MitoCode

SESIÓN 1:
INTRODUCCIÓN JAVA 17

Instructora: Almendra Paz Rodríguez

Correo:
javaocp@mitocodenetwork.com

www.mitocode.com

BIENVENIDOS!

COMENZAREMOS EN 3 MINUTOS

AGENDA DE LA SESIÓN

- Tecnología Java
- Creando un programa simple
  - Entendiendo la estructura de una clase Java
  - Java Identifiers
  - Compilación y ejecución
- Creando un programa con paquetes
  - Declaración de package e imports
  - Orden de elementos

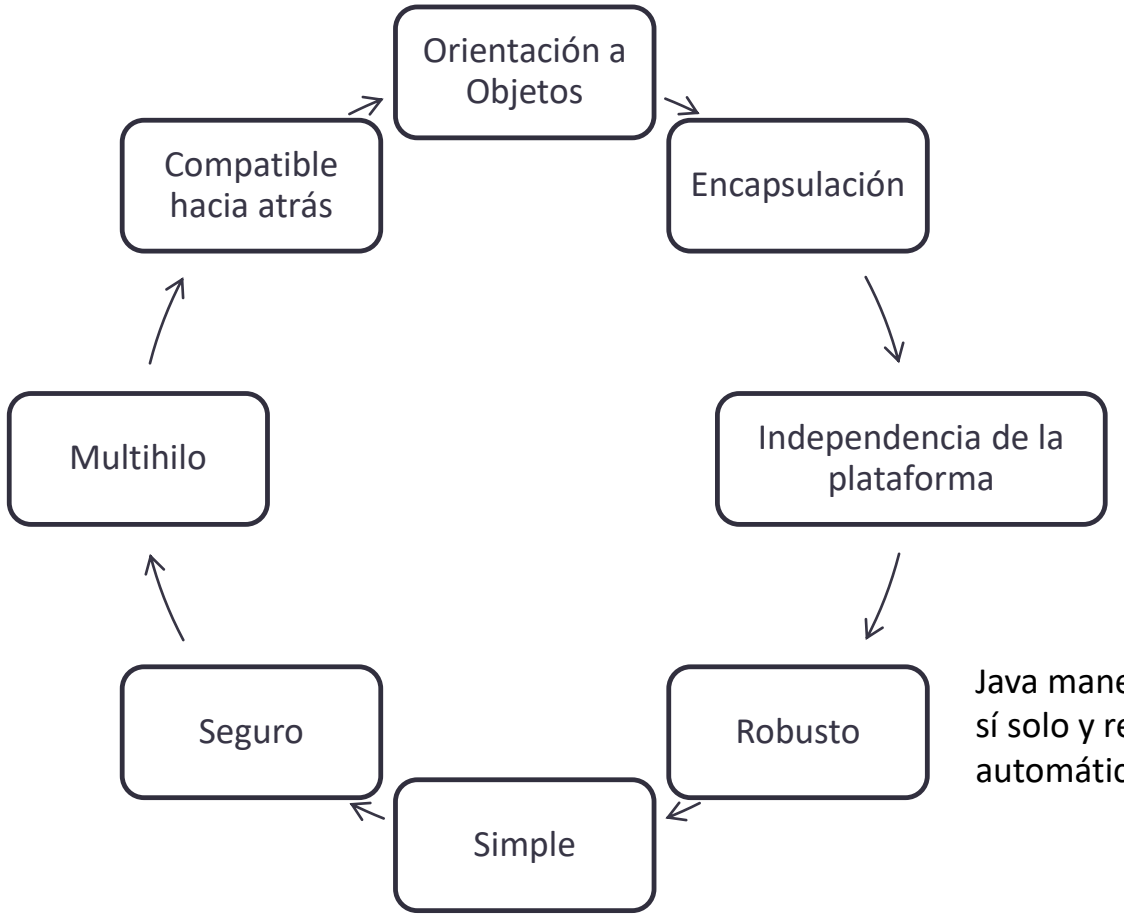
TECNOLOGÍA JAVA

- Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems.
- Es un lenguaje interpretado, es decir no es ejecutado directamente por el sistema operativo, sino por una máquina virtual conocida como JVM (Java Virtual Machine).
- ¿Por qué lo prefieren?
  - Java ha sido probado, ajustado, ampliado y probado por toda una comunidad de desarrolladores, arquitectos de aplicaciones y entusiastas de Java. Java está diseñado para permitir el desarrollo de aplicaciones portátiles de elevado rendimiento para el más amplio rango de plataformas informáticas posible. Con millones de desarrolladores que ejecutan más de 51.000 millones de instancias de Java Virtual Machine en todo el mundo, Java sigue siendo la plataforma de desarrollo preferida por empresas y desarrolladores.

TECNOLOGÍA JAVA

- Java Development kit (JDK): contiene el software mínimo necesario para desarrollar en java. Es oficialmente distribuido por *Adoptium community* y Oracle. Incluye:
  - Javac: convierte los archivos .java a .class
  - Java: inicia la máquina virtual de java y ejecuta el programa
  - Jar: empaqueta los archivos
  - Javadoc: genera la documentación
- Desde Java 11 ya no existe el módulo JRE.
- Java Api (Application Programming Interface): es la denominación que recibe las clases e interfaces que vienen en el JDK. Funcionalidades y algoritmo que requiere un desarrollador para lograr su objetivo.
- Las últimas versiones de java con soporte a largo plazo (lanzamientos cada 3 años) son la versión 11 y 17.
  - Java 7 cuenta con vigencia hasta 2022
  - Java 8 cuenta con vigencia hasta 2030
  - Java 11 cuenta con vigencia hasta 2026
  - Java 17 cuenta con vigencia hasta 2029

TECNOLOGÍA JAVA



"write once, run everywhere"

Java maneja la memoria por sí solo y realiza la recolección automática de basura.

CREANDO UN PROGRAMA SIMPLE

```
/*Clase Student, creado por juan.perez
*/
public class Student{
        private String name = “Juan Perez”;

        // Imprime en consola el nombre
        public void printName() {
                System.out.println(“Mi nombre es: ”+ this.name);
        }
}
```

- Clase: estructura básica
- Objeto: instancia en memoria de una clase
- Estructura de una clase
  - Fields: o también llamadas variables
  - Métodos: en otros lenguajes llamados functions o procedures
    - firma del método, declaración del método
  - Comentarios: línea única, multi línea. Pueden ir en cualquier lugar.

CREANDO UN PROGRAMA SIMPLE

```
public class MainMethod
{
   //String args[], String… args
   //public static void main(String args[])
   public static void main(String[] args)
   {
      System.out.println(“Hola mundo");
   }
}
```

- Método main(): es el enlace entre el inicio del proceso de java y la programación hecha por el desarrollador
  - tiene que ser público, estático y retornar void.
  - Necesita un parámetro array de String
  - main es diferente de Main
- Ejecución y compilación
  - javac MyClass.java
  - java MyClass

RELACIÓN ENTRE NOMBRE DEL ARCHIVO Y NOMBRE DE LA CLASE

- Si dentro del archivo hay un type público (class, enum, interface, record), el nombre del archivo debe llevar el nombre de la clase pública.
- Solo puede haber un type público en un archivo.
- Estas reglas solo aplican para clases top-level, no para “nested” clases (clases declaradas en el cuerpo de otra clase”).

- En caso dentro de un archivo no haya ningún type público, entonces podemos declarar muchos types dentro que no coincidan con el nombre del archivo.

```
class PublicClass1{
	public static void main(String args[]){
		System.out.println("Hello world 1");
	}
}

class PublicClass2{
	public static void main(String args[]){
		System.out.println("Hello world 2");
	}
}
```

JAVA IDENTIFIERS

CREANDO UN PROGRAMA SIMPLE

```
public class Identifiers$
{
    public static void main(String[] args)
    {
        String $ = "Almendra";
        String $_ = "Paz";
        String $_3 = "Rodriguez";
        System.out.println("Valor $: "+ $);
      System.out.println("Valor $_: "+ $_);
      System.out.println("Valor $_3: "+ $_3);
    }
}
```

- Nombre de variables, métodos y clases
- Reglas
  - Puede contener letras (mayúsculas y minúsculas), caracteres ($ _) y dígitos.
  - No se pueden usar palabras reservadas.
  - El primer caracter no puede ser un dígito.
  - No se permite solo un _

CREANDO UN PROGRAMA SIMPLE

```
public class MyClass
{
        public static void main(String[] args)
    {
        System.out.println("first param: "+ args[0]);
        System.out.println("first param: "+ args[1]);
        System.out.println("first param: "+ args[2]);
    }
}
```

- Paso de parámetros:
  - javac MyClass.java
  - java MyClass primer "segundo param" tercero
- Corriéndolo en una sola línea
  - En java 11, hay compilación en memoria y ejecución. No produce .class file.
  - Funciona para programas con un solo archivo.
  - java MyClass.java primer segundo tercero

CREANDO UN PROGRAMA CON PAQUETES

```
import java.time.LocalDate;


public class TodayClass
{
        public static void main(String[] args)
    {
        LocalDate myDate = LocalDate.now();
        System.out.println("today: "+ myDate);
}
}
```

- Declaración de packages e imports
  - Para utilizar clases declaradas en otros directorios.
  - Si empieza con java o javax es porque viene con el jdk. Por defecto se carga java.lang.*
  - Misma regla que los identifiers
    - <identifier>. <identifier>. <identifier>
- Wildcards:
  - Import java.time.*;
  - Incorrecto sería: java.* y java.*.*;
  - Solo uno por vez y siempre al final.

CREANDO UN PROGRAMA CON PAQUETES

```
import java.util.Date;
import java.sql.Date;


public class TodayDate
{
        public static void main(String[] args)
    {
        Date myDate = new Date();
}
}
```

conflictos

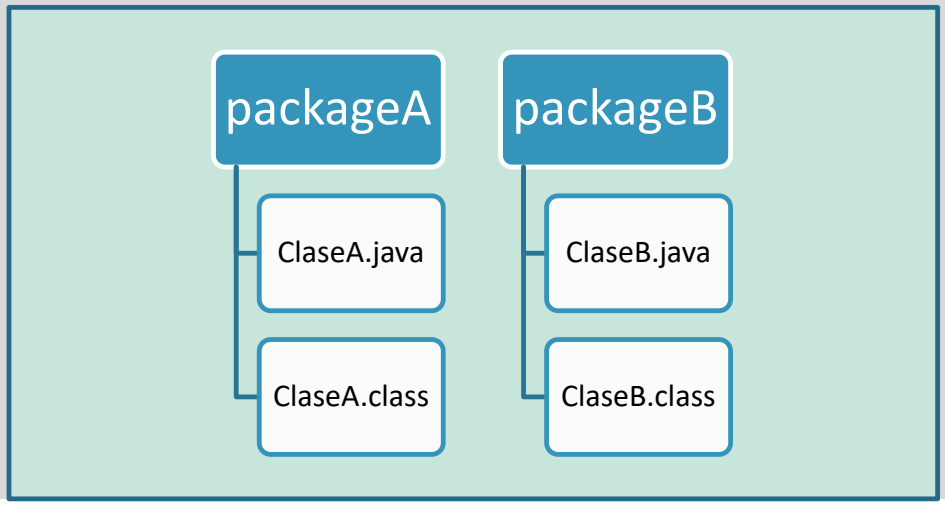
- Conflictos en imports
  - Tiene precedencia cuando se especifica la clase:
    - import java.util.Date;
    - import java.sql.*;
  - Si se requieren ambos el nombre completo de la clase:
    - java.util.Date myDate = new java.util.Date();
    - java.sql.Date myDate = new java.sql.Date(123);
- Creando packages
  - Hasta ahora se ha usado solo el package default.

CREANDO UN PROGRAMA CON PAQUETES

```
package packagea;

import packageb.ClaseB;

public class ClaseA{}
```



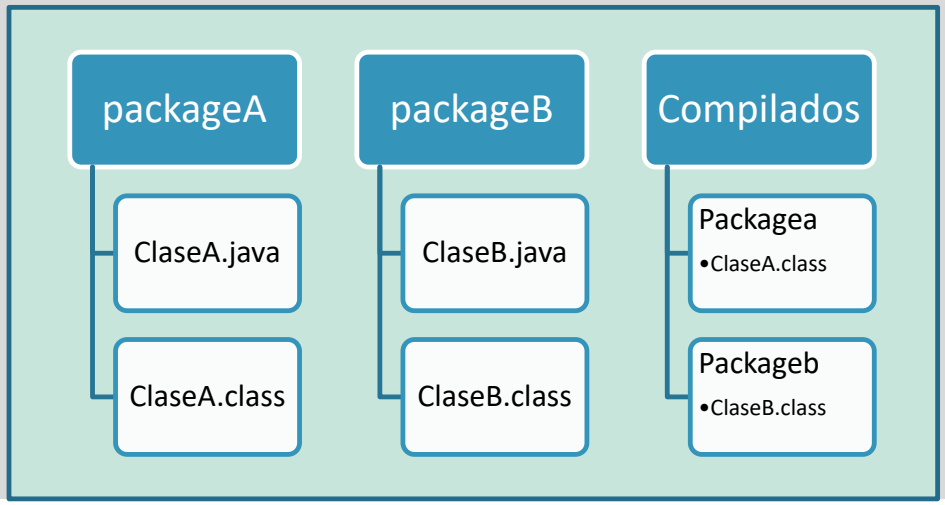
- Compilando y corriendo con diferentes packages
  - Funciona? >> java packagea/ClaseA.java
  - Compilación: >> javac packagea/ClaseA.java
  - Compilación con wildcards (*)
    >> javac packagea/*.java
  - Ejecución: >> java packagea.ClaseA
- El nombre de la clase + el nombre del paquete se denominan FQCN (Fully Qualified Class Name)
- Ordenamiento en la clase
  - PIC: package import class

CREANDO UN PROGRAMA CON PAQUETES

```
package packagea;

import packageb.ClaseB;

public class ClaseA{}
```



- Compilando y corriendo con diferentes packages
  - Directorio alternativo
    - Compilación
    - -d: directory

```
>> javac –d compilados packagea/ClaseA.java
```

    - ejecución

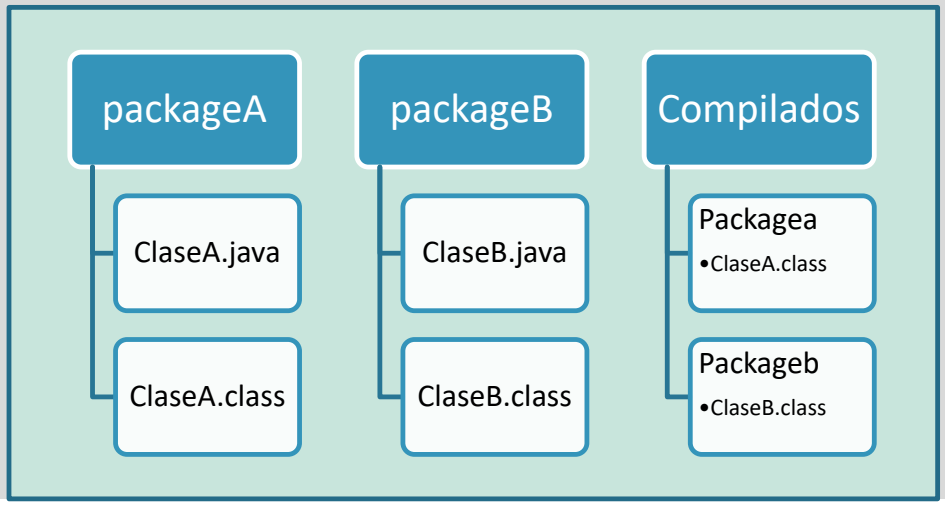
```
>> java -cp compilados packagea.ClaseA
>> java -classpath compilados packagea.ClaseA
>> java –class-path compilados packagea.ClaseA
```

CREANDO UN PROGRAMA CON PAQUETES

```
package packagea;

import packageb.ClaseB;

public class ClaseA{}
```



- Creando jar

```
>> jar –cvf myZip.jar .
>> jar --create --verbose --file myZip.jar .
```

  - Especificar el dir que se quiere zipear

```
>> jar –cvf myZip.jar –C dirDestino .
```

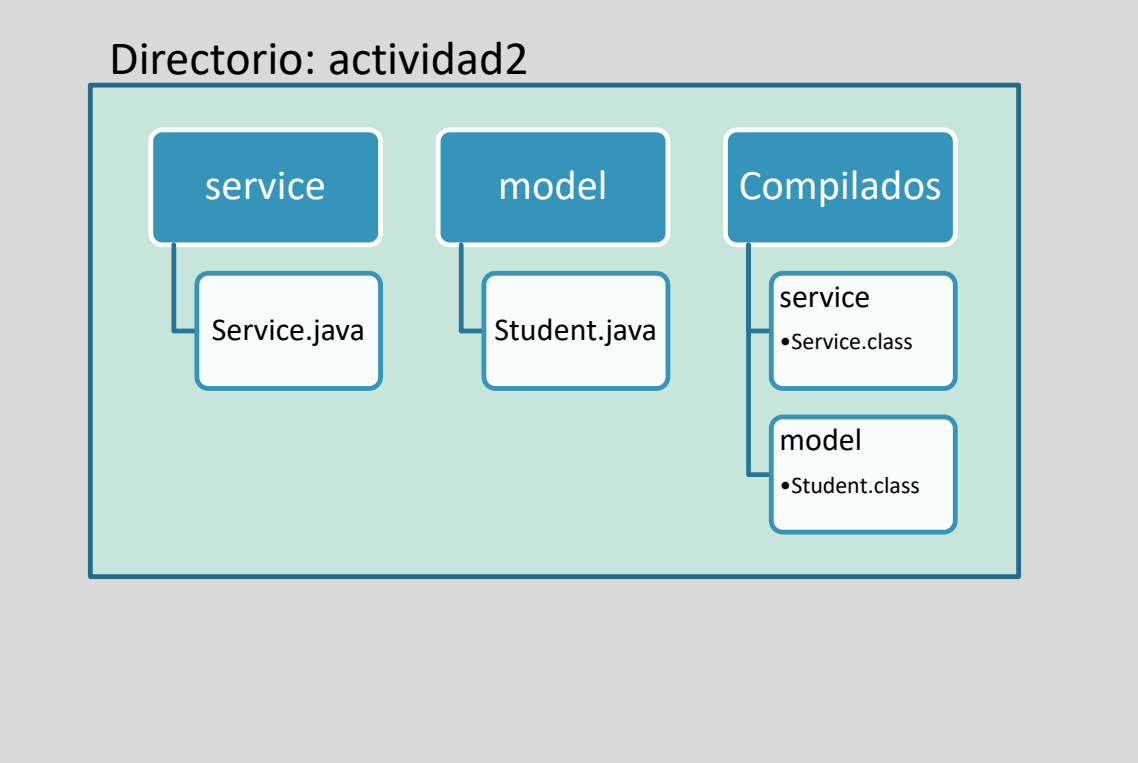
  - -- create: crea el archivo
  - -- verbose: imprime detalles
  - -- file: filename

ACTIVIDAD 1

```
public class Identifiers$
{
   public static void main(String[] args)
   {
        String $ = "Almendra";
        String $_ = "Paz";
        String $_3 = "Rodriguez";
        System.out.println("Valor $: "+ $);
      System.out.println("Valor $_: "+ $_);
      System.out.println("Valor $_3: "+ $_3);
   }
}
```

- Indicaciones
  - Cambie las variables para que contengan $ y _
  - Inserte comentarios múltiples delante de cada variable
  - Cambie el tipo del parámetro del método main por otro que compile correctamente

ACTIVIDAD 2



- Indicaciones
  - Cree el directorio actividad2 con la estructura de la izquierda
  - Cree la clase Student que solo tenga el siguiente método:

```
public String printName(){
System.out.println("hola alumno");
}
```

  - Cree la clase Service, la cual contenga un método main, donde instancie la clase Student y paso seguido llame a su método printName.
  - Coloque los compilados en el dir “compilados”
  - Ejecute desde el directorio “actividad2”

GRACIAS

ENLACE PARA FEEDBACK