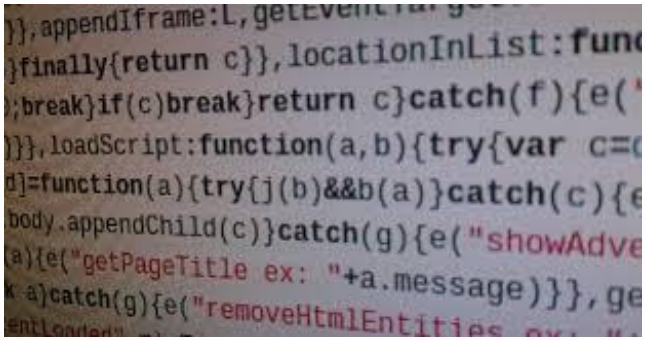


UT 7

INTRODUCCIÓN POO.

Utilización de Objetos

TERCERA PARTE

IES JUAN DE LA CIERVA
DPTO. INFORMÁTICA

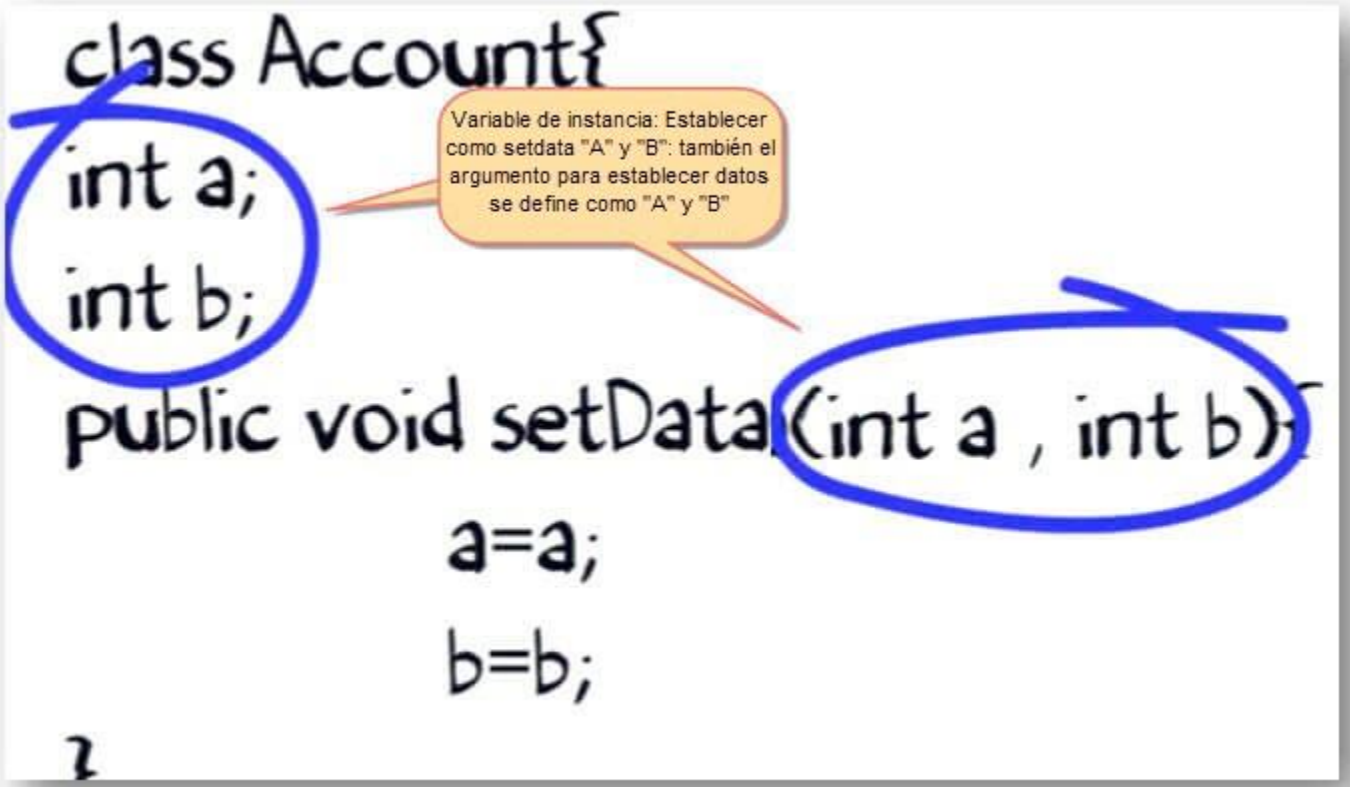
CONTENIDOS DE LA PRESENTACIÓN

Objetivos

Esta unidad propone la utilización de objetos ya definidos en el propio lenguaje, la creación por el usuario de objetos sencillos, así como las propiedades de estos.

Contenidos

1. Utilización de métodos.
2. Utilización de propiedades.
3. Parámetros y valores devueltos.
4. Ámbito de las variables.
5. Librerias de Objetos, clases envoltorio.
6. Sobrecarga de métodos
7. Constructores.
8. **Paso de parámetros por valor y por referencia.**
9. **Destrucción de objetos y liberación de memoria.**



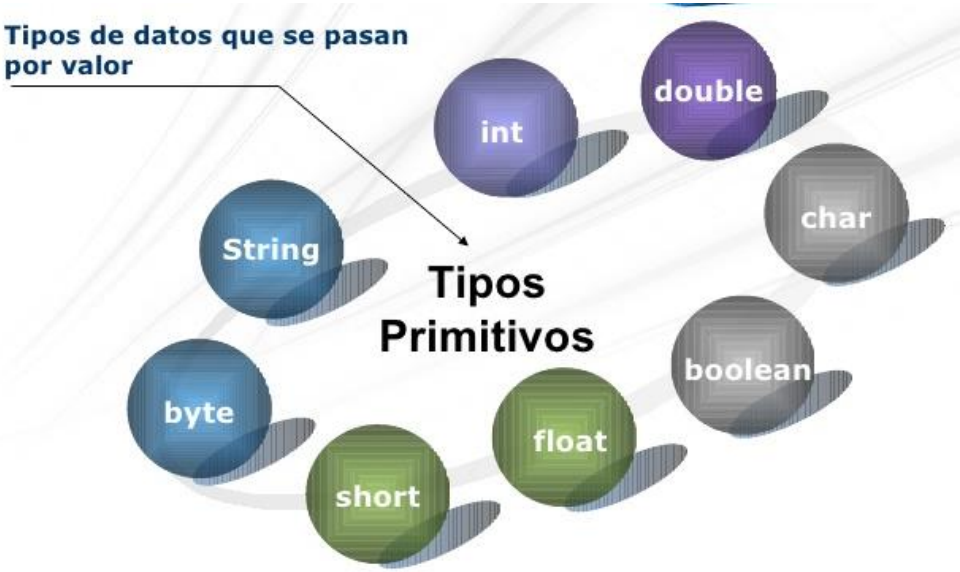
PASO DE ARGUMENTOS

Métodos: paso de parámetros

Métodos: paso de parámetros

Por valor.

- ✓ Recibimos una serie de argumentos y Java realiza una copia de esas variables que destruye al terminar el método.
- ✓ Son los tipos de datos primitivos.
- ✓ El valor original del argumento no varia.



Métodos: paso de parámetros

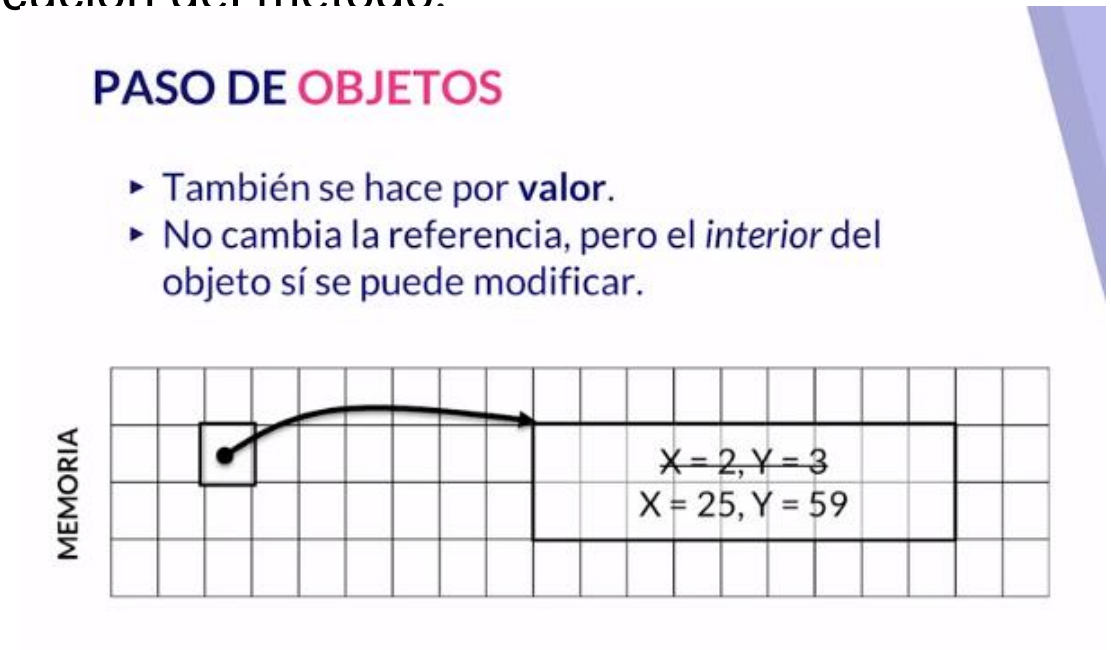
```java
public class PasoPorValor {

    public static void main(String[] args) {

        int x = 3;

        //invocamos el argumento y le pasamos x
        pasoPorValor(x);

        //imprimimos x y vemos si el parámetro ha cambiado
        System.out.println("Después de invocar pasoPorValor, x = " + x);

    }

    // cambiamos el valor en el método
    public static void pasoPorValor(int p) {
        p = 10;
    }
}
```

Métodos: paso de parámetros

Paso de Objetos

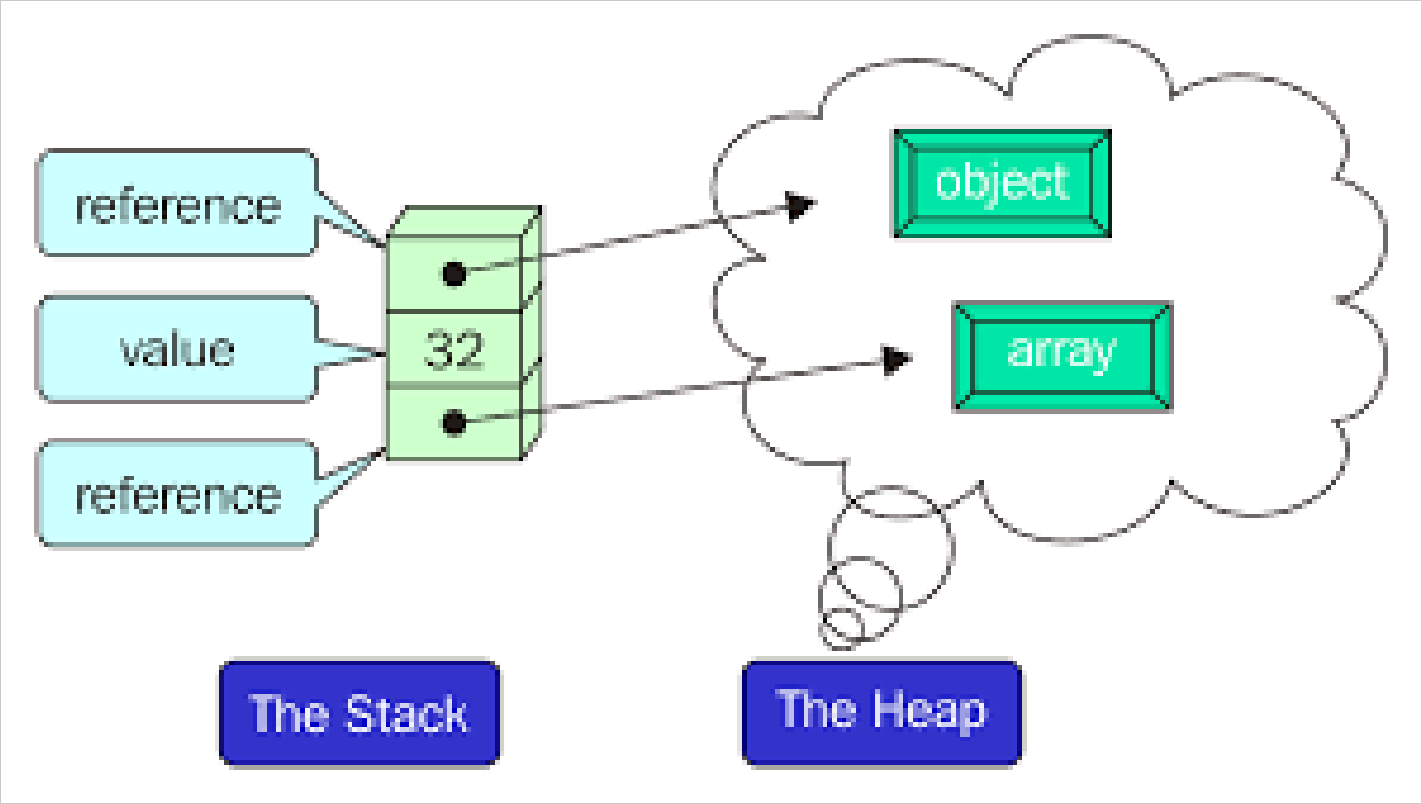
- ✓ El método no realiza una copia de los argumentos, sino que trabaja con los valores reales.
- ✓ Se realiza con objetos y arrays.
- ✓ El valor original del argumento varia, las modificaciones duran más allá del final de la ejecución del método.

En Java se pasa la referencia al objeto por valor.



Métodos: paso de parámetros

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

    String [] nombre1 = {"Maria","Juan","Lucas"};

    //Visualizar vector
    System.out.println("VECTOR ANTES DE LA MODIFICACIÓN");
    System.out.println(Arrays.deepToString(nombre1));

    pasoPorReferencia(nombre1);
    System.out.println("VECTOR DESPUES DE LA MODIFICACIÓN");
    System.out.println(Arrays.deepToString(nombre1));


}

// cambiamos el valor en el método
public static void pasoPorReferencia(String v[]) {
    //modificación del vector v
    v[0]="Pedro";
    v[1]="Roberto";
}
```



MEMORIA HEAP Y MEMORIA STACK

Uso de la Memoria en Java

Memoria de pila (Stack Memory) –Se utiliza para almacenar:

- Variables locales
- Primitivas
- Referencias a ubicaciones en la memoria de montón

Memoria de montón (Heap Memory) – se utiliza para almacenar ……

. objetos

Uso de la Memoria en Java

Descripción de un preso

- Propiedades:
  - Nombre
  - Altura
  - Años de sentencia
- Comportamientos:
  - Piense en lo que han **hecho**

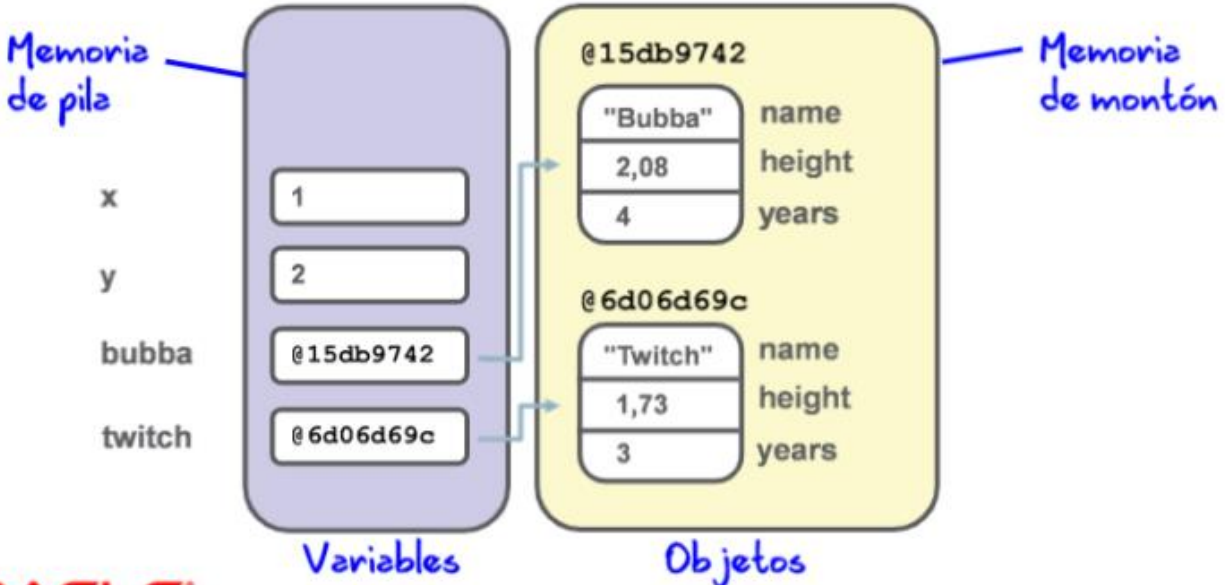
ORACLE

Uso de la Memoria en Java

Uso de la Memoria en Java

Referencias y objetos en memoria

```
int x = 1;
int y = 2;
Prisoner bubba = new Prisoner();
Prisoner twitch = new Prisoner();
…
```

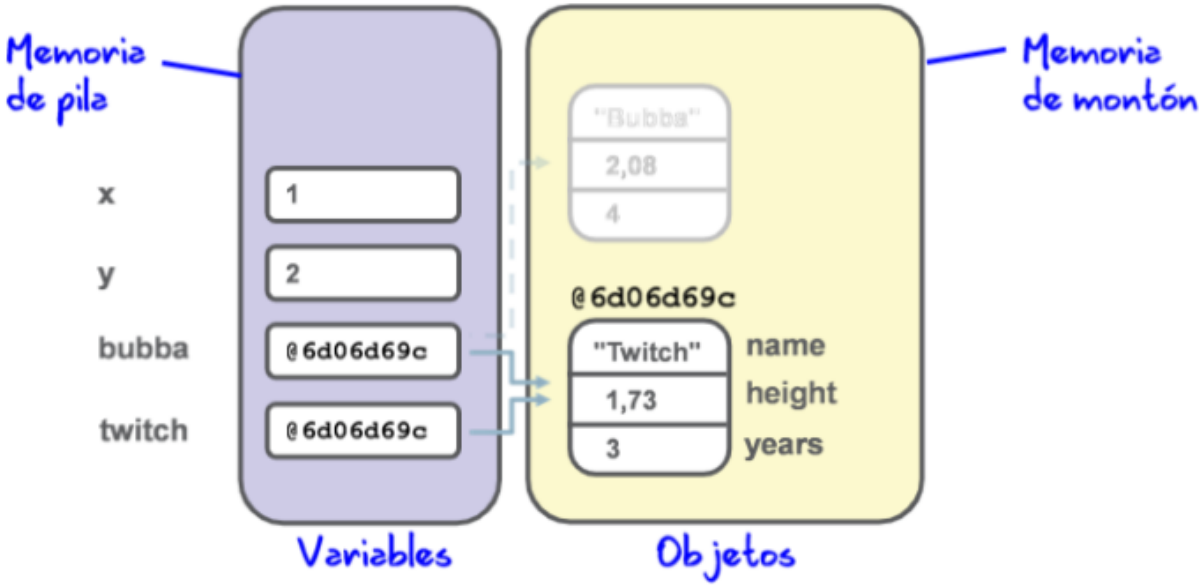


ORACLE

Uso de la Memoria en Java

Asignación de una referencia a otra

```
bubba = twitch;
```



ORACLE

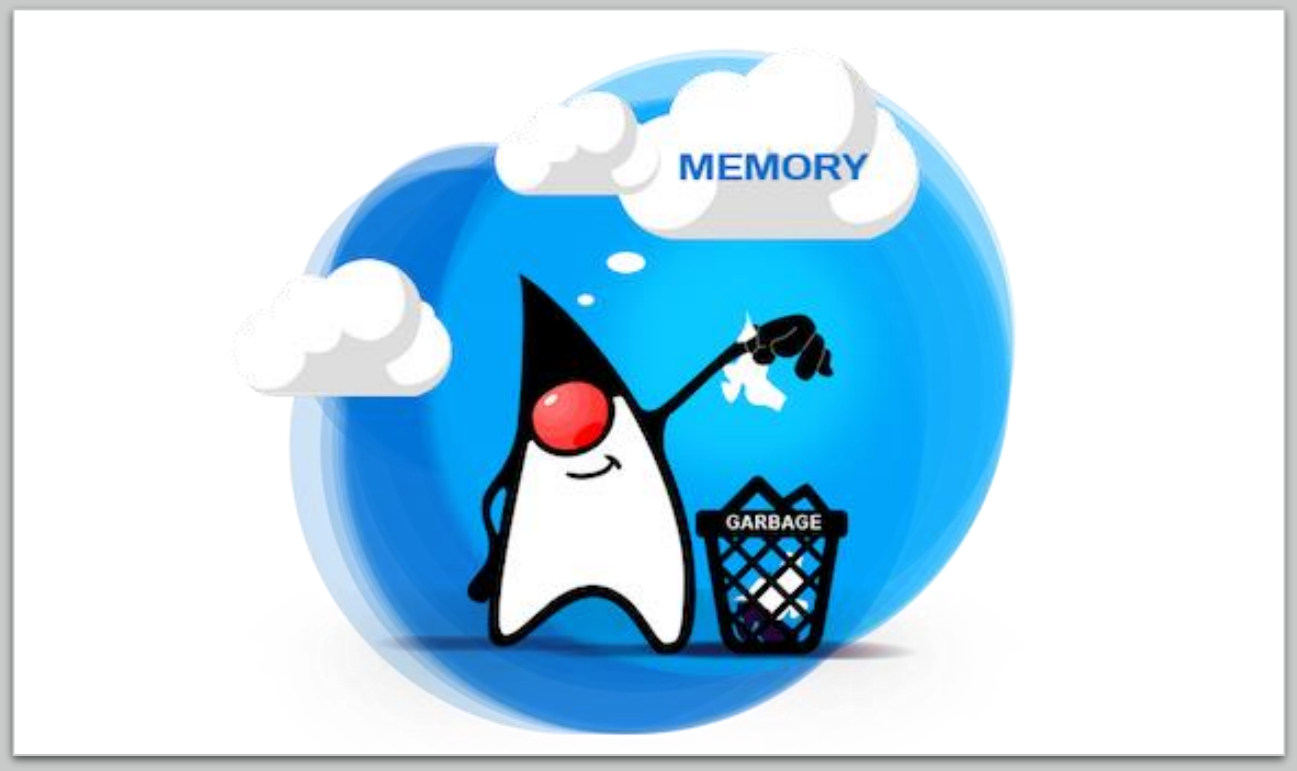
Uso de la Memoria en Java

Dos referencias, un objeto

View Notes

- A partir de la línea 14, `bubba` y `twitch` hacen referencia al mismo objeto.
- Cualquier variable de referencia se podría utilizar para acceder a los mismos datos.

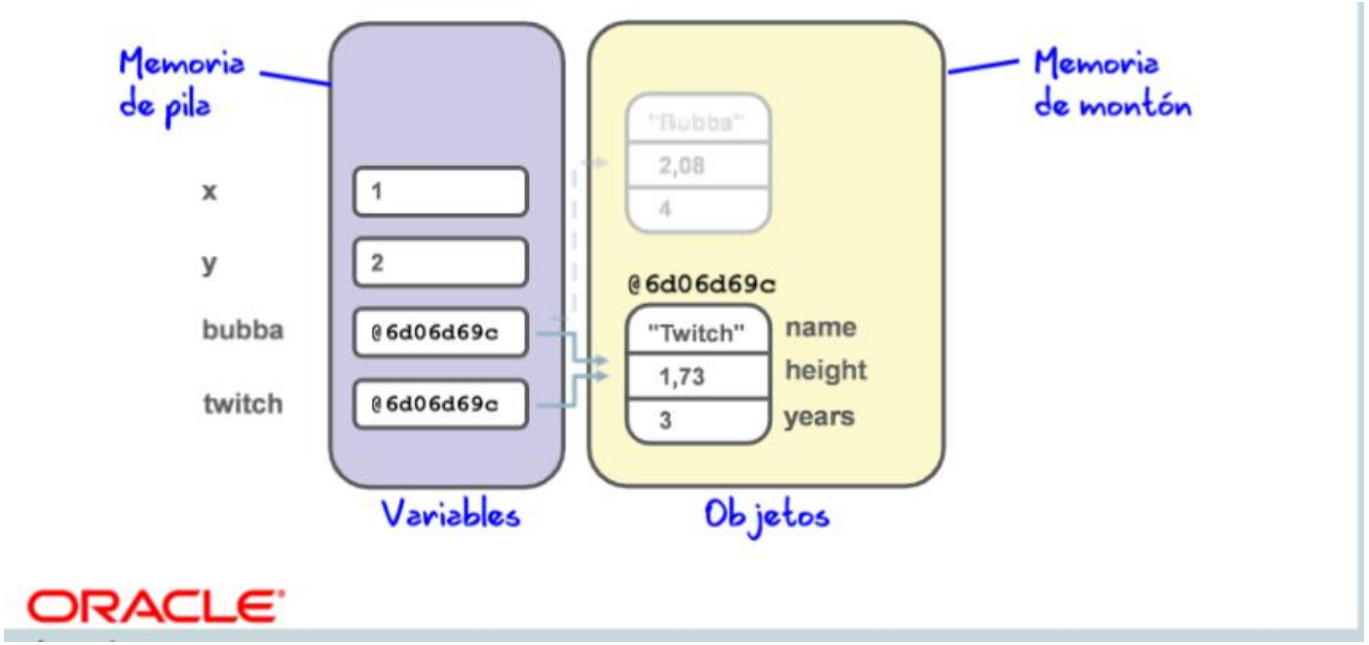
```
11 Prisoner bubba = new Prisoner();
12 Prisoner twitch = new Prisoner();
13
14 bubba = twitch;
15
16 bubba.name = "Bubba";
17 twitch.name = "Twitch";
19
20 System.out.println(bubba.name);       //Twitch
21 System.out.println(bubba == twitch); //true
```

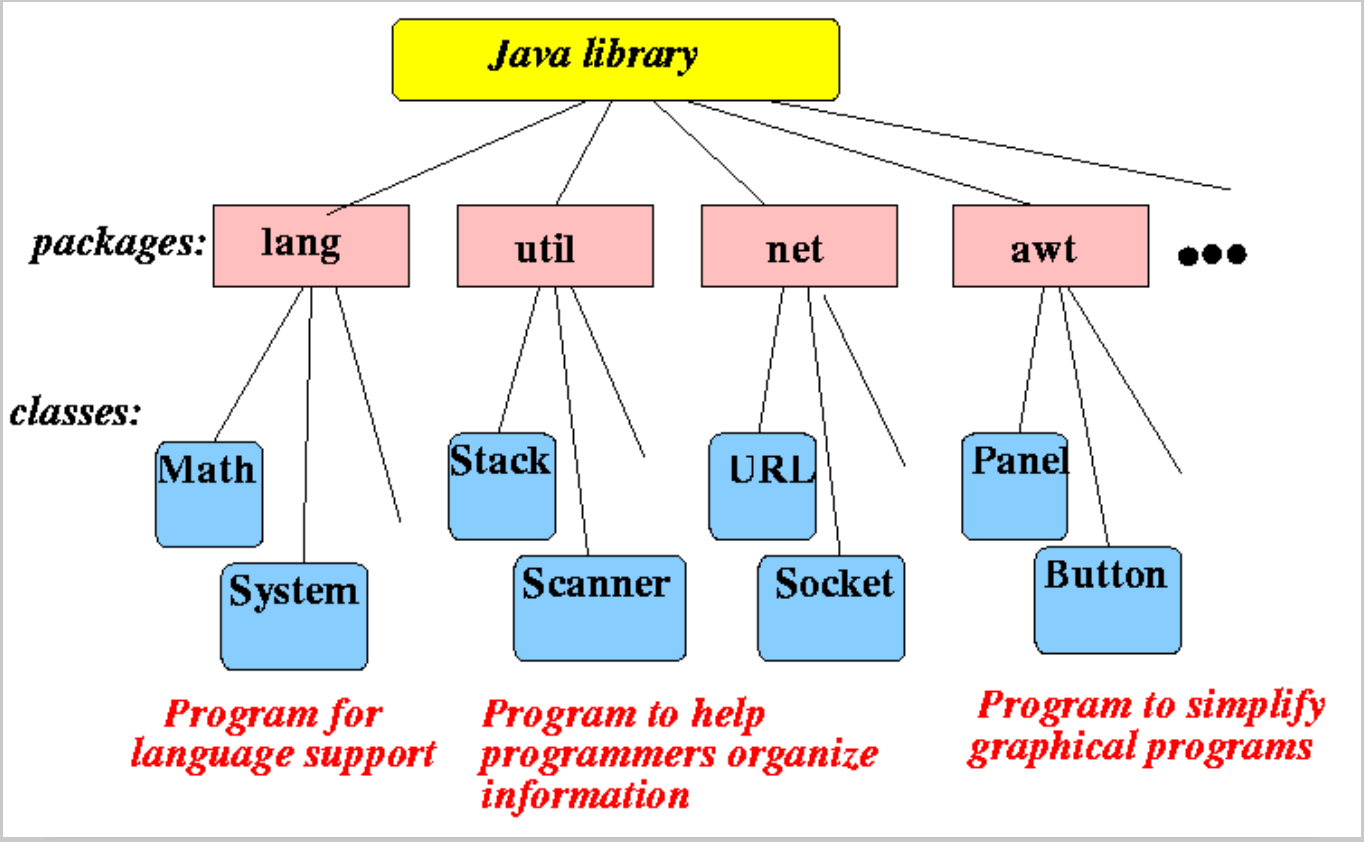


RECOLECTOR DE BASURA DE JAVA

Uso de la Memoria en Java
Garvage Collector

Si no hay variables de referencia que apunten al objeto, Java borra automáticamente la memoria que había ocupado el objeto.
Este proceso se llama **recolección de basura.**




Java library
packages:
lang
util
net
awt
classes:
Math
System
Stack
Scanner
URL
Socket
Panel
Button
Program for language support
Program to help programmers organize information
Program to simplify graphical programs

PACKAGE

PACKAGE

PAQUETE o PACKAGE, es un conjunto de clases que facilitan la gestión del acceso a las clases.

Permite proteger clases restringiendo el acceso y permite definir un espacio de nombres para organizar y localizar las clases.

Java, importa por defecto tres paquetes:

1. Java.lang
2. La clase actual.
3. El paquete por defecto.

Podemos decir que un paquete es una biblioteca de clases. Podemos incorporar todo el paquete o solo una clase.

```
import  java.util.*;
import  java.util.Date;
```

En la primera línea de código escribiremos **package nombre**

PACKAGE

- **package**, es la primera línea escrita en el fichero.
- Es una agrupación de clases y todas se encuentran en el mismo directorio (este puede tener subdirectorios).
- El directorio contendrá ficheros **class.**