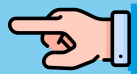
Flujo, ciclos y métodos

Utilizando Java

{desafío} latam_

Reconocer las características fundamentales del lenguaje Java para el desarrollo de aplicaciones empresariales

- Unidad 1: Flujo, ciclos y métodos
- Unidad 2: Arreglos y archivos
- Unidad 3: Programación orientada a objetos
- Unidad 4: Pruebas unitarias y TDD



Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Comprender las formas de trabajar en Java para ejecutar aplicaciones.*
- *Crear un proyecto para ejecutar bajo la Máquina Virtual de Java instalado.*

¿Cuáles son las formas de escribir un algoritmo?

{desafío} latam_

/* Utilizando Java */

Formas de trabajar en Java

Compilando por terminal (cmd)

1. Ir a la ruta donde está nuestro programa, usando el comando:

cd/”ruta_de_archivo”

2. Ejecutar el comando Javac y el nombre del programa.

Javac Nombre.Java

3. Luego, para ejecutarlo, se utiliza el comando Java.

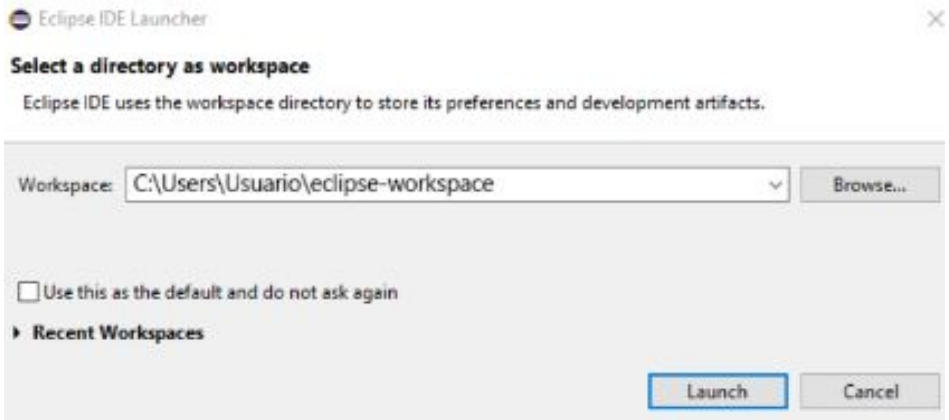
Java Nombre

4. Se realizarán las acciones del programa creado.

Formas de trabajar en Java

Eclipse

1. Seleccionar la ubicación del espacio de trabajo.



Al ejecutar por primera vez se desplegará la siguiente pantalla, en la cual debemos seleccionar dónde estará la ubicación de nuestro espacio de trabajo (es decir, los programas que vayamos creando), y damos clic a Launch.

Formas de trabajar en Java

Eclipse

2. Crear un nuevo proyecto en Java.

Luego aparecerá la pantalla con las distintas acciones que se pueden realizar con Eclipse, donde crearemos un nuevo proyecto Java.

Nuestro primer proyecto

Requerimientos previos

1. Java JRE y JDK
2. Eclipse IDE

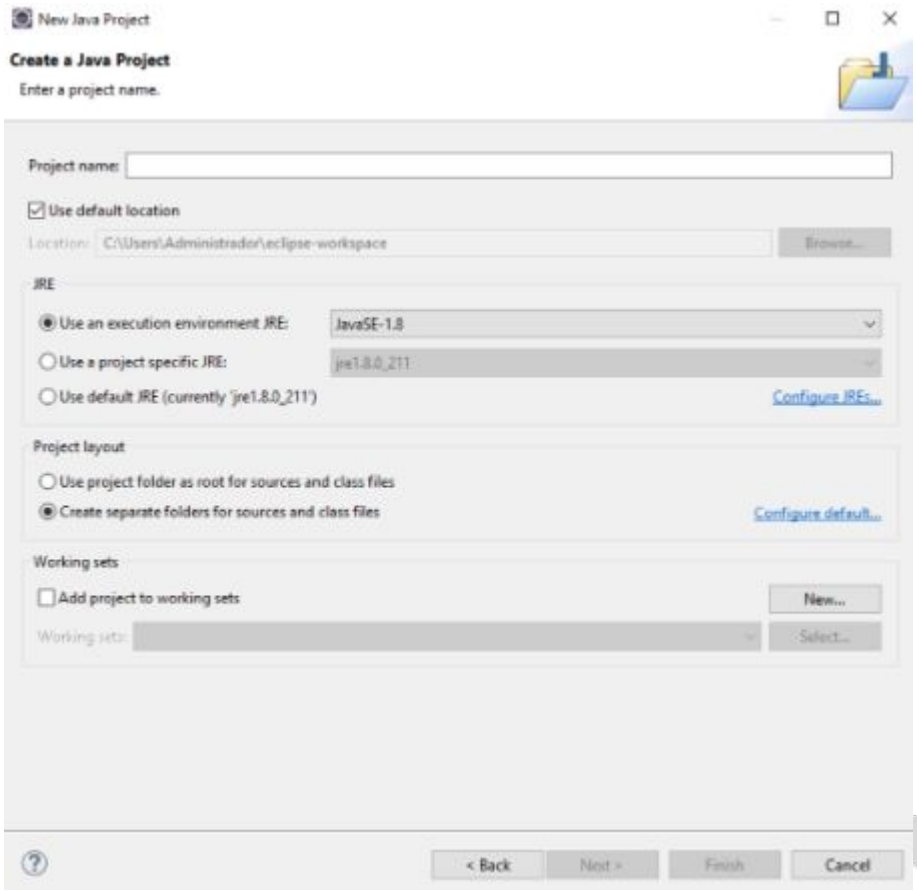
Crear la base de nuestro proyecto

1. Crear un nuevo proyecto en nuestro IDE (Eclipse).
2. Crear y organizar el proyecto en paquetes o package.
3. Crear la clase main o principal, que será la clase que ejecutará la lógica del programa.

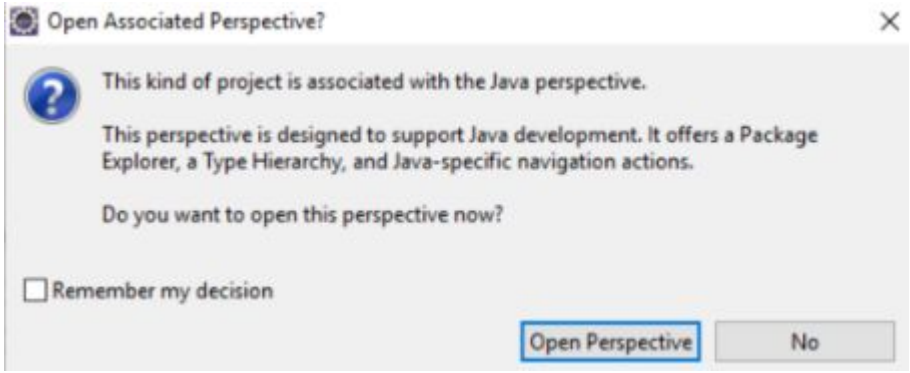
/* Al momento de dar clic al nuevo proyecto de Java, se desplegará una ventana donde debemos escribir el nombre del proyecto, elegir la ubicación y la versión de Java que se utilizará */

/* Para cada proyecto que creamos, podremos elegir indistintamente qué versión utilizar y no tendremos que estar preocupándonos por la versión que se está ejecutando en el terminal */

Al dar clic en el botón **Finish**, aparecerá un diálogo que nos preguntará si queremos asociar nuestro proyecto con la perspectiva de Java, la cual aceptaremos.



Asociar proyecto con perspectiva de Java.

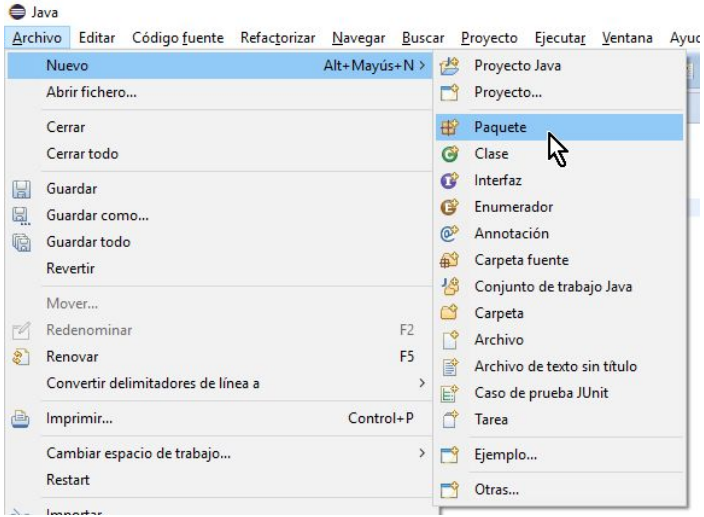


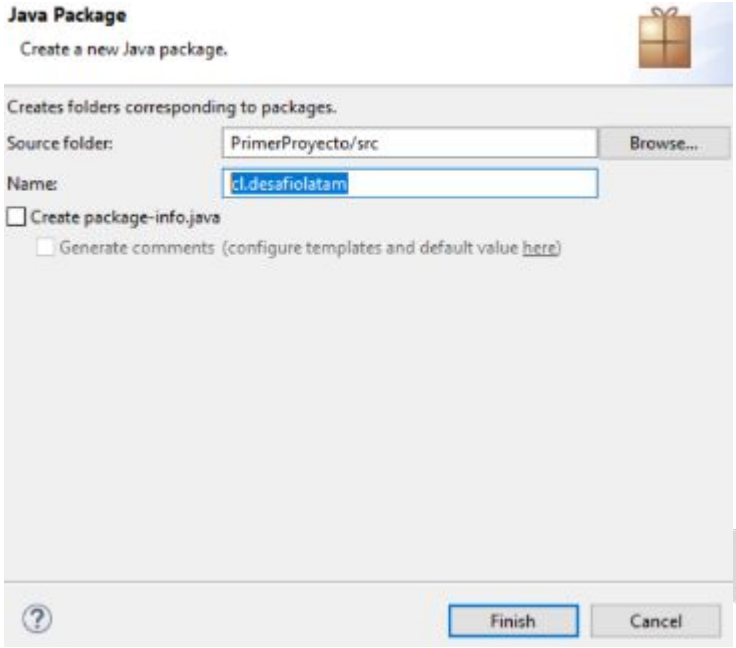
Revisemos el espacio de trabajo

- Del lado izquierdo tenemos el explorador de paquetes, es decir, donde tendremos nuestros ficheros a utilizar en el programa y estará el código Java.
- En la zona media, podremos ver el contenido de cada uno de los ficheros donde escribiremos el código.
- En la zona inferior se mostrarán los errores y cuando se ejecuta el programa.
- Debemos agregar la consola para ver la salida que obtendremos de nuestras ejecuciones.
- Para ello debemos ir al menú superior, **Window** → **Show View** → **Console**.

Crear el fichero donde escribiremos el código

1. Crear un paquete (package) en Eclipse. En la raíz del proyecto, haremos clic derecho sobre **src** → **New** → **Package**, y nombrarlo **cl.desafiolatam**

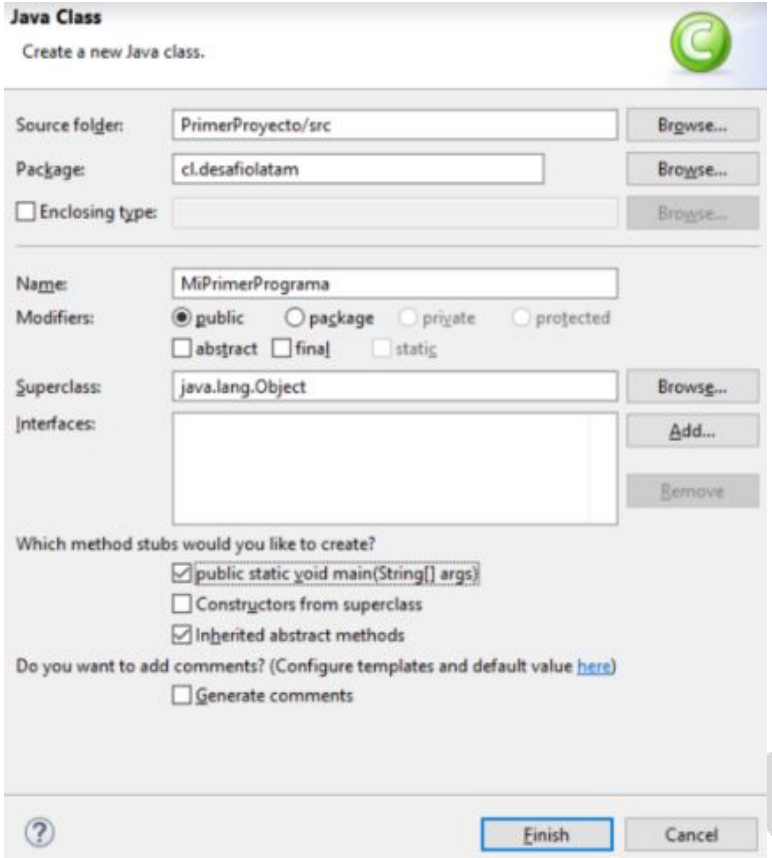




Crear el fichero donde escribiremos el código

2. Creación de clase main (Clase principal) en Eclipse.

Haremos clic derecho sobre el package **cl.desafiolatam** → **New** → **class**, a la cual le pondremos el nombre **MiPrimerPrograma**, donde aparecerán varias opciones. De momento marcaremos **public static void main (String[] args)** y **Finish**.



{desafío} latam_

Hola Mundo!

En Java

Ahora que tenemos nuestra clase creada, crearemos nuestro primer "Hola Mundo!"
Para hacer ello, usaremos el método **System.out.printf** que muestra por pantalla lo que necesitemos.
Hacer clic derecho sobre el paquete, y seleccionamos **Run As Java Application**.
En la zona inferior vemos que aparece una consola, donde dice "**Hola Mundo!**", qué fue lo que escribimos.

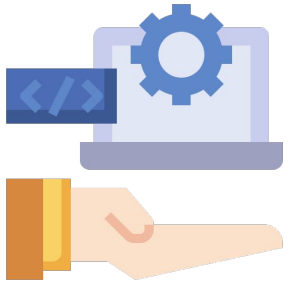
```
package cl.desafiolatam;
public class MiPrimerPrograma {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.printf("Hola Mundo!\n");
    }
}
```

Creando mi primer programa en Java

- Clic sobre **src** → **New** → Package, al cual le colocaremos el nombre **miprimerprograma**.
- Clic sobre el package **miprimerprograma** → new → class, a la cual le pondremos el nombre **MiPrimerPrograma**, donde aparecerán varias opciones.
- De momento marcaremos **public static void main(String[] args)** y **finalizar**.

Clase: **MiPrimerPrograma**
Paquete: **miprimerprograma**

Conceptos básicos del lenguaje



¿Qué debemos realizar para crear un código en Java?

Algoritmos, diagramas de flujo e implementación en Java

Para controlar el flujo y la lógica en cualquier programa en Java, debemos implementar algunos de los conceptos básicos que existen en cualquier lenguaje de programación:

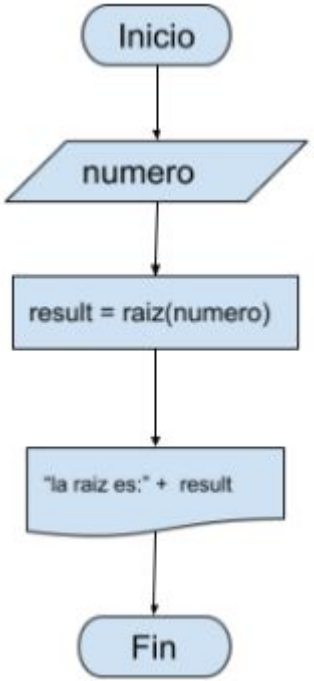
- **variables**
- **operadores**
- **estructuras de control** (condiciones).

Ejemplo

Crear un programa que calcule la raíz cuadrada de un número

En el diagrama de flujo expuesto, la operación de cálculo se representa mediante la palabra textual en español **raíz**, esto es debido a que el pseudocódigo o los algoritmos representados mediante diagrama de flujo deben ser autoexplicativos y tener lógica, pero no representa la implementación en un lenguaje en particular.

Además, el símbolo de "**la raíz es:" + result** representa una impresión por pantalla o bien en papel.



Implementación en Java

```
package cl.desafiolatam;
import java.util.Scanner;
public class MiPrimerPrograma {
	public static void main(String[] args) {
		/*Se imprimirá Ingrese un número: por pantalla */
		System.out.printf("Ingrese un número: ");
		/* La clase Scanner se utiliza para leer por consola un valor ingresado por el usuario */
		Scanner sc = new Scanner(System.in);
		/* número, recibe el valor ingresado por el usuario. nextLong devuelve un tipo de dato primitivo long */
		long numero = sc.nextLong();
		/* result, recibe el resultado del método sqrt de la clase Math, la cual calcula la raíz cuadrada de un número */
		double result = Math.sqrt(numero);
		/* finalmente, se imprime el resultado por pantalla con System.out */
		System.out.printf("La raíz cuadrada es: %f", result);
		/* Cerramos la clase Scanner */
		Sc.close();
	}
}
```

- La diferencia al traspasar el diagrama de flujo al código en lenguaje Java es precisamente el lenguaje.

- Debemos saber, por ejemplo:
  - el objeto Scanner para leer una variable por pantalla
  - long y double son tipos de datos primitivos
  - la clase Math nos ayuda a realizar operaciones matemáticas complejas

- Lo importante es la lógica del flujo y del programa más que el lenguaje o la técnica, ya que debemos considerar qué documentación para lo que queramos hacer hay mucha, solo debemos saber cómo implementarla.

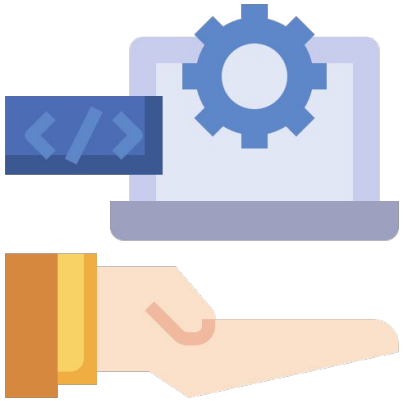
Conozcamos más sobre Java

{desafío} latam_

Sitios de interés

- Tutorial Java
  - Introducción a Java
  - Conceptos básicos de Java
  - Operadores de Java
  - Sentencias de control

- Conceptos básicos del lenguaje



Recomendaciones

Nombre de las clases y paquetes

- Si notaron, al definir el nombre de la clase se escribió la primera letra con mayúscula y, cada vez que empezaba una palabra nueva, vuelve a ser mayúscula: **MiPrimerPrograma**.
- En el caso del paquete, debe ser todo con minúscula: **cl.desafiolatam**.
- Esto se debe a que existe una convención para escribir el código en Java, no es mandatorio seguirlo estrictamente, pero se considera una *buena práctica*.

Próxima sesión...

- *Reconocer los elementos básicos de Java para usarlos e implementarlos al momento de construir algoritmos desarrollados en el lenguaje.*
- *Emplear comentarios, variables primitivas, variables de referencia a objetos, y constantes en Java.*

{desafío}
latam_

Academia de
talentos digitales