Java For Testers

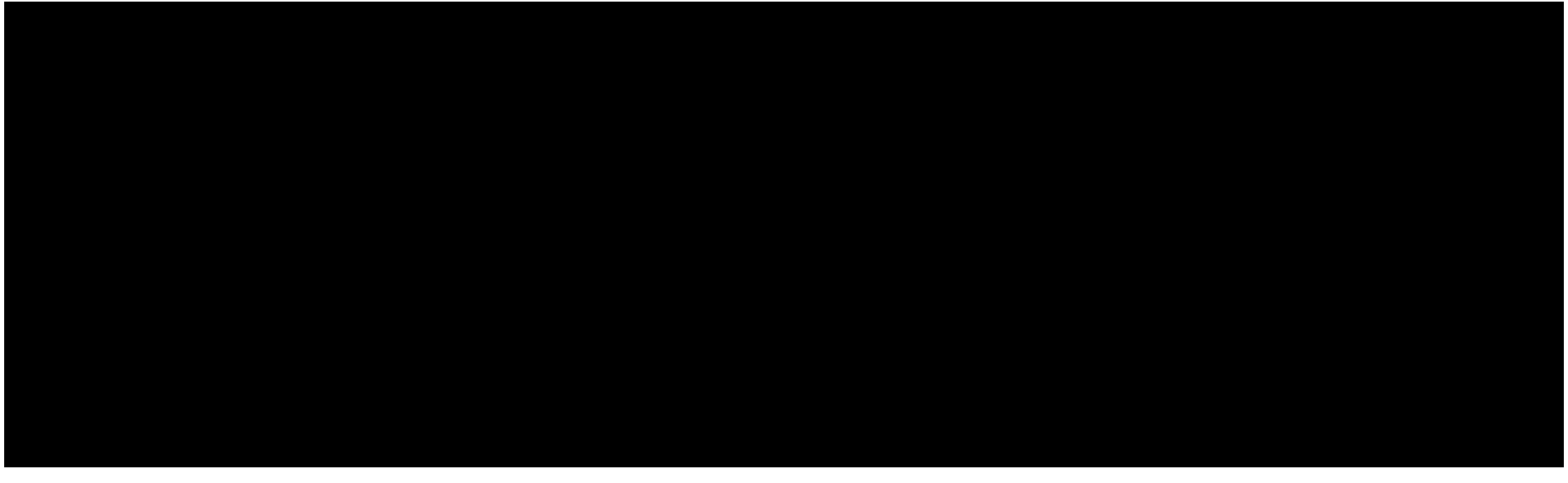
Fecha: 01/09/2025

3 Primeros Pasos con Java

1. Crear un proyecto Maven
2. Crear 1er clase Hola Java
3. Variables
    - String
    - int
    - double
    - boolean
    - Array
    - List
    - Hash Map
    - Variables Static
    - Variables de bloque VS Clase

3. Primeros Pasos con Java

3.1 Crear Proyecto en Maven



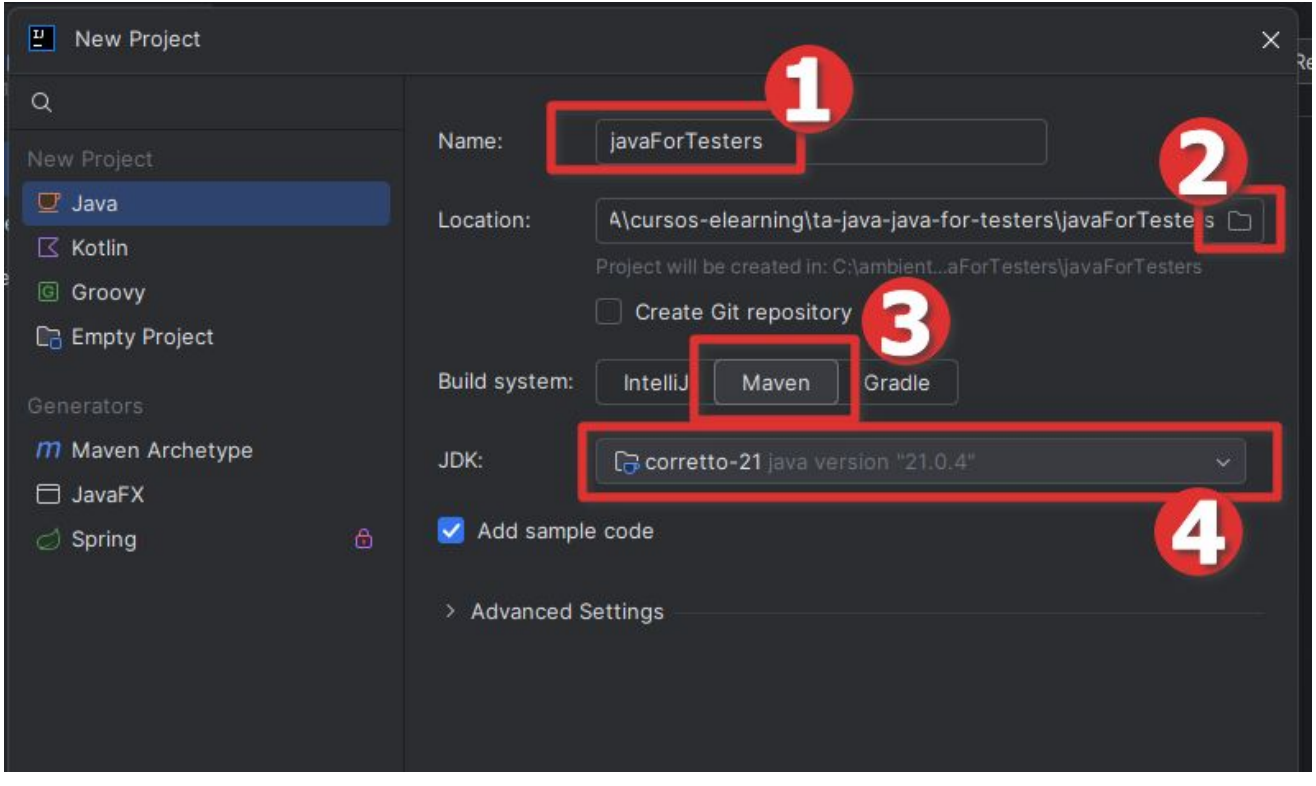
Maven

- ★ Maven es como un organizador automático de proyectos en Java.
- ★ Descargar librerias automáticamente
- ★ Mantener ordenado el proyecto
- ★ Ejecutar tareas rápidas.
- ★ Ej. tests -> mvn test
- ★ Hacerlo repetible y confiable

Por qué un Tester debería saber Java

- ★ **Errores**: Es probable que encuentres mensajes de error o fallos relacionados con Java. En un futuro ser un code reviewer
- ★ **Automatización**: Algunas herramientas de pruebas automatizadas (Selenium, Appium) se crean con Java
- ★ **Comunicación**: Si entiendes lo básico de Java, podrás hablar mejor con los desarrolladores cuando reportes problemas o pidas aclaraciones.

Pasos para crear Proyecto en Maven



```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                             http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <!-- Identidad del proyecto -->
    <groupId>com.testeracademy</groupId>
    <artifactId>mi-proyecto-qa</artifactId>
    <version>1.0-SNAPSHOT</version>

    <!-- Dependencias (las librerías que Maven descargará por ti) -->
    <dependencies>
        <!-- Selenium WebDriver -->
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>4.21.0</version>
        </dependency>
    </dependencies>

    <!-- Plugins (para compilar y correr) -->
    <build>
        <plugins>
            <!-- Plugins para ejecutar tareas -->
        </plugins>
    </build>
</project>
```

Actividad

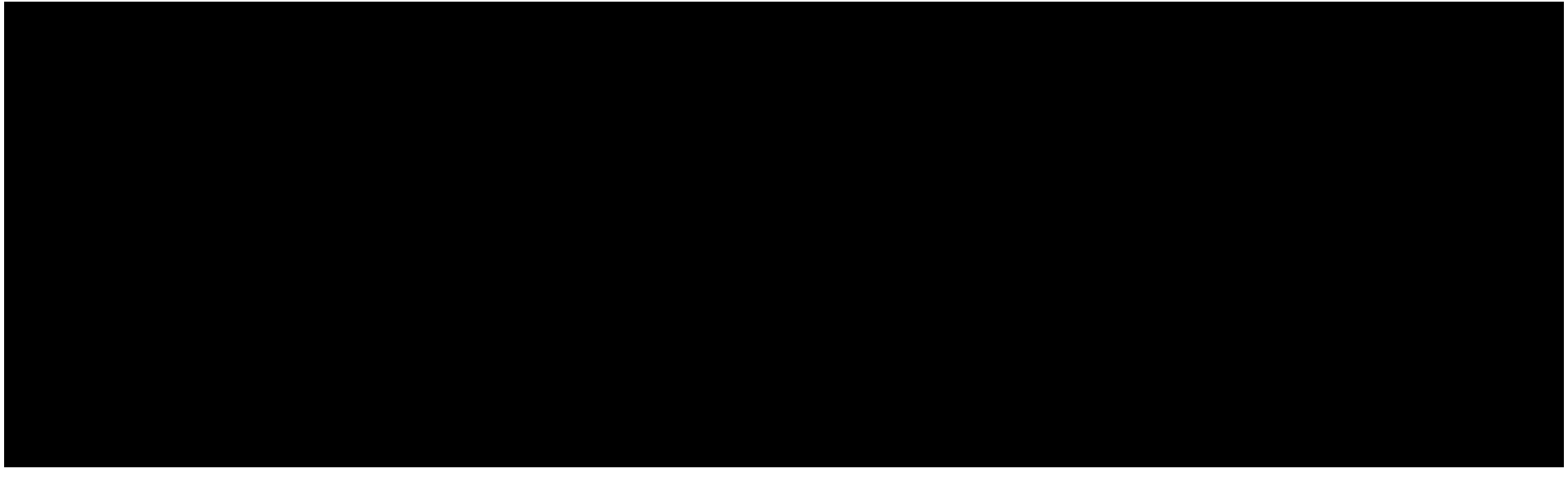
Realiza la siguiente actividad

1. Abrir IntelliJ
2. Crear tu primer proyecto Maven con el nombre **“javaForTesters”**
3. Explora el archivo **pom.xml**

****NOTA**

Los nombres de packages, directorios y archivos pueden cambiar en función de lo que el alumno elija no es forzoso mantener los nombre recomendados

3.2 Crear 1er Clase Hola Java



Piensa en una clase como un molde o plano. Ej. **“Plano de una casa”**

- ★ Con ese plano puedes construir muchas casas (objetos).
- ★ Todas las casas salen del mismo plano, pero cada una puede tener colores, muebles o dueños diferentes.

En Java, una clase define:

- ★ **Qué tiene** (atributos, como color, tamaño).
- ★ **Qué hace** (métodos, como abrir puerta, encender luz).

** Para imprimir información y variables vas empezar a usar **System.out.println**

** **1 archivo .java 1 Clase**

```
//Aquí van imports (importar librerías)

public class HolaJava {
    public static void main(String[]
args){
        System.out.println( "¡Hola Java,
soy
            tester!");
    }
}
```

Actividad

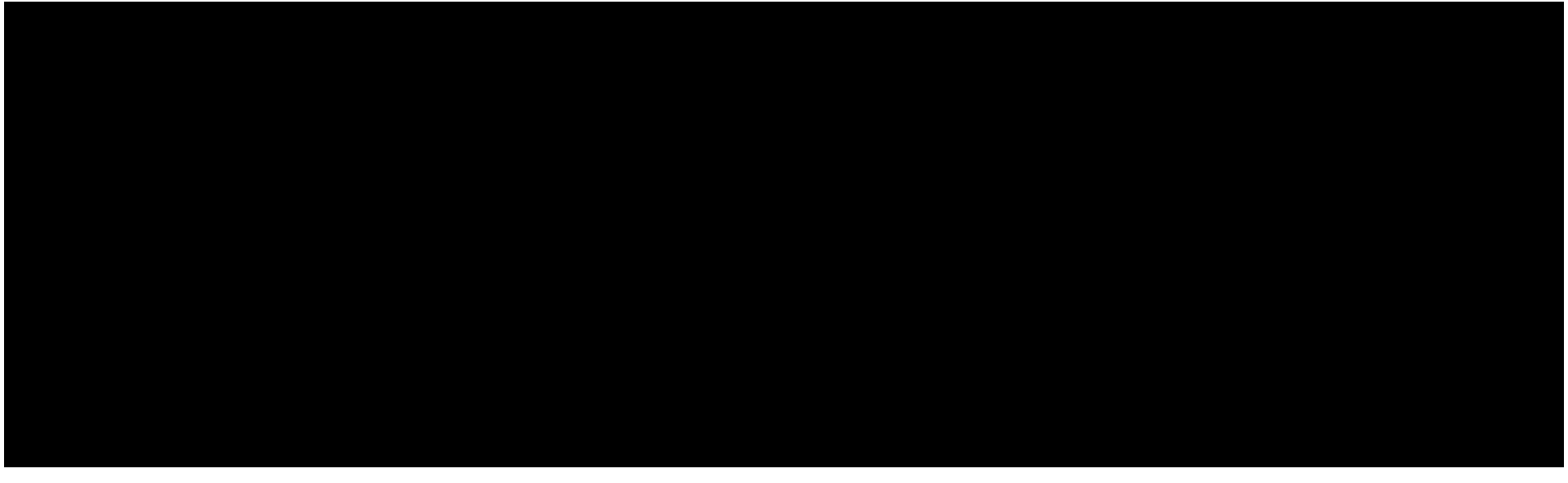
Realiza la siguiente actividad

1. Abre el proyecto previamente creado
2. Crea un package en src/test/java con el nombre **“cap03”**
3. Crea una clase dentro del paquete “src/test/java/cap03” con el nombre “HolaJava”
4. Imprime el mensaje “Hola Java”
5. Ejecuta el código
6. Agrega más print a tu código, exploralo, analizalo y vuelve a ejecutarlo

****NOTA**

Los nombres de packages, directorios y archivos pueden cambiar en función de lo que el alumno elija no es forzoso mantener los nombre recomendados

3.3 Variables



Variable	Ejemplo	Descripción
String	`String nombre = "Zoe";`	Texto o cadena de caracteres (palabras, frases).
int	`int edad = 7;`	Números enteros sin decimales.
double	`double precio = 19.99;`	Números con decimales (más precisión).
boolean	`boolean esTester = true;`	Solo puede ser verdadero o falso.
array	`int[] numeros = {1, 2, 3};`	Colección fija de valores del mismo tipo.
List	`List<String> frutas = new ArrayList<>();` `frutas.add("Manzana");`	Lista de elementos, más flexible que un array.
Hash Map	`HashMap<String, String> mapa = new HashMap<>();` `mapa.put("QA", "Tester");`	Guarda pares clave-valor (como una mini "tabla": llave → dato).

Tipo de dato	Primitivo	Referencia
Enteros	`int edad = 7;`	`Integer edad = 7;`
Decimales	`float precio = 19.99f;` `double precio = 19.99;`	`Float precio = 19.99f;` `Double precio = 19.99;`
Caracteres	`char letra = 'Z';`	`Character letra = 'Z';` `String nombre = "Zoe";`
Booleano	`boolean esTester = true;`	`Boolean esTester = true;`

PRIMITIVOS

- ★ Guarda solo el valor
- ★ Operaciones básicas como (sumar, restar, etc)

REFERENCIA

- ★ Guarda dirección de un objeto
- ★ Operaciones básicas como (sumar, restar, etc)
- ★ Objeto con métodos y acciones

Array

- ★ Tamaño fijo (no crece ni se reduce)
- ★ Más simple y básico.
- ★ Se usa cuando sabes exactamente cuántos datos tendrás.

Array List

- ★ Tamaño dinámico (puedes agregar o eliminar elementos).
- ★ Más flexible y fácil de usar.
- ★ Tiene métodos útiles (add, remove, get, set).

Mejoras List

- ★ No necesitas definir el tamaño desde el inicio.
- ★ Puedes modificar la lista en tiempo de ejecución.
- ★ Es más práctico para la mayoría de los casos en testing.

Collections

- ★ Clase de utilidad que permite a través de métodos estáticos hacer operaciones con las listas.
- ★ Operaciones que se pueden realizar “Ordenar, Desordenar, etc”

```java
// 1. List (Interface) - Polimorfismo
List<String> lista = new ArrayList<>();

// 2. ArrayList (Implementation)
ArrayList<String> listaConcreta = new
ArrayList<>();

// 3. Collections (Utility Class)
Collections.sort(lista);
Collections.reverse(lista);
List<String> listaVacia =
Collections.emptyList(); // Retorna una
lista inmutable vacía
```

Nombre	Edad
David	30
Josy	20

- ★ Piensa en 1 lista como 1 tabla de excel
- ★ Piensa en el hashmap como lectura de columnas

- ★ Objeto que lee KEY, VALUE
- ★ lista.get(0)
  - ○ 1ER Fila
  - ○ Es tipo hashmap
- ★ lista.get(0).get("Nombre")
  - ○ 1ER Fila, Columna "Nombre"
  - ○ Valor => David
- ★ lista.get(0).get("Edad")
  - ○ 1ER Fila, Columna "Edad"
  - ○ Valor => 30
- ★ lista.get(1).get("Nombre")
  - ○ 1ER Fila, Columna "Nombre"
  - ○ Valor => Josy

Variables Static

- ★ Son variables que **pertenecen a la clase**, no al objeto.
- ★ Se pueden usar sin crear un objeto.
- ★ Se acceden con el nombre de la clase directamente.
- ★ Útiles para valores compartidos (ej. constantes, contadores).
- ★ Su valor se carga desde que se ejecuta el programa

```
public class JavaForTesters{
    static String nombre = "Bicho";

    public static void main(String[] args){
      System.out.println(JavaForTesters.nombre);
    }
}
```

Variables de Bloque (Locales)

- ★ Se declaran dentro de un método, constructor o bloque { }.
- ★ Solo existen mientras se ejecuta ese bloque.
- ★ Cada vez que entras al método se crean de nuevo.

Variables de Clase

- ★ Se declaran dentro de la clase pero fuera de cualquier método.
- ★ Pertenecen al objeto (o a la clase si son static).
- ★ Existen mientras el objeto/clase viva en memoria.

```java
//Ejemplo Variable de Bloque
public class JavaForTesters {
    public static void main(String[] args) {
        int edad = 36; // variable de bloque,
                       vive solo dentro del main
        System.out.println("Edad: " + edad);
    }
}


//Ejemplo Variable de Clase
public class JavaForTesters{
    static String nombre = "Bicho";
    String apellido = "Belmont"; //Más adelante
                                 veremos qué es esto

    public static void main(String[] args){
      System.out.println(JavaForTesters.nombre);
    }
}
```

Actividad

Realiza la siguiente actividad

1. Abre el proyecto previamente creado
2. Crea una clase dentro del paquete “src/test/java/cap03” con el nombre “Persona”
3. Practica agregando variables a la persona static o de bloque, como prefieras
4. Imprime la información de la persona
5. Ejecuta el código

****NOTA**

Los nombres de packages, directorios y archivos pueden cambiar en función de lo que el alumno elija no es forzoso mantener los nombre recomendados

Actividad

Realiza la siguiente actividad

Te sugerimos esta actividad

1. Abre el proyecto previamente creado
2. Crea un nuevo archivo CSV de algo que te guste, mascotas.csv por ejemplo, agrega columnas e información
3. Intenta leer el archivo e imprimir información, deberás poner en práctica el uso de listas y hashmaps
4. No te preocupes por entender el código de lectura del CSV en utils, enfócate en poder hacer uso de él para leer tu archivo
5. Usa como referencia para trabajar el archivo: **“cap03.Ej10_HashMapListFromCSV.java”**

****NOTA**

Los nombres de packages, directorios y archivos pueden cambiar en función de lo que el alumno elija no es forzoso mantener los nombre recomendados

Actividad

Realiza la siguiente actividad

Te sugerimos esta actividad

1. Ve a la página del curso y resuelve el **Quiz 03**
2. Estos ejercicios te ayudarán a mejorar tu skill de Java
3. Te recomendamos intentar resolverlos por tu cuenta sin depender de una IA

****NOTA**

Los nombres de packages, directorios y archivos pueden cambiar en función de lo que el alumno elija no es forzoso mantener los nombre recomendados

Quizz

Quizz

Realiza los siguientes

- ★ Quiz 01
- ★ Quiz 02
- ★ Quiz 03

Recuerda:

- ★ Trata de no memorizar orden de las respuestas, entiende, asimila y aprende
- ★ En la página del curso, todos los capítulos tienen su acceso directo a los quizzes

Recursos del curso

Recursos

En esta sección encontrarás recursos más importantes del curso

- ★ Página del curso
- ★ Repositorio de Git
- ★ YouTube videos del curso

Si deseas cursos personalizados u otro servicio no dudes en contactarnos

https://testeracademy.com.mx/contacto

Gracias

🧠 “No aprendemos para aprobar pruebas, aprendemos para transformar realidades.”

👋 ¡Hasta pronto!