         }
9     }
10 }
```



javac Saludo.java

¿Cómo generamos la secuencia de instrucciones ByteCode en un archivo .jar?



```
Saludo.java
1 public class Saludo {
2     public static void main(String[] args) {
3         String mensaje;
4
5         for(int i=0; i<1000000; i++) {
6             mensaje = "HOLA A TODOS";
7             System.out.println(mensaje);
8         }
9     }
10 }
```



```
jar cvmf MANIFIEST.MF Saludo.jar Saludo.class
```




Test

Manejo de Memoria

Archivo: Dato.java

```
public class Dato {  
    private int A;  
    private int B;  
  
    public int getA() {  
        return A;  
    }  
  
    public void setA(int A) {  
        this.A = A;  
    }  
  
    public int getB() {  
        return B;  
    }  
  
    public void setB(int B) {  
        this.B = B;  
    }  
}
```

```
public class Test {  
    public static void main(String[] args) {  
        Dato d1 = new Dato();  
        d1.setA(10);  
        d1.setB(20);  
  
        Dato d2 = new Dato();  
        d2.setA(30);  
        d2.setB(40);  
  
        Dato d3 = new Dato();  
        d3.setA(50);  
        d3.setB(60);  
  
        d1 = d3;  
        d3.setB(45);  
        System.out.println(Integer.toString(d2.getA() + d1.getB()));  
    }  
}
```



Universidad
Tecnológica
del Perú



¿Qué valor imprime?

Veamos ...

¿Cómo organiza Java la memoria?



principalmente

...

STACK and HEAP

1. Llamadas a funciones
(Call functions -> Call Stack)
2. Variables de tipo básico
3. Variables de tipo referencia
(Referencias a objetos)

1. Objetos



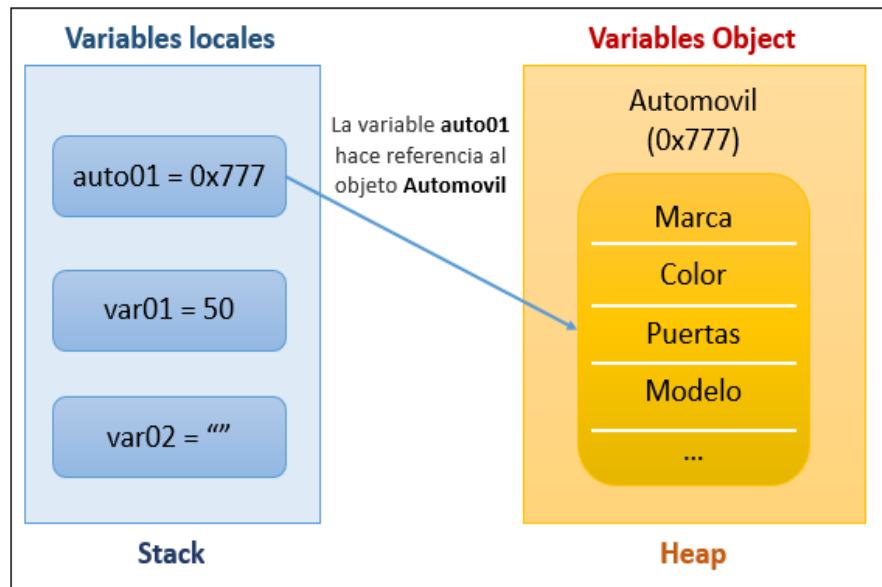
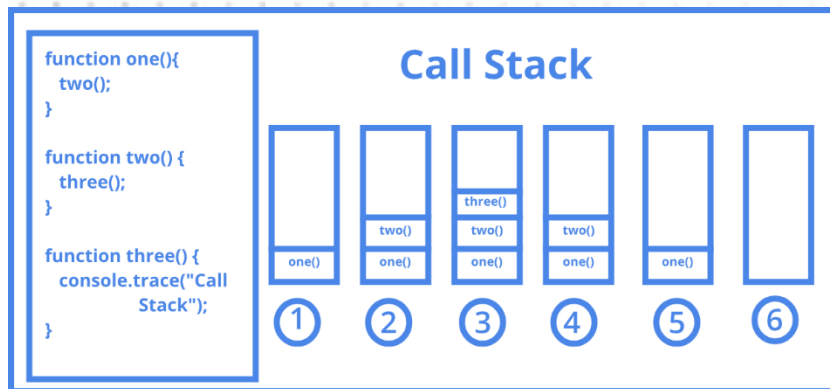
¿Cómo maneja Java la memoria?



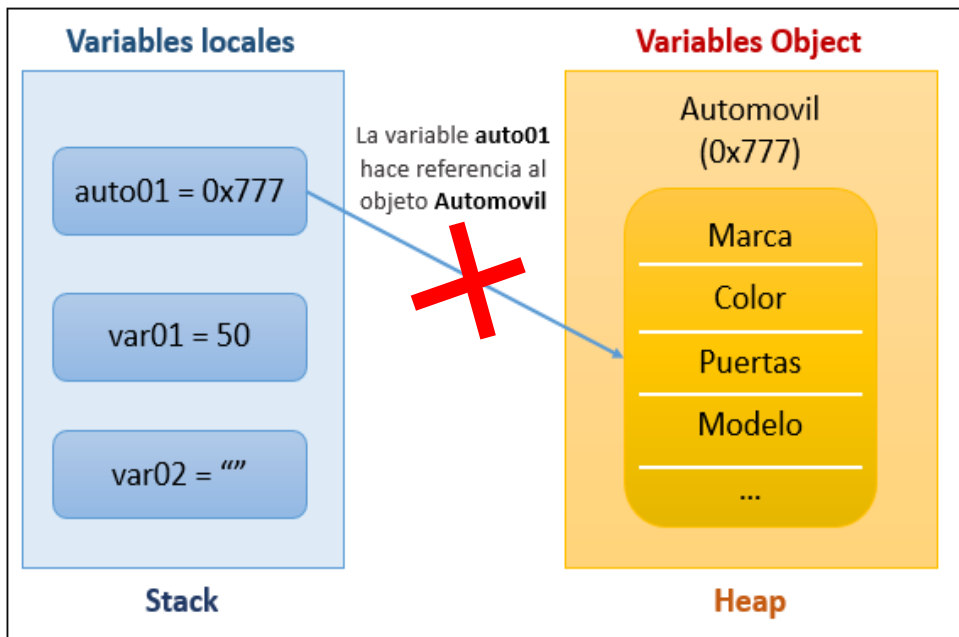
Variables

```
Automovil auto01 = new Automovil();  
int var01 = 50;  
String var02 = "";
```

Funciones



```
Automovil auto01 = new Automovil();  
int var01 = 50;  
String var02 = "";
```



auto01 = null;

***Objeto no
referenciado***

¿Cómo visualizamos la memoria consumida?



VisualVM

VisualVM 2.1.9

File Applications View Tools Window Help

Applications x

- Local
 - VisualVM
 - Saludo (pid 8780)
 - [heapdump] 06:32:19 AM
- Remote
- VM CoreDumps
- JFR Snapshots
- Snapshots

[heapdump] 31/12/1969 07:00:00 PM x

[heapdump] 31/12/1969 07:00:00 PM

Heap Dump

Summary ▾

Heap	Environment
Size: 1,282,144 B	System
Classes: 854	Architecture:
Instances: 28,997	Java Home:
Classloaders: 3	Java Version:
GC Roots: 811	Java Name:
Objects Pending for Finalization: 0	Java Vendor:
	JVM Uptime:

<https://visualvm.github.io/>

¿Cómo libera Java la memoria consumida por los objetos?



Libera memoria ocupada por
objetos que ya no son accesibles o
necesarios, evitando así fugas de
memoria y mejorando la eficiencia



```
auto01 = null;
```

***Objeto no
referenciado***

GC

Garbage Collector

*Sistema de gestión
automática de memoria*

OutOfMemoryError

Indicaciones

Simular el uso de memoria de una aplicación Java



Practiquemos

Tiempo: 15 minutos

Terminado el tiempo, cada grupo regresa a la sala principal y expone

Archivo: Dato.java

```
public class Dato {  
    private int A;  
    private int B;  
  
    public int getA() {  
        return A;  
    }  
  
    public void setA(int A) {  
        this.A = A;  
    }  
  
    public int getB() {  
        return B;  
    }  
  
    public void setB(int B) {  
        this.B = B;  
    }  
}
```

Archivo: Test.java

```
public class Test {  
    public static void main(String[] args) {  
        Dato d1 = new Dato();  
        d1.setA(10);  
        d1.setB(20);  
  
        Dato d2 = new Dato();  
        d2.setA(30);  
        d2.setB(40);  
  
        Dato d3 = new Dato();  
        d3.setA(50);  
        d3.setB(60);  
  
        d1 = d2;  
        System.out.println(Integer.toString(d1.getA() + d1.getB()));  
  
        d3 = d1;  
        d2 = d3;  
        System.out.println(Integer.toString(d1.getA() + d2.getA()));  
  
        d1 = d3;  
        d3.setB(60);  
        System.out.println(Integer.toString(d2.getA() + d1.getB()));  
  
        d1.setA(200);  
        System.out.println(Integer.toString(d3.getA() + d3.getB()));  
    }  
}
```

1

2

3

4



Universidad
Tecnológica
del Perú

¿Qué valor
imprime en
la línea
1, 2, 3 y 4?



¿Qué cantidad de memoria consume del heap?

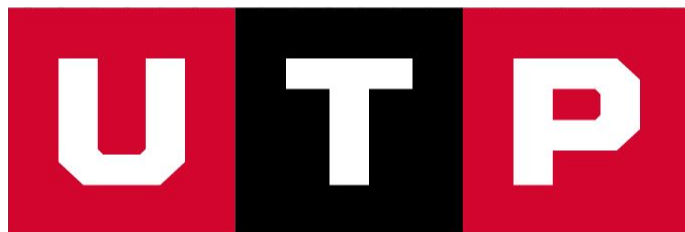
```
Saludo.java x
1 public class Saludo {
2     public static void main(String[] args) {
3         String mensaje;
4
5         for(int i=0; i<1000000; i++) {
6             mensaje = "HOLA A TODOS";
7             System.out.println(mensaje);
8         }
9     }
10 }
```



¿Qué aprendiste el día de hoy?

- ¿Qué es un archivo JAR?
- ¿Cuáles son las principales áreas de la memoria que maneja Java?
- ¿Que almacena Java en la memoria STACK?
- ¿Que almacena Java en la memoria HEAP?
- ¿Qué rol cumple el GARBAGE COLECTOR?

Levanta la mano para participar por audio o chat



**Universidad
Tecnológica
del Perú**