UTN F r a
Avellaneda

Programación II

Clase 01

Introducción a Java

Lic. Matias Ramos

Temario

Presentación y encuadre

- Bienvenida y presentación de docentes y alumnos.
- Introducción a la material
- Forma de trabajo y evaluación
- Herramientas que necesitarán

Introducción al lenguaje Java

- Historia de Java
- Características principales
- Estructura básica de un programa Java

De Python estructurado a Java orientado a objetos

- ¿Qué es un paradigma de programación?
- Principales paradigmas
- Python estructurado
- Programación Orientada a Objetos (Java)

Cierre

- ¿Por qué aprender Java hoy en día?

¿Qué es un paradigma de programación?

- Es un modelo o estilo de pensar y resolver problemas con código.
- Define cómo organizamos las instrucciones y los datos en un programa.
- No existe un “mejor” paradigma, sino el más adecuado según el problema.

Principales paradigmas

- Imperativo / Estructurado:
- Se centra en *cómo* resolver el problema (secuencia de pasos)
- Usa variables, funciones, bucles, condicionales.

- Orientado a Objetos (OOP):
- Se centra en los objetos que representan al problema.
- Cada objeto combina datos y comportamientos.

Python estructurado

- Lenguaje que usaron en Programación I

- Paradigma estructurado:
  - Datos y funciones **separados**
  - Ejecución secuencial.

- Ejemplo: función que recibe datos y los procesa

```
saldo = 1000

def depositar(saldo, monto):
    return saldo + monto

saldo = depositar(saldo, 200)
```

Programación Orientada a Objetos (Java)

- Nuevo paradigma que verán en este curso.

- Objetos = combinación de datos + métodos
- Ventajas: modularidad, escalabilidad, reutilización.
- Ejemplo: cuenta.depositar(monto) en lugar de pasar datos sueltos

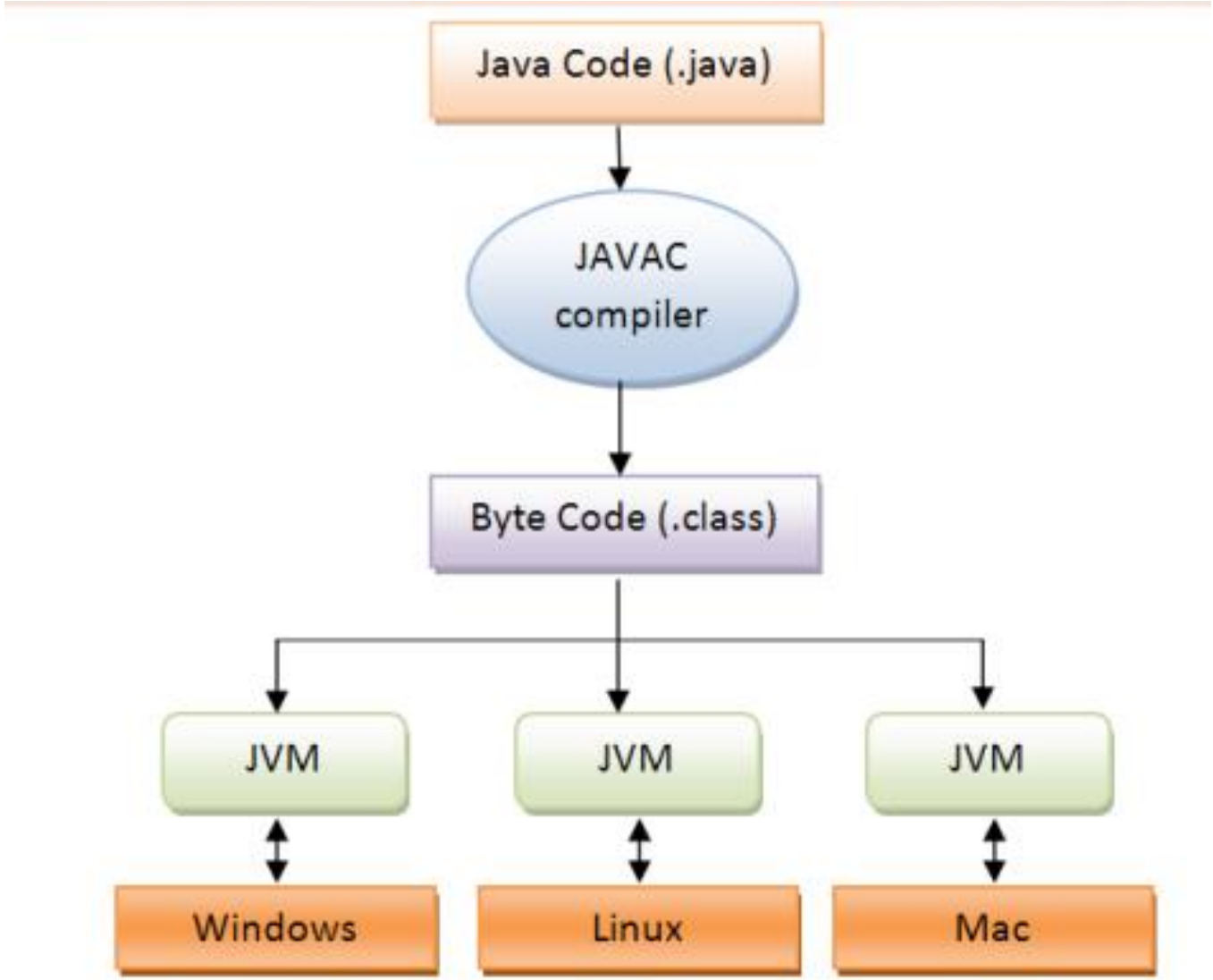
```
class Cuenta {
    int saldo = 1000;

    void depositar(int monto) {
        saldo += monto;
    }
}
```

Introducción a Java

- Creado en 1995, pensado para ser multiplataforma.
- Es un lenguaje de programación de alto nivel y orientado a objetos
- Usos principales: IoT, aplicaciones web, móviles (Android), backend empresarial
- Tipado fuerte y estático.
- Multiplataforma (eslogan: Write once, run anywhere).
- Amplia comunidad y bibliotecas.

Multiplataforma

- Lenguaje compilado e interpretado



Tipos de datos y Flexibilidad

Python y Java tienen una manera distinta de manejar **tipos de datos**, Java al ser un lenguaje **fuertemente tipado** nos obliga a declarar el tipo de nuestra variable antes de asignarla, de esta forma dejando en claro que esa variable solo va a poder tomar valores correspondientes a ese tipo de dato.

```python
moverse = True
moverse = False
moverse = 1337

print(moverse)
```


```java
boolean moverse = true;
moverse = false;
System.out.println(moverse);
```

En Python las variables son **inmutables**. Esto significa una variable al cambiar de valor (al asignar un nuevo valor), la misma cambia de dirección de memoria (se crea una nueva variable). Excepto las listas.

Como las variables en Java son **mutables,** al asignar un valor, las mismas mantienen sus direcciones de memoria. Excepto los strings (cadenas).

Finalización de comandos

Otra gran diferencia entre los dos lenguajes es la **finalización de comandos**, reconocemos como un **comando** al inicio de una instrucción.

```python
moverse = True
moverse = False
moverse = 1337

print(moverse)
```


```java
boolean moverse = true;
moverse = false;
System.out.println(moverse);
```

Python para finalizar las instrucciones utiliza los **saltos de línea**.

El “**;**” es nuestro finalizador de instrucción en Java.

Bloques de codigo

Un **bloque de código** en los lenguajes de programación es una sección con una o más declaraciones o sentencias.

```python
moverse = True
print("Revisando.......")

if moverse:
    moverse = False
    print("Cambiando a falso")
```

En Python la forma en la que se encierra el código en un **bloque** es con la **indentación**, todo el código que está indentado y abajo de la indentación del if se va a ejecutar en el **bloque.**


```java
boolean moverse = true;
System.out.println("Revisando.......");

if(moverse){
    moverse = false;
    System.out.println("Cambiando a falso");
}
```

Para separar **bloques** de código usamos las llaves “**{}**” estas nos permiten encerrar nuestro código en un **scope** y ejecutarlo dependendiendo si la condición if se cumple.

Tipos de datos

Tipos de datos primitivos (no son clases)	
Tipo	**Rango**
byte	-128 .. 127
short	-32768 .. 32767
int	-2147483648 .. 2147483647
long	-9.223.372.036.854.775.808 .. 9.223.372.036.854.775.807
float	-3.4x10 38 .. 3.4x10 38 (mínimo 1.4x10 –45)
double	-1.8x10308 .. 1.8x10308 (mínimo 4.9x10-324)
boolean	true o false
char	unicode

Tipos de datos compuestos (son clases)
String
Vectores y matrices
Colecciones
Clases
wrappers/envolventes

Operadores

Operadores ariméticos	
Operador	**Nombre**
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo / Resto
++	Incremento
--	Decremento

Operadores de asignación	
Operador	**Nombre**
=	Asignación
+=	Suma y asignación
-=	Resta el valor de la izquierda al de la variable de la derecha y almacena el resultado en la misma variable.

```
int numero1=5;
int numero2=6;
int resultado=0;

resultado=numero1+numero2;
resultado=numero1-numero2;
resultado=numero1*numero2;
resultado=numero1/numero2;
```

Operadores

Operadores de comparación	
Opr.	**Nombre**
<	Menor que
>	Mayor que
<=	Menor o igual a
>=	Mayor o igual a

Operadores lógicos	
Opr.	**Nombre**
!	Negación lógica
&	AND lógico
&&	AND condicional lógico / cortocircuito
|	OR lógico
||	OR condicional lógico / cortocircuito

Operadores de igualdad	
Opr.	**Nombre**
==	Igualdad
!=	Desigualdad

```
boolean esMayor =edad>=18;
boolean esMenor =edad<18;
boolean esArgentino = pais == "Argentina";
boolean noEsArgentino = !esArgentino;
boolean habilitado = esArgentino && esMayor;
boolean noHabilitado = noEsArgentino || esMenor;
```

Sentencias condicionales simples


```python
if x < 10:
    metodo_uno()
```

```java
3   if (x < 10)
4   {
5       MetodoUno();
6   }
```

Sentencias condicionales dobles

```python
if x < 10:
    metodo_uno()
else:
    metodo_dos()
```


```java
if (x < 10)
{
    MetodoUno();
}
else
{
    MetodoDos();
}
```

Sentencias condicionales múltiples


```python
if x < 10:
    metodo_uno()
elif x >= 20:
    metodo_dos()
else:
    metodo_tres()
```

```
4
5   if (x < 10)
6   {
7       MetodoUno();
8   }
9   else if (x >= 20)
10  {
11      MetodoDos();
12  }
13  else
14  {
15      MetodoTres();
16  }
```

Lic. Matias Ramos

Sentencias de selección múltiple

```python
a = 9

match a:
    case 0:
        metodo_uno()
    case 1:
        metodo_dos()
    case _:
        metodo_tres()
```


```java
int a = 9;

switch  (a)
{
    case 0:
        MetodoUno();
        break;

    case 1:
        MetodoDos();
        break;

    default:
        MetodoTres();
        break;
}
```

Estruturas repetitivas

```java
for (int dia : dias) {
    System.out.println(dia);
}

for (int i = 0; i < 5; i++) {
   System.out.println(i);
}
```

```java
int i = 0;

while (i < 5) {
     System.out.println(i);
     i++;
}

do {
System.out.println(i);
i++;
} while (i < 5);
```


LIVE DEMO