



# ICC311

# Estructuras de Datos

Semestre I, 2020

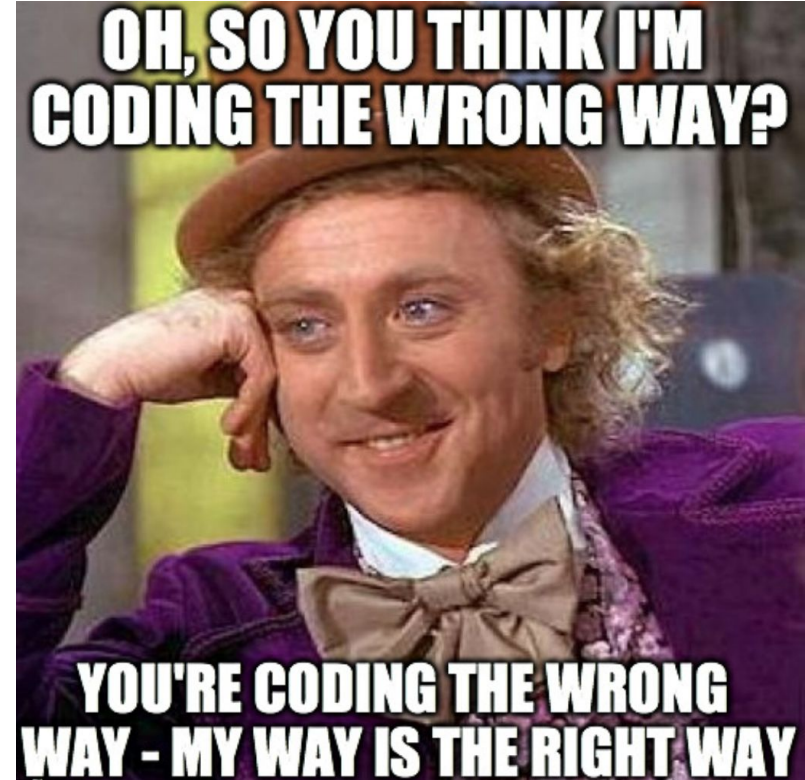
Profesor: Pablo Valenzuela

# Semana 01 - Parte 01

---

## Clase 2: Modelos de memoria

- Datos primitivos
- Objetos



# Preguntas

```
Auto a = new Auto("Mazda", 2005);  
Auto b;  
b = a;  
b.marca = "BMW";  
System.out.println(a);  
System.out.println(b);
```

**Afecta el cambio de "b" a "a"?  
¿Sí - No?**

**marca: BMW, año: 2005  
marca: BMW, año: 2005**

```
int x = 5;  
int y;  
y = x;  
x = 2;  
System.out.println("x es: " + x);  
System.out.println("y es: " + y);
```

**Afecta el cambio de "x" a "y"?  
¿Sí - No?**

**x es: 2  
y es: 5**

## **El computador almacena información en memoria**

- La información se almacena como una secuencia de unos y ceros.
- Por ejemplo:
  - 72 se almacena como 01001000
  - Letra H se almacena como 01001000
  - True se almacena como 00000001

## **Cada tipo en java se interpreta de manera diferente**

- 8 tipos de datos primitivos:
  - byte, short, int, long, float, double, boolean, char

# Declaración de una variable (simplificado)

## Cuando se declara una variable de cierto tipo en java:

- El computador separa exactamente los bits necesarios para almacenar lo que se necesite. Por ejemplo:
  - Declarar un int separa un espacio de 32 bits
  - Declarar un Double separa un espacio de 64 bits
- Java crea una tabla interna que mapea cada nombre de variable a una posición
- Java no escribe "algo" sobre los espacios reservados
  - Por seguridad, no se permite el acceso a variables no inicializadas

```
→ int x;  
double y;  
x = - 1431195969;  
y = 567213.112;
```

x

10100101010101010101011111000

y

010101010100111111000000110001010101010101000011

# Modelos de memoria

---

## Tipos de datos primitivos

- Escribiremos modelos usando notación legible por humanos
- Ejemplo con tipos de datos primitivos:

```
int x;  
double y;  
x = - 1431195969;  
y = 567213.112;
```

**código**

**x**

1431195969

**y**

567213.112

**modelo**

# Modelos de memoria

---

## Regla del símbolo "=" para datos primitivos

- Dados dos variables  $x$  e  $y$ :
  - $y = x \rightarrow$  copia todos los bits desde  $x$  a  $y$

**x**

1431195969

**y**

1431195969

**modelo de  
memoria**

# Modelos de memoria

- Herramienta: Java Visualizer



The screenshot displays the Java Visualizer tool interface. On the left, a code editor shows a Java class with a main method. The code is as follows:

```
1 public class ClassNameHere {  
2     public static void main(String[] args) {  
3         int x = 1431195969;  
4         int y;  
5         y = x;  
6  
7     }  
8 }
```

Below the code editor is a progress bar and navigation buttons: "<< First", "< Back", "Program terminated", "Forward >", and "Last >>". A link "Edit code" is also present.

On the right, the "Frames" panel shows the current frame "main:7" with variables:

Variable	Value
x	1431195969
y	1431195969
Return value	void

Below the frames panel is the "Objects" panel, which is currently empty.

The top right corner of the interface features the Java Visualizer logo (a blue cup with glasses) and the text "Java Visualizer (beta: [report a bug](#))".

**x**

1431195969

**y**

1431195969

**modelo de  
memoria**



# Modelos de memoria

## Tipos de referencias (objetos)

- Todo lo que no es de tipo primitivo, incluyendo los arreglos, es un tipo de referencia (objeto)
- Cuando se instancia una clase (por ejemplo, la clase Auto):
  - Primero, Java separa un espacio de memoria (cierta cantidad de bits) para cada instancia y lo completa con un valor por defecto (por ejemplo, 0 o null)
  - Segundo, el constructor completa el espacio con el valor que corresponda

```
static public class Auto {  
  
    public int anio;  
    public double kilometraje;  
  
    public Auto(int m, double km){  
        anio = m;  
        kilometraje = km;  
    }  
}
```

**código**

```
Auto auto = new Auto(2010, 2000);
```

Auto instance

**32 bits**

anio 2010

**64 bits**

kilometraje 2000.0

**modelo**

# Modelos de memoria

- Todo lo que no es de tipo primitivo, incluyendo los arreglos, es un tipo de referencia (objeto)
- Cuando se instancia una clase (por ejemplo, la clase Auto):
  - Primero, Java separa un espacio de memoria (cierta cantidad de bits) para cada instancia y lo completa con un valor por defecto (por ejemplo, 0 o null)
  - Segundo, el constructor completa el espacio con el valor que corresponda

```
Auto auto = new Auto(2010, 2000);
```

Verde es **anio**, Azul es **kilometraje**

# Modelos de memoria

- Se puede pensar que "new" retorna la dirección de un objeto recién creado
- Ejemplo (idea general): si el objeto es creado en la dirección 2384723423
  - "new" retorna 2384723423

2384723423 (avo) bit

[illegible]

2384723423

```
Auto auto = new Auto(2010, 2000);
```

## 32 bits

## Auto instance

## 64 bits

anio	2010
kilometraje	2000.0

**96 bits**

Verde es **anio**, Azul es **kilometraje**

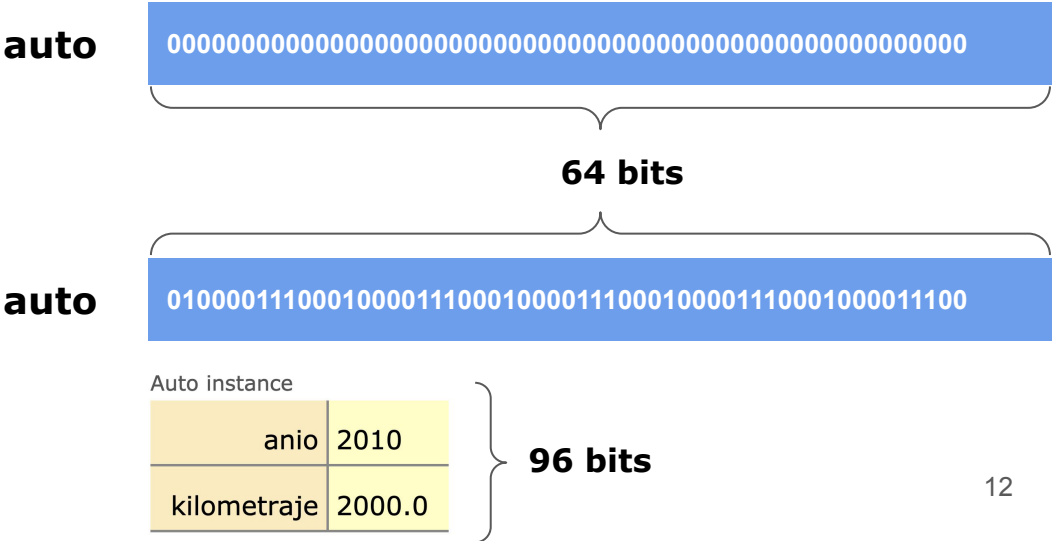
# Modelos de memoria

## Declaración de clases

- Java separa exactamente 64 bits, no importante el objeto
- Estos bits pueden ser definidos como:
  - Null (todos ceros)
  - La dirección de 64 bits de una instancia específica de la clase (lo que retorna "new")

```
Auto auto;  
auto = null;
```

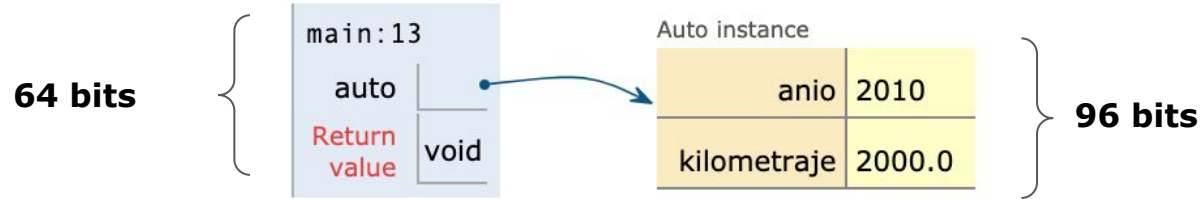
```
Auto auto;  
auto = new Auto(2010, 2000);
```



# Modelos de memoria

## Declaración de clases

- Java separa exactamente 64 bits, no importando el objeto
- Estos bits pueden ser definidos como:
  - Null (todos ceros)
  - La dirección de 64 bits de una instancia específica de la clase (lo que retorna "new")

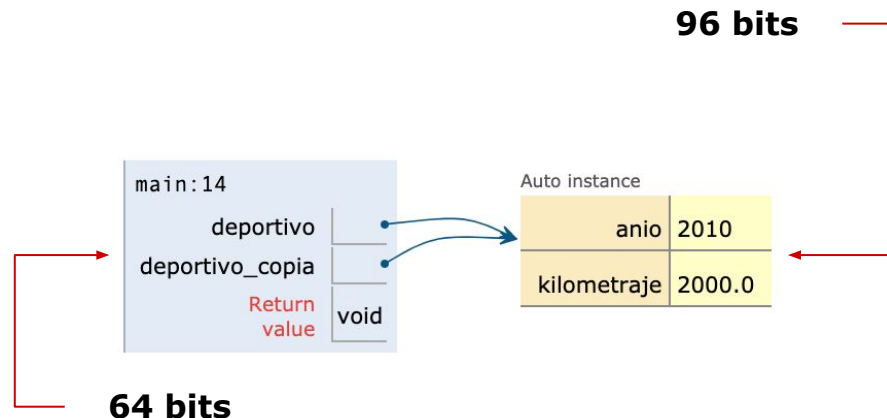


# Modelos de memoria

## Regla del símbolo "=" para objetos

- Tal como en el caso de datos primitivos, se copian los bits
- Representamos esto visualmente con un modelo de memoria

```
Auto deportivo;  
deportivo = new Auto(2010, 2000);  
Auto deportivo_copia;  
deportivo_copia = deportivo;
```





# ICC311

# Estructuras de Datos

Semestre I, 2020

Profesor: Pablo Valenzuela