

Flujo (Parte I)

Introducción a la programación

Competencias

- Emplear algoritmos básicos mediante técnicas de pseudocódigos y diagramas de flujo para resolver un problema informático.
- Comprender la sintaxis del pseudocódigo para construir nuestro primer sistema informático.

Introducción

Si vamos a la etimología de la palabra *Programación*, esta nos define un plan ordenado y organizado de una serie de pasos para realizar una acción. Por ejemplo, un *programa* de televisión está compuesto por una serie de pasos que se deben seguir para cumplir el objetivo de dicho programa. Apertura, presentación, comerciales, hablar de política, presentar al siguiente invitado, despedida y cierre.

Pues, hoy en día las tecnologías informáticas o de la información, nos permiten resolver casi cualquier problema lógico (que se puede resolver por una serie de pasos) y así lograr optimizar en dar solución a ciertos problemas. Por ejemplo, antes de que existieran las calculadoras, realizar cálculos aritméticos complejos resultaba muy costoso.

En resumen, mediante la programación de software, podemos dar solución a esos problemas que resultan costosos de hacer en la vida diaria. Solo necesitamos conocer el flujo (la serie de pasos para resolver un problema de inicio a fin) y estos llevarlos a un lenguaje que nuestra máquina (computador, celular) pueda conocer.

Algoritmos

Se llaman *algoritmos* a la forma o técnica que tiene un programador de escribir o representar a una serie de pasos estructurados de manera lógica y ordenada para resolver un problema determinado. Por ejemplo:

Problema: Hacer un huevo frito.

Pasos para resolver el problema:

1. Encender el fuego en la cocina.
2. Poner la paila en el fuego.
3. Colocar aceite a la paila.
4. Romper el huevo.
5. Colocar el huevo en la paila.
6. Colocar sal al huevo.
7. Si se desea huevo revuelto.
 - a. Revolver.
8. Si se desea huevo entero.
 - a. Tapar la paila.
9. Esperar dos minutos.
10. Apagar el fuego.
11. Servir.

Lo anterior son los pasos que debemos pensar antes de escribir el algoritmo que representará de forma gráfica o escrita los pasos de la solución descrita.

Existen tres formas de escribir un algoritmo:

1. Diagramas de flujo (Forma gráfica).
2. Pseudocódigo (Forma escrita).
3. Implementado directamente en un lenguaje de programación (Forma escrita).

Diagramas de flujo

Un diagrama de flujo es una forma de representar de forma gráfica los pasos a seguir para solucionar un problema. Al ser una forma gráfica, se deben tener en consideración la simbología a utilizar para representar cada paso que formará el algoritmo.

Símbolos de un diagrama de flujo



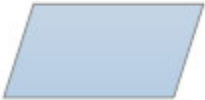
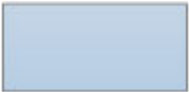

Símbolo	Nombre	Función
	Inicio / Final	Representa el inicio y el final de un proceso
	Línea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa la lectura de datos en la entrada y la impresión de datos en la salida
	Proceso	Representa cualquier tipo de operación
	Decisión	Nos permite analizar una situación, con base en los valores verdadero y falso

Imagen 1. Símbolos de un diagrama de flujo

Fuente: [Smartdraw](https://www.smartdraw.com).

Ejemplo (1)

Sumar dos números y mostrarlos en la salida.

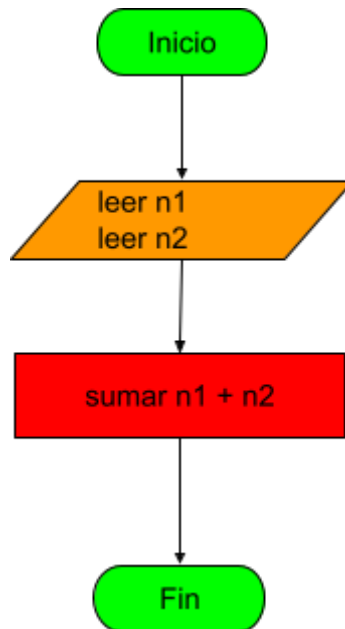


Imagen 2. Diagrama de flujo que suma dos números
Fuente: Desafío Latam.

Diagrama de flujo de la imagen que especifica los pasos a seguir para sumar dos números. El diagrama representa dos entradas de datos mediante las variables n1 y n2 (símbolo naranjo), las cuales guardarán el valor de cada número ingresado y el proceso de sumar (símbolo rojo), quien se encarga de representar el proceso de sumar dos números en cuestión.

Ejemplo (2)

Ingresar dos números y mostrar el mayor de ellos.

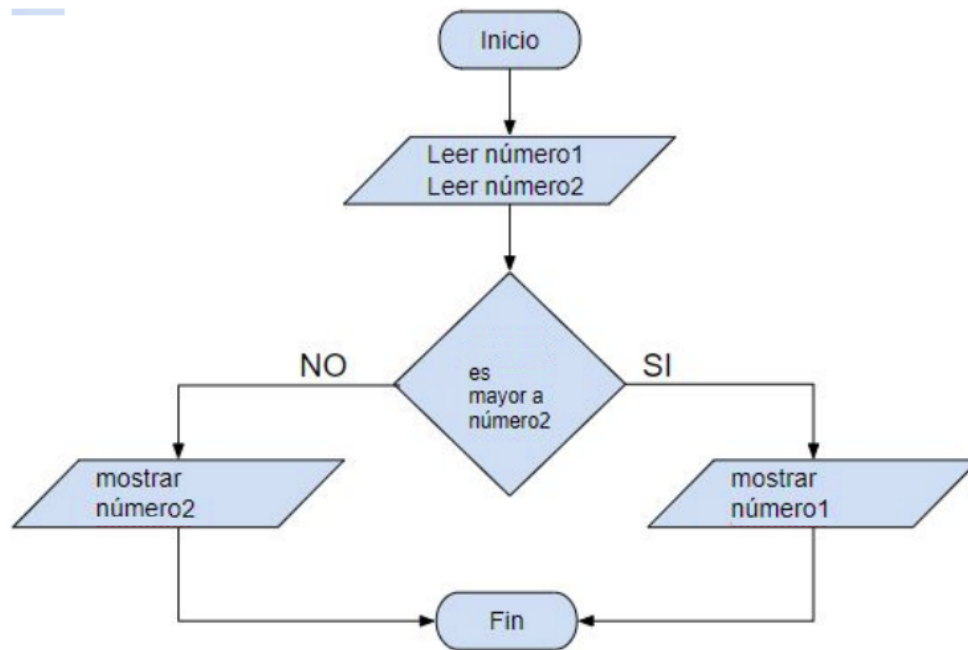


Imagen 3. Diagrama de Flujo que evalúa el mayor de dos números.

Fuente: Desafío Latam.

El diagrama de la imagen representa el algoritmo que da solución al problema de identificar cuál es el número mayor y mostrarlo por pantalla. En este flujo se implementa el rombo de decisión.

Ejercicio guiado: Cálculo de IVA

Crear diagrama de flujos de algoritmo que permita calcular el IVA en base al monto total de una factura.

- **Paso 1:** Se debe ingresar o leer el monto bruto de la factura.
- **Paso 2:** Calcular el IVA de la factura en base a la siguiente función:
$$\text{iva} = \text{montoBruto} * 0,19.$$
- **Paso 3:** Calcular el monto total o monto líquido de la factura en base a la siguiente función:
$$\text{montoLiquido} = \text{montoBruto} + \text{iva}.$$
- **Paso 4:** Se debe mostrar `monto bruto`, `iva` y `montoLíquido`.

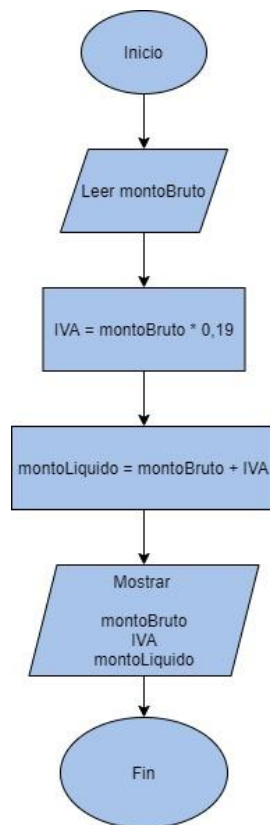


Imagen 4. Cálculo de IVA.
Fuente: Desafío Latam.

Pseudocódigo

Es otra forma de representar la solución de un algoritmo, de la manera más detallada posible, diseñada para la lectura, el cual puede ser parecida al lenguaje que posteriormente se utilizará para la codificación del mismo.

Ejemplo:

```
Algoritmo Suma
  Leer valor1
  Leer valor2
  Mostrar valor1 + valor2
FinAlgoritmo
```

En pseudocódigo se utiliza la instrucción **leer** para especificar que el usuario tiene que ingresar un valor y **mostrar** para imprimir el valor en pantalla.

Ventajas del pseudocódigo

El pseudocódigo, al igual que el diagrama de flujo, nos permite pensar en términos independientes al lenguaje de programación y concentrarnos en describir lo que estamos tratando de hacer y los pasos necesarios, en lugar de cómo lograrlo.

Ejemplos de pseudocódigo

Ejemplo 1

Crear un programa que permita calcular el área de un cuadrilátero, sabiendo que esta se calcula con la siguiente fórmula: $\text{área} = \text{base} \times \text{altura}$. Se debe leer por pantalla los valores para realizar el cálculo.

```
Algoritmo AreaCuadrilatero
    Leer base
    Leer altura
    Mostrar base * altura
FinAlgoritmo
```

En el ejemplo anterior, el signo `*` nos permite representar una multiplicación. Así mismo, si queremos realizar una división, podemos utilizar el signo `/`.

Ejemplo 2

Problema: Se necesita leer dos fechas de nacimiento de dos personas y mostrar por pantalla cuál de las dos personas es mayor.

```
Algoritmo QuienEsMayor
  Leer nombrePersona1
  Leer fechaNacimiento1

  Leer nombrePersona2
  Leer fechaNacimiento2

  si fechaNacimiento1 es mayor a fechaNacimiento2 entonces
    Mostrar nombrePersona1 + " es mayor a " + nombrePersona2
  si no
    si fechaNacimiento2 es mayor a fechaNacimiento1 entonces
      Mostrar nombrePersona2 + " es mayor a " + nombrePersona1
    si no
      Mostrar "Las personas tienen la misma edad"
FinQuienEsMayor
```

En este ejemplo, se ha utilizado el operador "+", el cual nos permite, aparte de hacer una suma aritmética entre dos números, realizar una concatenación de dos cadenas de caracteres (palabras). Por ejemplo:

- El usuario ingresa en `nombrePersona1` (a esto llamamos variable) el nombre de "Pepito". Entonces si queremos escribir por pantalla Pepito clavó un clavito, esto sería: `nombrePersona1 + " clavó un clavito"`.

De pseudocódigo a diagrama de flujos

Como nos podemos dar cuenta, un algoritmo es una serie de pasos o instrucciones secuenciales que se escriben de inicio a fin. Pues, al ser el pseudocódigo y los diagramas de flujo dos formas de representar un algoritmo, un mismo problema podemos escribirlo de ambas formas. Por ejemplo, el problema de saber quién de dos personas es mayor en base a su fecha de nacimiento:

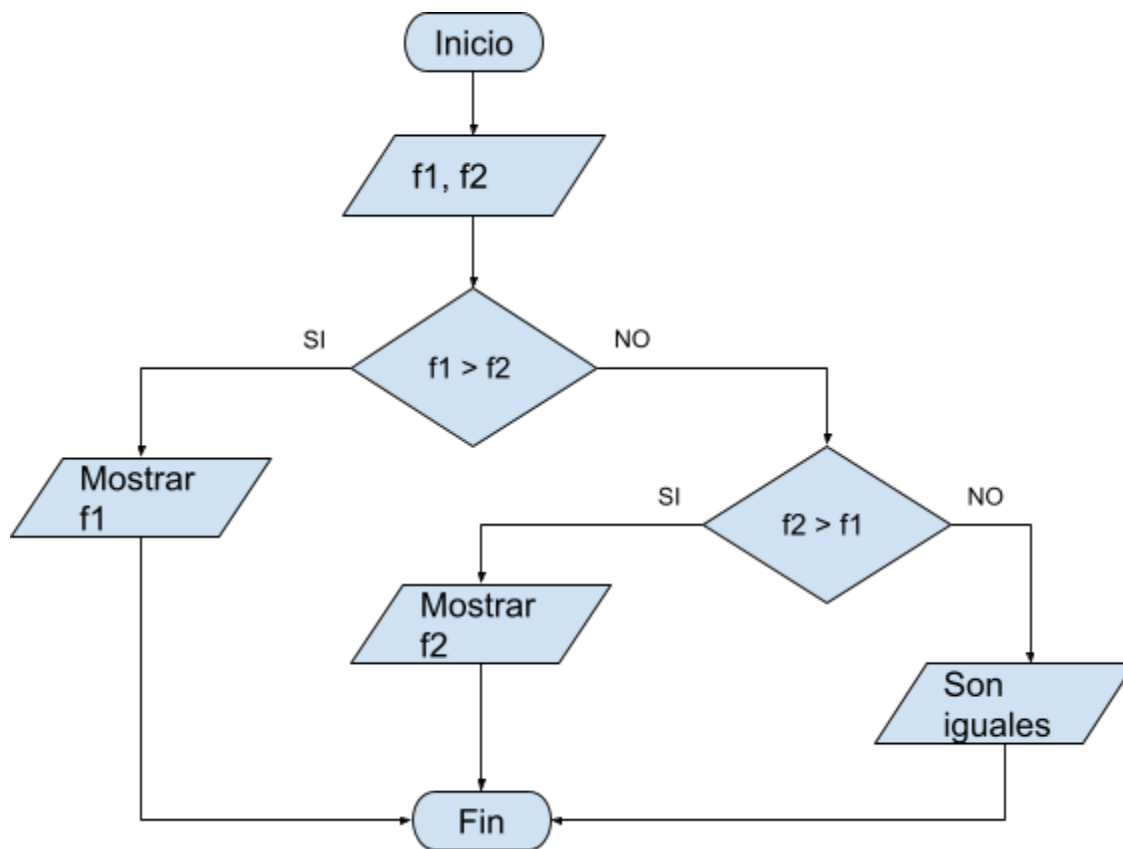


Imagen 5. Diagrama de flujo quién es mayor.
Fuente: Desafío Latam.

El diagrama de flujo de la imagen, evalúa dos fechas de nacimiento ingresadas e identifica cuál de ellas es mayor o bien si son iguales. Nótese que las palabras del flujo en este caso al ser un diagrama gráfico, no es tan literal. Es decir, se utilizan signos $>$ o $<$ para decir si es mayor o menor.

Ejercicio propuesto (1)

Crear un diagrama de flujo junto a su pseudocódigo que permita calcular el promedio de 3 notas y mostrar si el alumno aprueba o reprueba.

Requerimientos:

1. Ingresar 3 notas entre 1 y 7.
2. Calcular el promedio de notas en base a la siguiente función:

```
promedio = nota1 + nota2 + nota3 / cantidad De Notas (3 en este caso)
```

3. Si el promedio es menor a 4 se mostrará por pantalla que el alumno está reprobado.
4. Si el promedio es igual o mayor a 4, se mostrará por pantalla que el alumno está aprobado.

Introducción a Java

Competencias

- Comprender el lenguaje Java para así tener una primera introspectiva de la tecnología de programación.
- Ejecutar Java en el sistema Operativo Windows para comenzar a crear los primeros programas desarrollados en el lenguaje.

Introducción

Java actualmente es conocido como uno de los lenguajes de programación más extendidos y usados en el mundo, con más de 9 millones de desarrolladores además de estar presente en más de 7 mil millones de dispositivos. Con el surgimiento de Android, Java queda establecido como el lenguaje de programación más extendido del mundo. Android es utilizado en casi 90% de los smartphones en la actualidad, donde el lenguaje principal de Android es Java.

JVM

La Máquina Virtual Java o por sus siglas en inglés, *Java Virtual Machine* (JVM), es el entorno en el que se ejecutan los programas Java, su misión principal es la de garantizar la portabilidad de las aplicaciones Java.

Cuando se compila una aplicación escrita en lenguaje Java, en realidad no se compila en lenguaje de máquina del sistema operativo del dispositivo, sino a un lenguaje intermedio denominado "Byte Code".

¿Por qué utilizar Java?

- Java es independiente de la plataforma, ya que utiliza la Máquina Virtual de Java (JVM).
- Es un lenguaje orientado a objetos que es el paradigma que más se acerca a la manera de pensar del ser humano.
- Para crear aplicaciones Android se necesita saber Java.
- La comunidad de Java tiene disponible un gran soporte y documentación tanto en inglés como en español.
- Se utiliza en aplicaciones del mundo real. Está en varias aplicaciones web vistas antes.
- Tiene librerías estándar y variedad de IDEs que aportan gran ayuda a la programación.

JRE y JDK

- **Java JRE:** Java Runtime Environment es el paquete que contiene todo lo necesario para correr un programa ya compilado en Java, incluyendo la Java Virtual Machine (JVM), entre otros.
- **Java JDK:** Java Development Kit, contiene todo lo que trae JRE y, además, incluye el compilador (Javac), por lo que es capaz de crear y compilar programas.

Elección de un Entorno de Desarrollo Integrado (IDE)

Para desarrollar en Java existe una gran variedad de IDEs que se pueden utilizar, tales como NetBeans, Eclipse, Android Studio, BlueJ, entre otros, como también un simple editor de texto como Bloc de notas.

Ventajas de un IDE

- Tienen soporte del lenguaje. Nos facilita un editor de texto donde podemos ir escribiendo directamente.
- Permite depurar el código, facilitando escribir el código de manera más rápida y correcta, ya que en tiempo real nos sugiere cambios en caso que no hayamos escrito correctamente.
- Nos proporciona una interfaz gráfica donde podemos ver todo nuestro código de manera ordenada.
- Permite compilar de manera directa sin tener que usar el terminal.

Eclipse IDE

Para trabajar con Java utilizaremos Eclipse, que deberán descargar e instalar desde su sitio web [Eclipse.org](https://eclipse.org).

Al momento de abrir la instalación debemos elegir instalar Eclipse IDE for Enterprise Java Developers.



Imagen 6. Seleccionar Eclipse IDE for Enterprise Java Developers.

Fuente: [Eclipse](https://eclipse.org).

Los ejemplos que se mostrarán de aquí en adelante serán trabajados en Eclipse.

Instalando Java

Antes de comenzar, debemos saber como instalar Java en nuestro sistema operativo, de tal manera que podamos y logremos compilar y ejecutar nuestros programas en la máquina en que estemos trabajando.

Antes de todo, debemos cumplir con los requisitos mínimos para la instalación de Java en nuestra estación de trabajo.

Requisitos

Windows

- Windows 10 (8u51 y superiores).
- Windows 8.x (escritorio).
- Windows 7 SP1.
- Windows Vista SP2.
- Windows Server 2008 R2 SP1 (64 bits).
- Windows Server 2012 y 2012 R2 (64 bits).
- RAM: 128 MB.
- Espacio en disco: 124 MB para JRE; 2 MB para Java Update.
- Procesador: Mínimo Pentium 2 a 266 MHz.
- Exploradores: Internet Explorer 9 y superior, Firefox.

Mac OS X

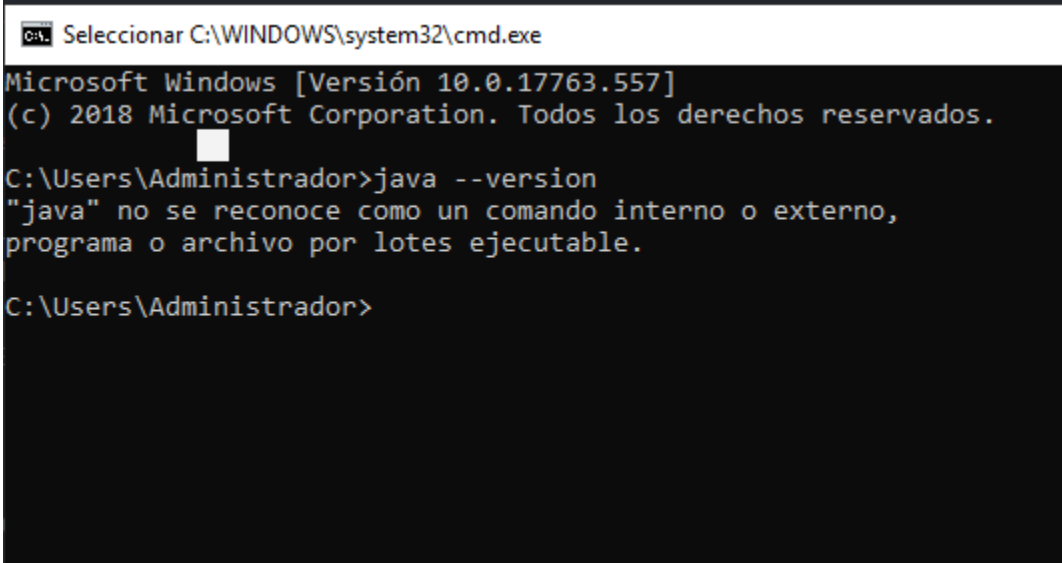
- Mac con Intel que ejecuta Mac OS X 10.83+ 10.9+.
- Privilegios de administrador para la instalación.
- Explorador de 64 bits (Safari, por ejemplo) para ejecutar Oracle Java en Mac.

Linux

- Oracle Linux 5.5+1.
- Oracle Linux 6.x (32 bits), 6.x (64 bits)2.
- Oracle Linux 7.x (64 bits)2 (8u20 y superiores).
- Red Hat Enterprise Linux 5.5+1, 6.x (32 bits), 6.x (64 bits)2.
- Red Hat Enterprise Linux 7.x (64 bits)2 (8u20 y superiores).
- Suse Linux Enterprise Server 10 SP2+, 11.x.
- Suse Linux Enterprise Server 12.x (64 bits)2 (8u31 y superiores).
- Ubuntu Linux 12.04 LTS, 13.x.
- Ubuntu Linux 14.x (8u25 y superiores).
- Ubuntu Linux 15.04 (8u45 y superiores).
- Ubuntu Linux 15.10 (8u65 y superiores).
- Exploradores: Firefox.

Instalando Java en Windows

1. Primero debemos verificar si nuestro sistema ya tiene una versión de Java instalada. Debemos abrir una ventana CMD, dando clic derecho sobre el ícono de Windows y seleccionar "Ejecutar" y escribir cmd.
2. Para verificar la versión de Java, ejecutamos el comando `Java -version`. Si el comando devuelve un mensaje de error es porque Java no está instalado.



```
C:\> Seleccionar C:\WINDOWS\system32\cmd.exe

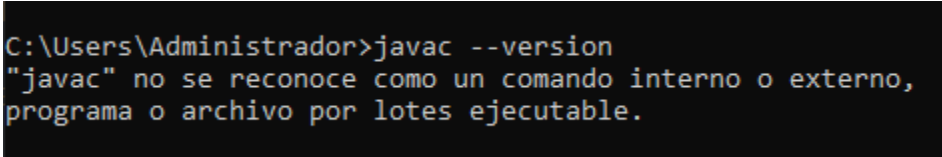
Microsoft Windows [Versión 10.0.17763.557]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Administrador>java --version
"java" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Users\Administrador>
```

Imagen 6. Comprobando la versión de Java.
Fuente: Desafío Latam.

Si la ventana de comandos nos indica una versión de Java, quiere decir que el JRE está presente, por lo que debemos verificar si el JDK está también instalado. Para ello, utilizaremos el comando `Javac`.



```
C:\Users\Administrador>javac --version
"javac" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.
```

Imagen 7. Comando cmd Javac --version.
Fuente: Desafío Latam.

Si al ejecutar el comando indicado en la imagen anterior, nos indica que Javac no se reconoce, debemos instalar el JDK.

Descargar Java para Windows / Ubuntu / macOS

1. Lo primero que debemos hacer es descargar el Java SE Development Kit, directamente de la página de Oracle. [Link descarga.](#)

Java SE 8u211 / Java SE 8u212
Java SE 8u211 / Java SE 8u212 includes important bug fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release.
[Learn more](#)

- Installation Instructions
- Release Notes
- Oracle License
- Java SE Licensing Information User Manual
 - Includes Third Party Licenses
- Certified System Configurations
- Readme Files
 - JDK ReadMe
 - JRE ReadMe

JDK
DOWNLOAD

Server JRE
DOWNLOAD

JRE
DOWNLOAD

Imagen 8. Descarga Java SE Development Kit.
Fuente: Desafío Latam.

Java SE Development Kit 8u211

You must accept the [Oracle Technology Network License Agreement for Oracle Java SE](#) to download this software.

☐ Accept License Agreement
☒ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.86 MB	jdk-8u211-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.76 MB	jdk-8u211-linux-arm64-vfp-hflt.tar.gz
Linux x86	174.11 MB	jdk-8u211-linux-i586.rpm
Linux x86	188.92 MB	jdk-8u211-linux-i586.tar.gz
Linux x64	171.13 MB	jdk-8u211-linux-x64.rpm
Linux x64	185.96 MB	jdk-8u211-linux-x64.tar.gz
Mac OS X x64	252.23 MB	jdk-8u211-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	132.98 MB	jdk-8u211-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.18 MB	jdk-8u211-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	133.57 MB	jdk-8u211-solaris-x64.tar.Z
Solaris x64	91.93 MB	jdk-8u211-solaris-x64.tar.gz
Windows x86	202.62 MB	jdk-8u211-windows-i586.exe
Windows x64	215.29 MB	jdk-8u211-windows-x64.exe

Imagen 9. Selección de sistema operativo.
Fuente: Desafío Latam.

Para la instalación debemos seguir los pasos del ejecutable, seleccionar la ubicación de la instalación, y debemos esperar a que finalice.

2. Una vez terminada la instalación, deberemos agregar a las variables de entorno de Windows la ruta donde quedó instalado nuestro jdk.
 - Abrir Panel de Control -> Sistema -> Configuración avanzada del sistema.
 - Bajo la pestaña Opciones avanzadas, entrar a Variables de entorno. Nota: Usualmente la ruta donde se instala Java es de esta forma C:\Program Files\Java\jdk1.8.0_211.
 - En variables de usuario, crearemos una nueva variable y la llamaremos JAVA_HOME y el valor de la variable será C:\Program Files\Java\jdk1.8.0_211

Cabe destacar que la ruta **C:\Program Files\Java\jdk1.8.0_211** es referencial, y debe corresponder a la que tendrá cada uno instalada en su computador. Se recomienda ir a la ruta o carpeta y verificar la ruta de instalación.

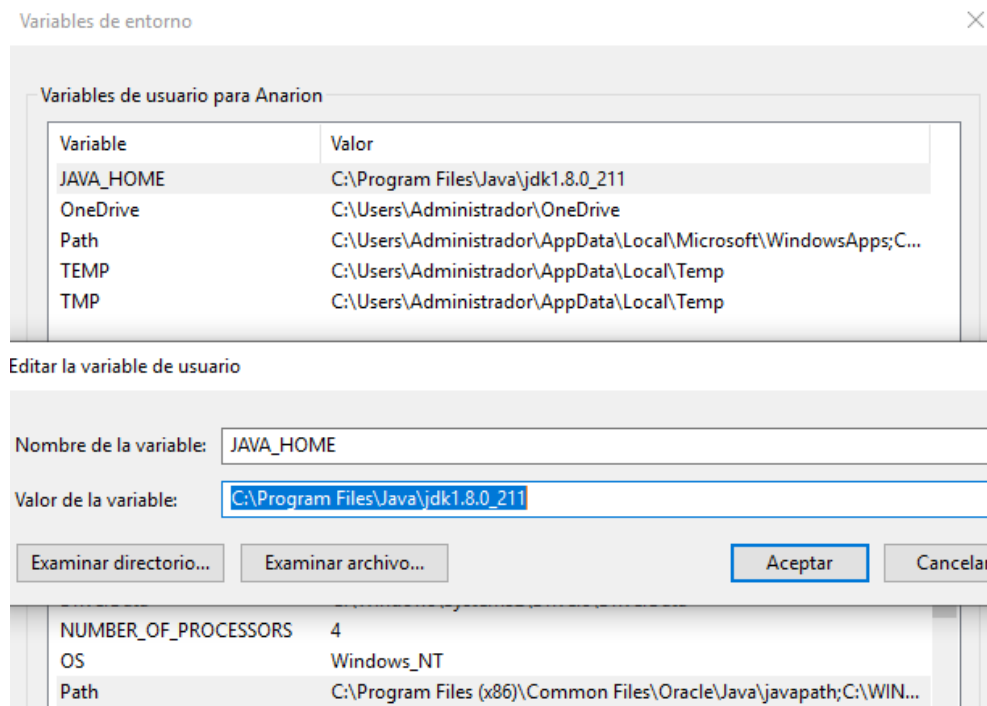


Imagen 10. Variables de entorno.
Fuente: Desafío Latam.

- Ahora en variables de sistema, editamos la variable PATH (si no existe, crearla) y agregar `C:\Program Files\Java\jdk1.8.0_211\bin`

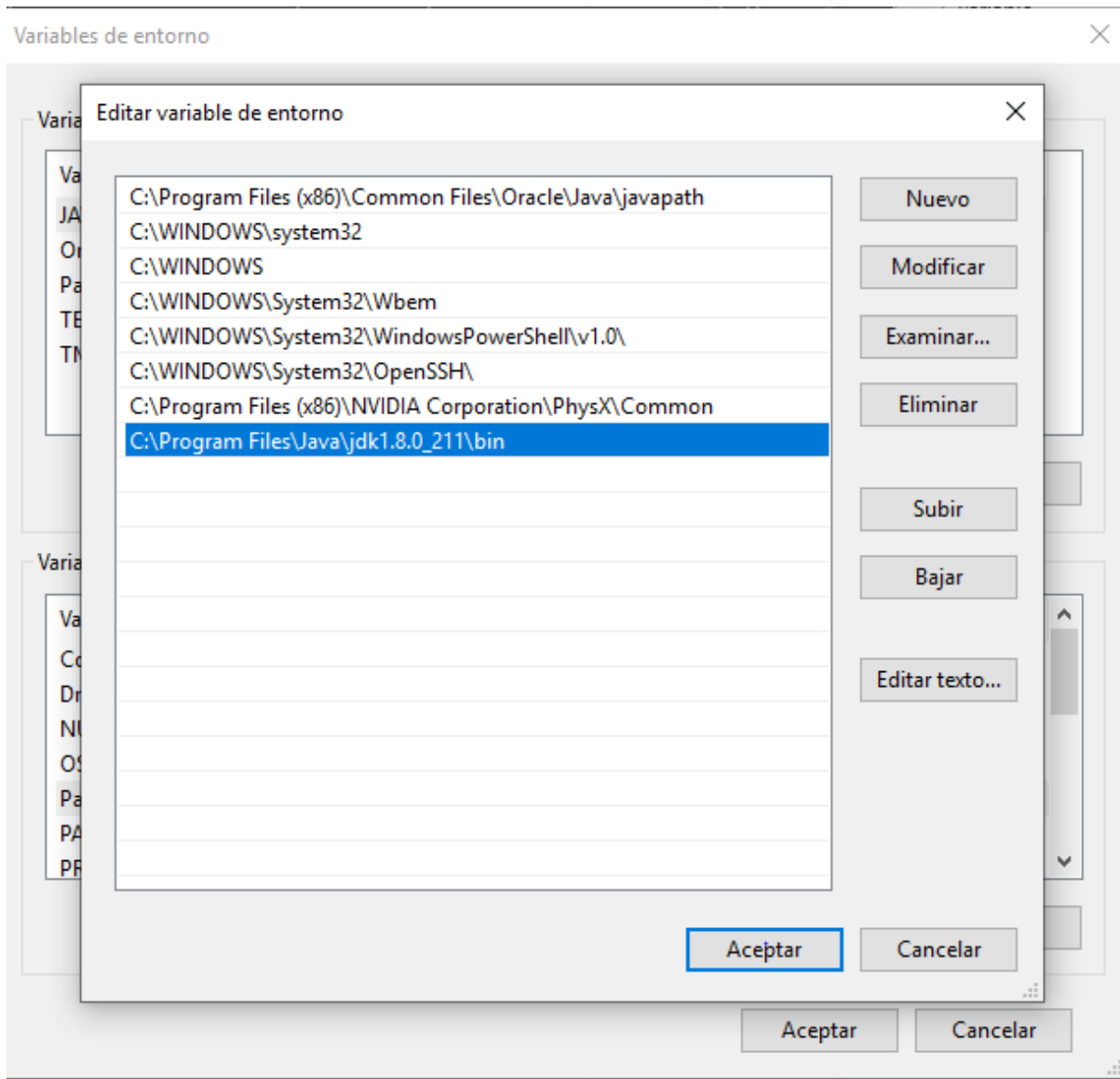


Imagen 11. Configuración variables del sistema.
Fuente: Desafío Latam.

Comprobar la instalación correcta

Si tenemos abierto Símbolos del Sistema (cmd), debemos cerrarlo y abrirlo nuevamente y ejecutar el comando: `Java -version`, y `Javac -version`.

```
Microsoft Windows [Versión 10.0.17763.557]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.
C:\Users\Administrador>Java -version
Java version "1.8.0_211"
Java(TM) SE Runtime Environment (build 1.8.0_211-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.211-b12, mixed mode)
C:\Users\Administrador>Javac -version
Javac 1.8.0_211
C:\Users\Administrador>
```

Instalación en macOS y Ubuntu

Al igual que Windows, deberán descargar Java desde el link anterior e instalarlo.

Una vez terminada la instalación, deberán comprobar ejecutando los comandos `java -version`, y `javac -version`

Utilizando Java

Competencias

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

Introducción

Antes de comenzar debemos instalar Java en nuestro equipo, para eso se recomienda revisar el documento **Instalando Java - Lectura complementaria** ubicado en “Material Complementario”.

A continuación, tendremos el primer acercamiento a programar con Java, donde revisaremos cómo trabajar con Java ya sea por terminal (CMD) o utilizando Eclipse para crear nuestro primer código.

Formas de trabajar en Java

Al trabajar pequeños trozos de código no existe gran diferencia entre trabajar con un IDE o un editor de texto, sin embargo, en programas un poco más complejos, es mucho más eficiente contar con las herramientas que provee el IDE, optimizando el tiempo de desarrollo.

Compilando por terminal (cmd)

Al crear un programa en Java, este deberá tener el formato de archivo Nombre.Java.

En el terminal, debemos ir a la ruta donde está nuestro programa (usando el comando `cd /"ruta_de_archivo"`) y ejecutar el comando `Javac` y luego el nombre del programa.

```
Javac Nombre.Java
```

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

```
Java Nombre
```

Y éste deberá realizar las acciones del programa creado.

Utilizando Java en Eclipse IDE

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.

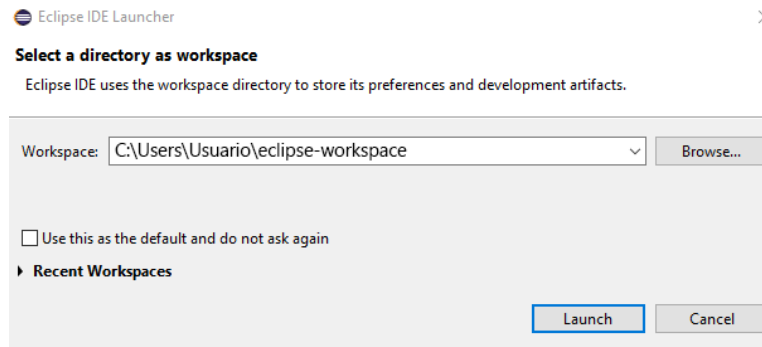


Imagen 12. Seleccionar ubicación del espacio de trabajo.

Fuente: Desafío Latam.

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

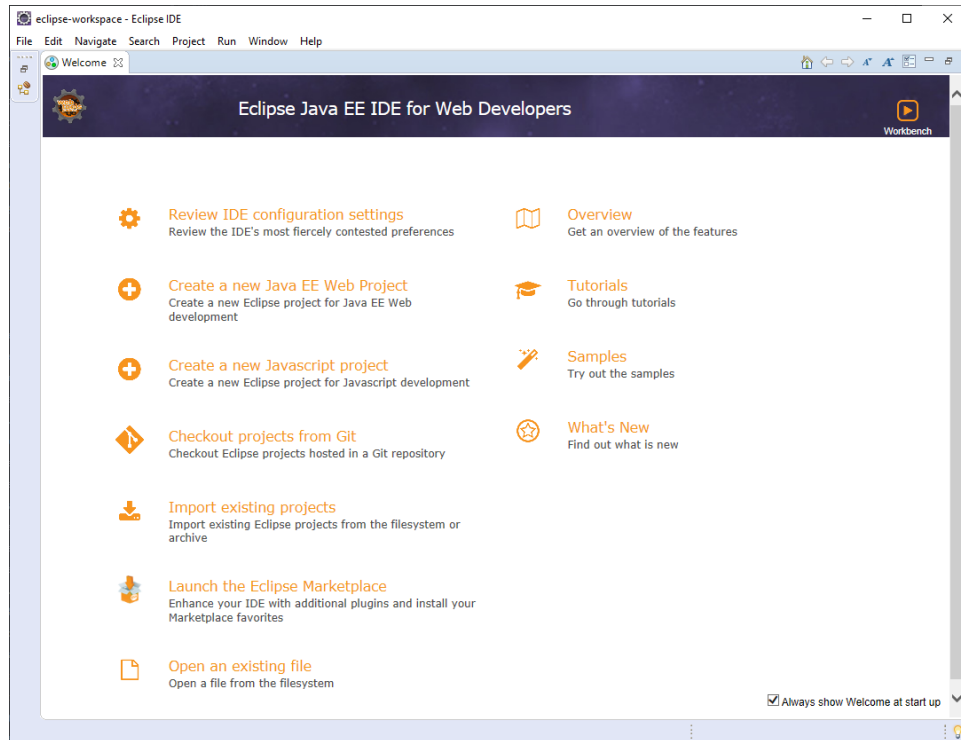


Imagen 13. Crear un nuevo proyecto Java.
Fuente: Desafío Latam.

Creando el primer programa en Java

Ya hemos aprendido algoritmos, específicamente pseudocódigo y diagramas de flujo. Pues, lo aprendido lo llevaremos a la práctica comenzando con un primer acercamiento al lenguaje Java.

Requerimientos previos

1. Versión de Java JRE y JDK 1.8 (Lectura complementaria: Instalando Java).
2. Eclipse IDE instalado.
3. Conocimiento en utilización de Java.

Ejercicio guiado: Nuestro primer proyecto

Para crear nuestro primer programa en Java, se deben seguir los siguientes pasos:

1. Crear un nuevo proyecto en nuestro IDE (Entorno de Desarrollo Integrado) de preferencia. Para nuestro caso será **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.

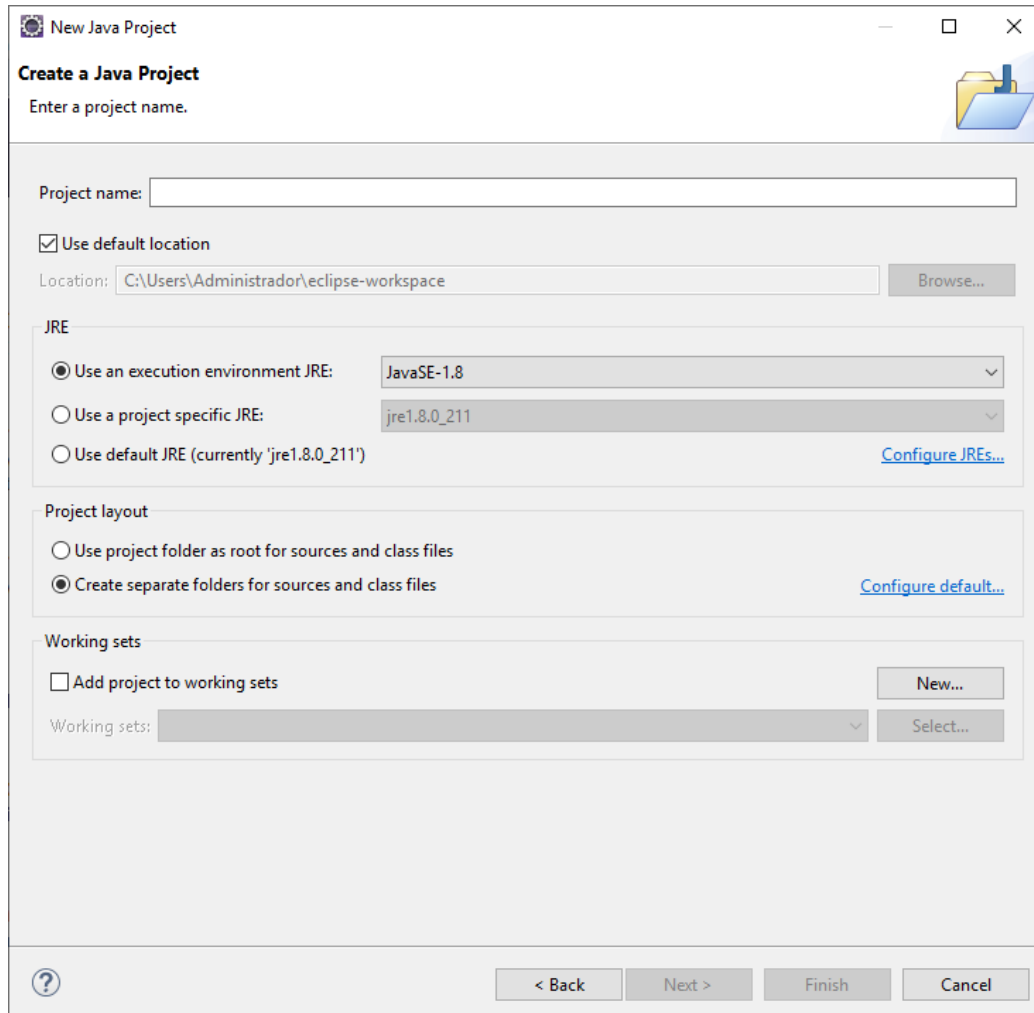


Imagen 14. Creando un nuevo proyecto en eclipse.
Fuente: Desafío Latam.

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

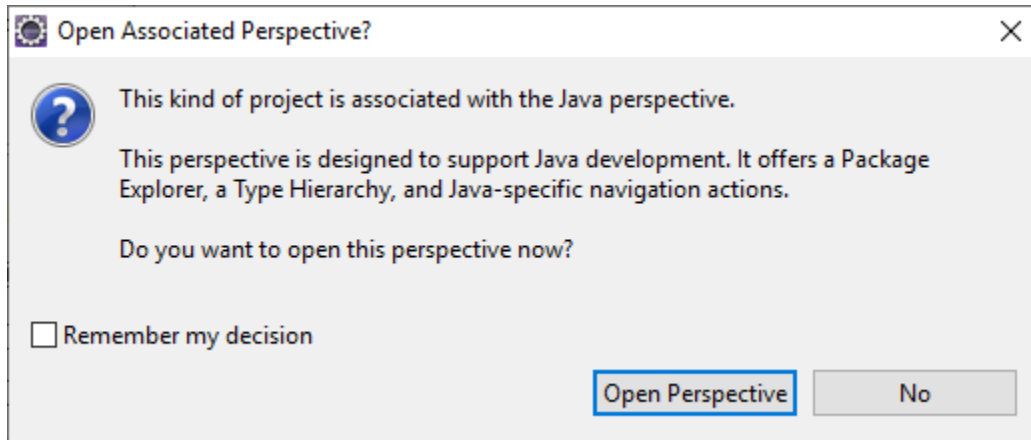


Imagen 15. Asociar proyecto con perspectiva de Java.
Fuente: Desafío Latam.

Revisando 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.

Continuemos

Ya creamos la base de nuestro proyecto, ahora debemos crear el fichero donde escribiremos el código como tal.

1. En la raíz del proyecto, haremos clic derecho sobre src -> New -> Package, al cual le colocaremos el nombre: cl.desafiolatam.

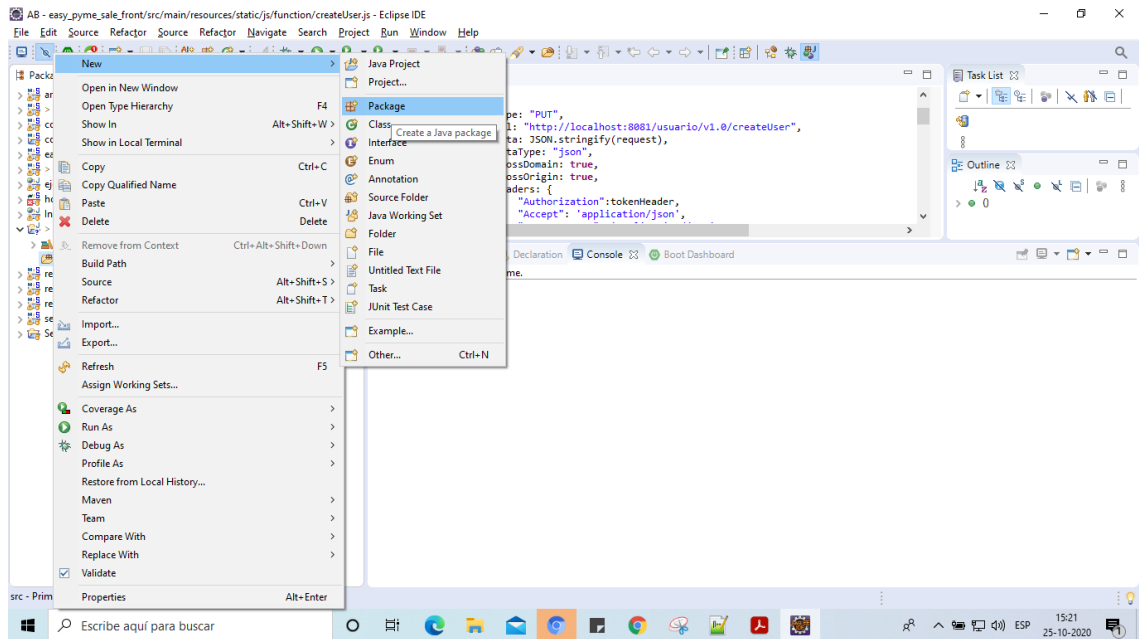


Imagen 16. Creación de un nuevo package (paquete) en eclipse.

Fuente: Desafío Latam.

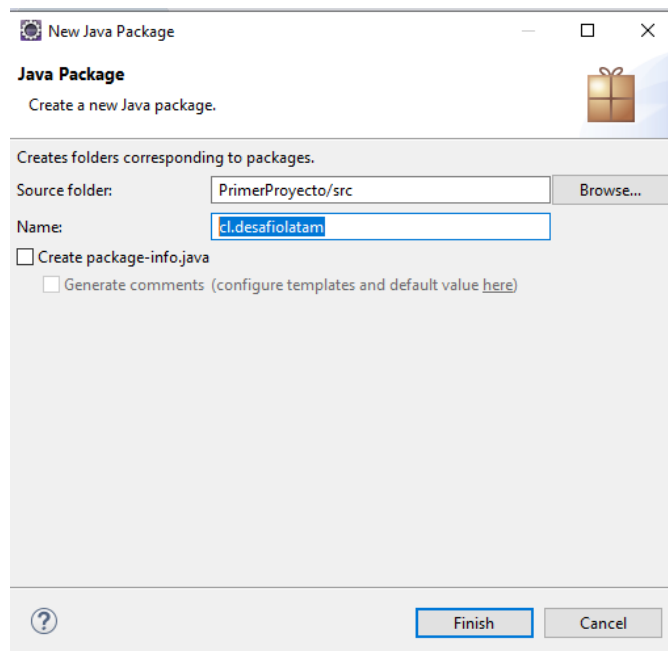


Imagen 17. Creación de un nuevo package (paquete) en eclipse.

Fuente: Desafío Latam.

2. 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 finalizar.

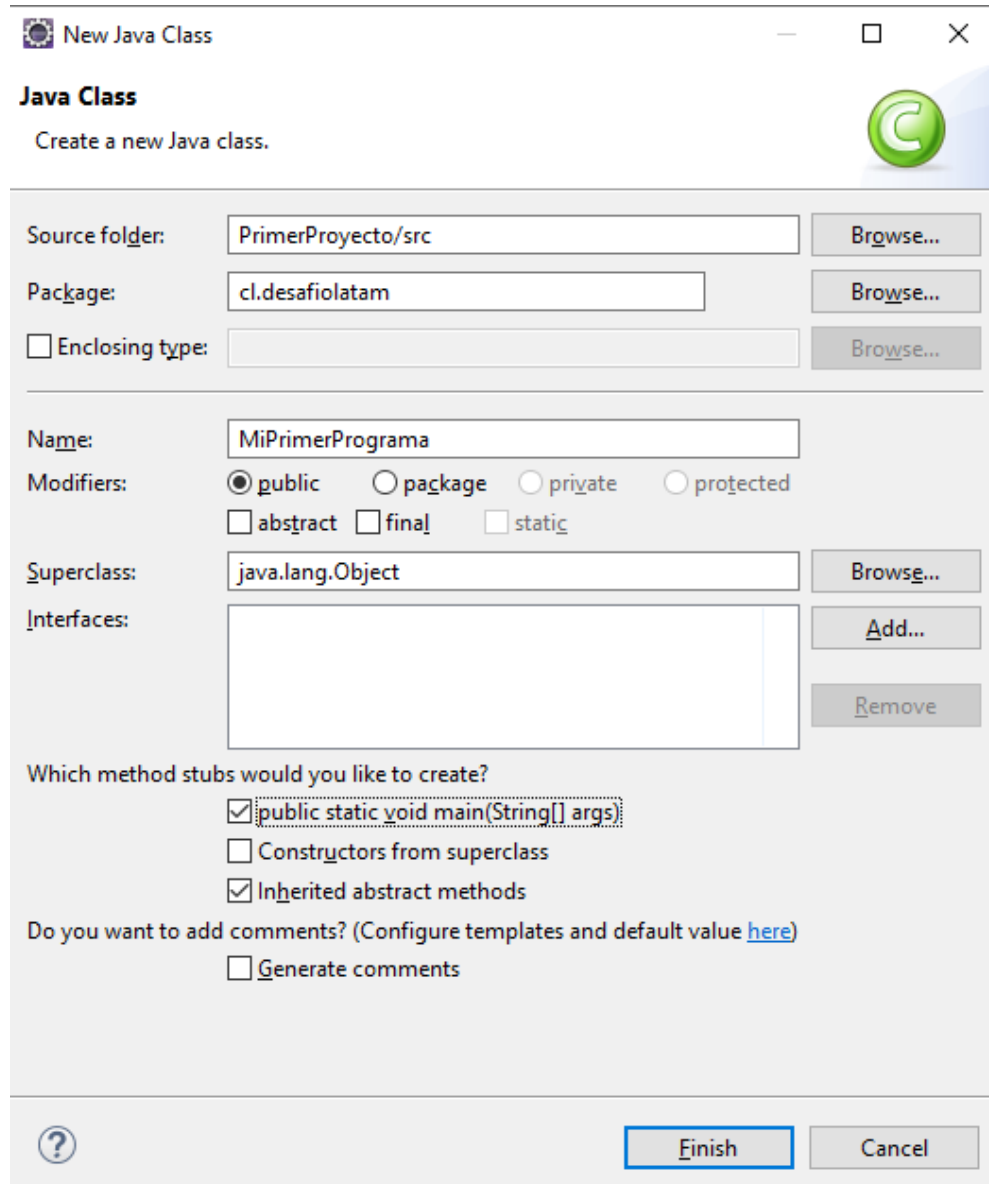


Imagen 18. Creación de clase main (Clase principal) en eclipse.
Fuente: Desafío Latam.

¡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 necesitamos:

```
package cl.desafiolatam;

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

Ahora para ejecutarlo, hacemos 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.

Al final del mensaje se colocó `\n` que corresponde a un salto de línea. Más adelante profundizaremos más en cuanto a los formatos de `printf`.

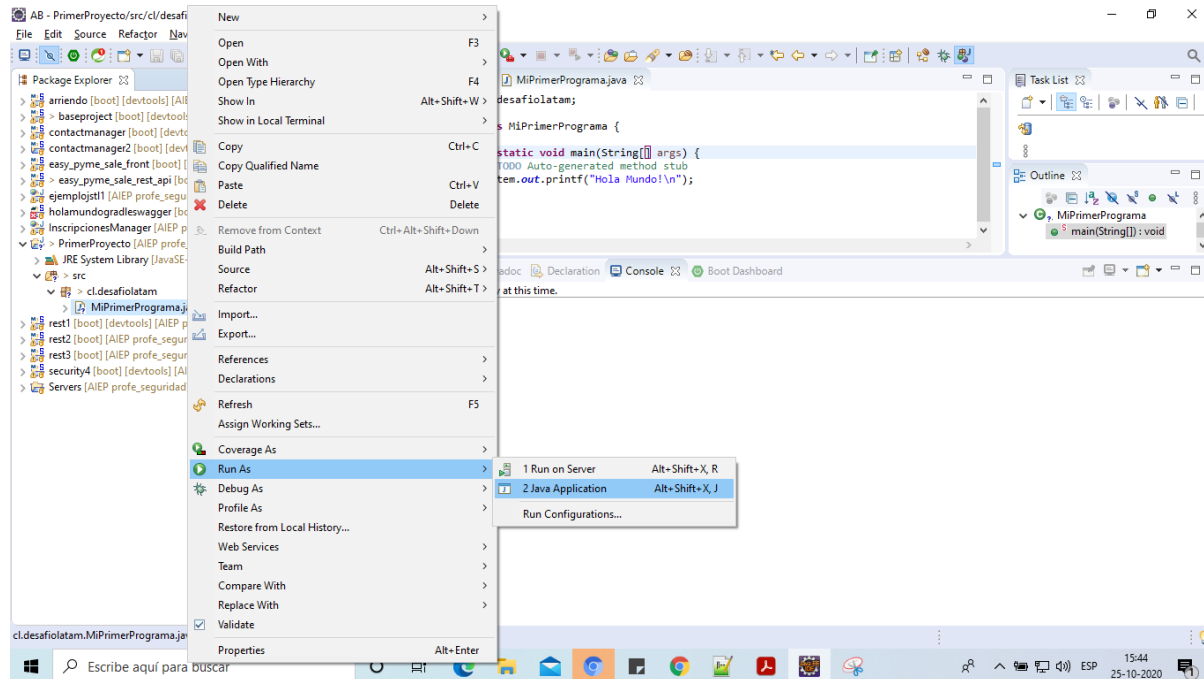


Imagen 19. Ejecutar programa Java en Eclipse IDE.
Fuente: Desafío Latam.

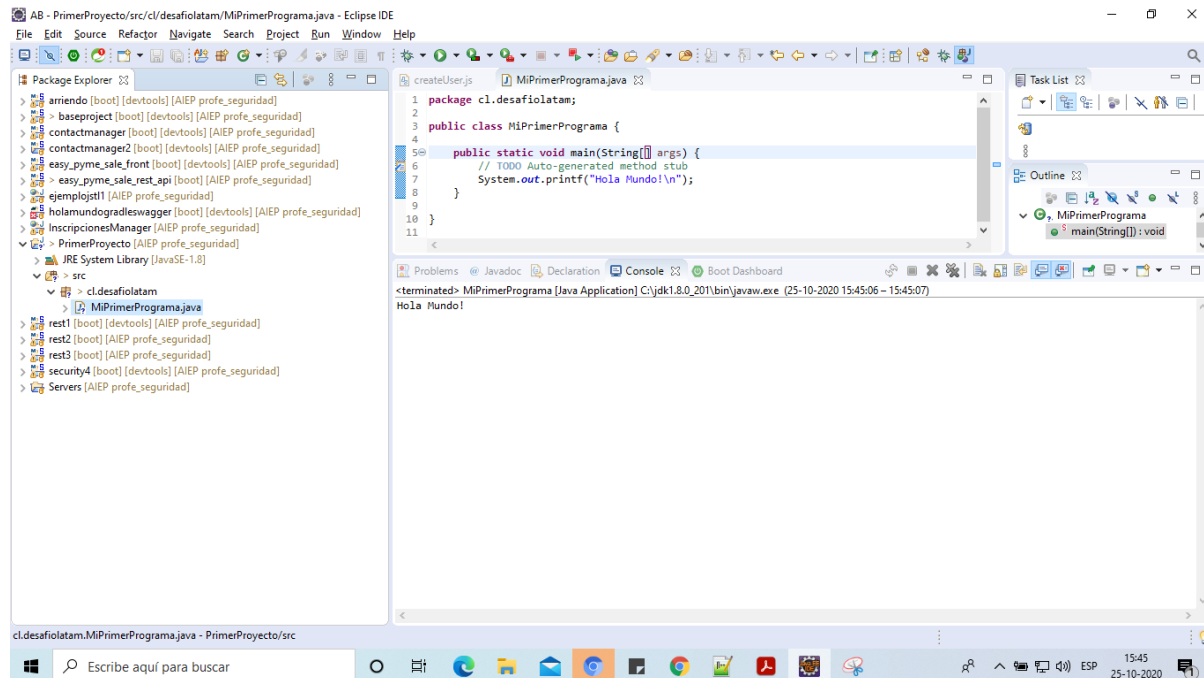


Imagen 20. Ejecución de nuestro primer programa en Java Hola Mundo!.
Fuente: Desafío Latam.

Java es un lenguaje que se debe aprender como si fuera cualquier otro lenguaje, por ejemplo, el Inglés, Francés, etc. Al ser un “lenguaje” tiene ciertas formas gramaticales que debemos aprender en base a la práctica constante. Se recomienda leer la [documentación web complementaria](#), específicamente los siguientes apartados:

- Introducción a Java.
- Conceptos básicos en Java.
- Operadores Java.
- Sentencias de control.

Resumen

Para crear código en Java, debemos crear nuestra clase principal (más adelante profundizaremos en qué es una clase), la cual contendrá la función principal llamada main.

Cada clase que eventualmente creemos debe estar contenida dentro de un paquete (que vendría a ser como una carpeta de Java).

Todo el código que vayamos a escribir de ahora en adelante siempre estará dentro del main, que es la función que llamará al ejecutar el programa.

Observación con el 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 seguirla.

Algoritmos, diagramas de flujos 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, estas son las variables, operadores y estructuras de control (condiciones). Por ejemplo, se requiere crear un programa que calcule la raíz cuadrada de un número:

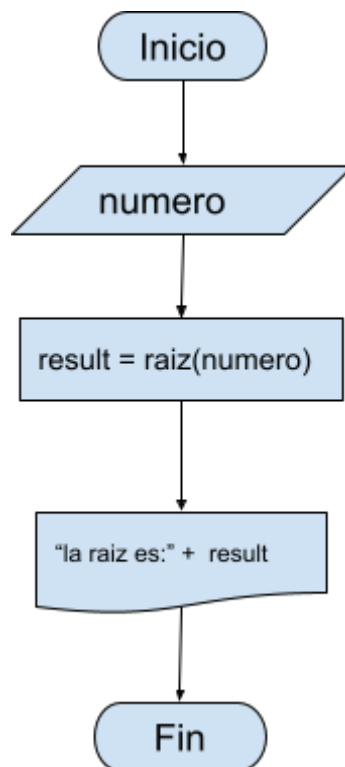


Imagen 21. Diagrama de flujo del cálculo de la raíz cuadrada de un número.

Fuente: Desafío Latam.

En el diagrama de flujo expuesto en la imagen 16, nótese que 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 representar la implementación en un lenguaje en particular. Además notar el símbolo para impresión “la raíz es: + result”. Este es el símbolo que representa una impresión por pantalla o bien en papel.

La implementación del flujo anterior en Java sería la siguiente:

```
package cl.desafiolatam;

import java.util.Scanner;

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

La diferencia al traspasar el diagrama de flujo al código en lenguaje Java es precisamente el lenguaje. Debemos saber, por ejemplo, que debemos ocupar 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, etc.

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.

Elementos básicos de Java

Competencias

- 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.

Introducción

A continuación revisaremos los elementos básicos para empezar a codificar pequeñas piezas de software. Comenzaremos con los comentarios y, posteriormente, con los distintos tipos de variables que encontraremos en Java y cómo realizar operaciones con ellas.

Comentarios

Los comentarios son líneas de código que son ignoradas por el compilador, pero sirven para dar indicaciones y documentación a nuestro código.

```
// Esta línea es un comentario
// Los comentarios son ignorados
// Pueden ser una línea nueva
int a = 2 // 0 puede acompañar una línea de código existente
// a = 2 + 3 Si comentamos al principio todo la línea será ignorada
```

Comentarios en múltiples líneas

Existe otra forma de hacer comentarios, donde se puede abarcar una mayor cantidad de código, como por ejemplo cuando queremos comentar un método completo. Para esto utilizaremos `/*` al inicio y `*/` al final, donde todo lo que esté encerrado entre esos símbolos quedará comentado.

```
/* Todo el código
escrito
en estas líneas
será ignorado
*/
int i; //esta variable ya no será ignorada
```

Los ejemplos que vayamos utilizando y los resultados a las operaciones las iremos dejando comentadas.

Introducción a variables

- Una variable en Java es un identificador que representa una palabra que contiene información.
- El tipo de información almacenado en la variable, solo puede ser del tipo con que se declaró la variable.
- Se denominan variables ya que el contenido que almacenan puede variar.

Partes de una variable

Se compone de:

- Un nombre o identificador.
- Un valor.
- Tipo de dato.

Por ejemplo, podemos tener las siguientes variables:

Declaración	Identificador	Tipo	Valor
int i;	i	Entero	no asignado
int b = 2;	b	Entero	2
String s;	s	Referencia a String	no asignado
boolean a = false;	a	Booleano	false

Tabla 1. Partes de una variable.
Fuente: Desafío Latam.

Tipos de datos primitivos

Los tipos de datos primitivos almacenan directamente un valor que siempre va a pertenecer al rango de ese tipo. Acá podemos encontrar los tipos de dato como: int, short, float, boolean, double, char, byte, entre otros.

- **int:** Los números enteros son números sin decimales que pueden ser positivos o negativos. En inglés se les denomina Integers.

```
int a = 2;
```

- **float:** Los números flotantes son números que pueden tener hasta 7 dígitos decimales:

```
float a = 3.5f;
```

- **double:** Corresponde a un dato "float x2", son números que pueden tener hasta 15 dígitos decimales:

```
double a = 3.5d;
```

- **boolean:** El tipo de dato boolean almacena únicamente dos valores, verdadero o falso:

```
boolean activo = true;  
boolean alto = false;
```

- **char:** El tipo de dato char representa un único carácter y se escribe entre comillas simples ' ' .

```
char valor = 'c';  
char otroValor = 'k';
```


Referencias a objetos

Otro tipo de variables son las de referencias a objetos que profundizaremos más adelante cuando veamos los conceptos de clases y objetos.

De momento necesitamos saber que el tipo de dato String es una referencia a un objeto, y con ella podremos realizar operaciones que con las variables de tipo primitiva no podremos.

String

El String básicamente es una cadena de caracteres, en la que podemos almacenar ya sea una palabra, frase o texto. Todo String debe ser escrito entre comillas " " .

```
"Esto es un String"  
"hola"  
"a"
```

Todos los ejemplos son distintos tipos de Strings.

Creando un String

Cuando creamos el Hola Mundo! en el ejemplo anterior, escribimos:

```
System.out.println("Hola Mundo!");
```

Ahí se creó implícitamente un String.

También se pudo haber creado una variable de tipo String y asignarle el valor:

```
String cadena = "Hola Mundo!";
```

Y como alternativa también se puede crear como:

```
String cadena = new String("El primer programa");
```

Si queremos crear un String vacío (o nulo) existen estas alternativas:

```
//Considerar el vacío como un dato vacío  
String cadena = ""; //Es un String vacío  
String cadena = new String(); //Crea una referencia a un objeto  
//null o nulo no es un dato, es un prospecto a ser un string  
String cadena = null;  
//Después de la línea anterior, se debe crear el objeto  
cadena = new String();  
/*Nótese que no se coloca la palabra reservada String, debido a que la  
variable objeto cadena ya se declaró como String en la línea anterior */
```

Declaración de un String vacío

Un string nulo es aquél que no contiene caracteres, pero es un objeto de la clase String .Sin embargo, al escribir:

```
String cadena;
```

Está declarando un objeto cadena de la clase String , pero aún no se ha creado ningún objeto de esta clase.

Esto quiere decir que para usar la variable `cadena`, debemos usar una de las alternativas antes mencionadas.

Operando con variables

Con los diferentes tipos de variables podemos realizar distintos tipos de operaciones. Ahora trabajaremos con los números enteros `int` y las cadenas de caracteres `String`.

Sumas y restas

Podemos operar con variables de tipo entero, como si estuviéramos operando con el número mismo, por ejemplo:

```
int a = 4;  
int b = 1;  
System.out.println(a+3); //7  
System.out.println(b-a); //-3
```

Acá usamos otro método de `System.out.println("variable")`, donde muestra el resultado en una línea y automáticamente hace el salto de línea, sin tener que usar `\n`.

Multiplicaciones y divisiones

```
int a = 10;  
int b = 3;  
System.out.println(a*3); //30  
System.out.println(a/b); //3
```

Observamos que al dividir $10/3$ el resultado que nos entrega es 3, ya que estamos trabajando con números enteros y no flotantes.

Sobreescribiendo variables

Podemos reemplazar el valor de una variable realizando una nueva asignación, pero teniendo en cuenta el tipo de dato de dicha variable.

```
int a = 10;  
a = 3;  
System.out.println(a); //3
```

Sumando enteros y Strings

Veamos qué pasa si tratamos de sumar dos variables de tipos distintas:

```
int a = 3;  
String b = "dos";  
System.out.println(a+b); //3dos
```

Pero si tratamos de guardar la suma en otra variables:

```
int a = 3;  
String b = "dos";  
int c = a+b;  
System.out.println(a+b); //3
```

En la consola aparece que no se puede convertir un String a int, ya que la lógica dice que un número puede ser un String, sin embargo, un String no puede ser un número, ya que este no asegura que sea un String con un dato numérico, si no que puede ser cualquier palabra. Más adelante veremos cómo realizar esta operación.

Modificando una variable existente

También podemos modificar el valor de una variable operando sobre ella misma, este tipo de operación es muy utilizada:

```
int a = 4;  
a = a + 1;  
System.out.println(a); //5
```

Variables de tipo String

¿Qué obtendremos al sumar dos Strings?

```
String a = "Hola";  
String b = " Mundo";  
System.out.println(a+b);  
//Hola Mundo
```

En programación la acción de “sumar” dos o más Strings se conoce como **concatenación**.

Creando un String a partir de variables

Podemos generar Strings a partir de otros Strings y variables usando especificadores y el método `format` de String.

```
int edad = 34;  
String nombre = "William";  
String salida = String.format("%s tiene %d años.", nombre, edad);  
System.out.println(salida);  
//William tiene 34 años.
```

Donde `%s` indica que el tipo de dato que tiene la primera variable (en este caso es de tipo String), `%d` indica que el tipo de dato es de tipo entero para la variable edad.

Otros especificadores para el formato son los siguientes:

```
// int %d
// char %c
// float %f
// String %s
```

Aquí podemos entender mejor cómo usar printf, que se comporta de la misma manera que String.format, donde primero se indica el formato que tendrá el String y luego las variables a utilizar.

Buscando un String

Aquí tenemos dos casos para encontrar una subcadena dentro de un String:

- **substring(int startIndex):** retorna un nuevo String que contiene el String desde el índice indicado.
- **substring(int startIndex, int endIndex):** retorna un nuevo String que contiene el Strings desde el índice indicado hasta el índice final exclusivo.

```
String s="Paralelepipedo";
System.out.printf("%s\n",s.substring(4)); // lelepipedo
System.out.printf("%s\n",s.substring(0,4));// Para
```

Información del String

Podemos ver la longitud de un String

```
String cadena = "Mi primer programa";  
int longitud = cadena.length(); // 18 - considera los espacios  
System.out.println(longitud);
```

Si comienza con un determinado sufijo

```
String cadena = "Mi primer programa";  
// Devuelve true si comienza con el sufijo especificado  
boolean resultado = cadena.startsWith("Mi");  
System.out.println(resultado);
```

En este ejemplo la variable resultado tomará el valor **true**.

Si se quiere obtener la posición de la primera ocurrencia de la letra **'p'**

```
//Desde cero, la p esta en la posición 16  
String cadena = "que clavo clavo pepito?";  
int pos = cadena.indexOf('p'); // 16 - se comienza a contar desde cero  
System.out.println(pos);
```

En caso de no existir, retornará el valor -1.

Constantes

Una constante sirve para almacenar un valor que no va a cambiar.

Para definir una constante se escribe de la siguiente manera:

```
final int DIAS_SEMANA = 7;  
final int MAX_ITERACIONES = 10;
```

Añadiendo antes del tipo de dato la palabra “final”, lo que hará que estos valores sean fijos y no sean modificables durante la ejecución del programa.

Convención sobre las constantes

Una convención es un estándar implantado como buenas prácticas en la programación. Dicho esto, para definir el nombre de una constante, esta debiese ser escrita completamente con mayúsculas, y si es una palabra compuesta, es decir más de dos palabras, cada palabra debe ir separada por un guión.

Entrada de datos

Nuestro programa podría necesitar interactuar con el usuario, ya sea para seleccionar una opción de un menú o algún valor sobre el cual el programa va a operar.

Para realizar dicha acción utilizaremos la clase Scanner que permitirá acceder a lo que vayamos ingresando por teclado.

```
Scanner sc = new Scanner(System.in);  
String cadena = sc.nextLine()
```

Pero antes, al inicio del fichero debemos agregar `import java.util.Scanner;`

```
package testeandoTiposDeDatos;  
import java.util.Scanner;  
public class testeandoTiposDeDatos{  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in); //Se crea el objeto Scanner  
        String i = sc.nextLine(); //Se lee una línea  
        System.out.println(i);  
    }  
}
```

Con esto obtendremos la línea ingresada como String en la consola, o bien, podemos obtener el próximo Entero (`nextInt()`), Flotante (`nextFloat()`), y más.

Ventajas del IDE Eclipse

Cuando no sabemos un nombre de un método que queremos utilizar, al escribir el punto luego de `sc`, aparecerá una lista de sugerencias con las distintas acciones que podemos escribir.

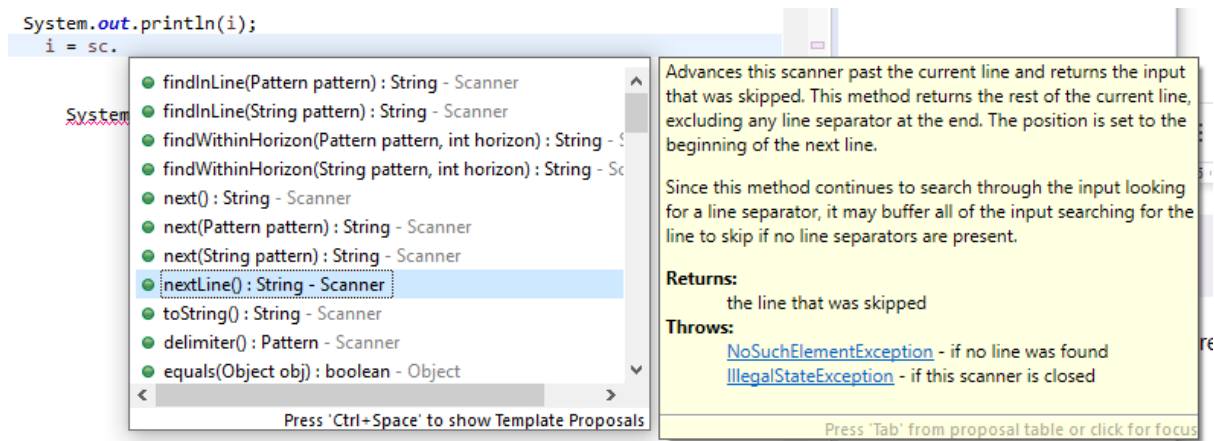


Imagen 22. Mostrar sugerencias de métodos en Eclipse IDE.

Fuente: Eclipse IDE.

Transformación de datos

Si queremos pasar un numero flotante a entero, por ejemplo, debemos hacer:

```
float a = 8.61f;
int b;
b = (int)a;
System.out.println(b); //El resultado será 8, es decir la variable a sin los
decimales
```

Así le estamos diciendo a la variable a de tipo flotante que tome solo la parte entera de ella.

En cambio si queremos transformar un número a String, debemos hacer lo siguiente:

```
int number = -782;
String numeroAString = String.valueOf(number);
System.out.println(numeroAString); // El resultado será "-782"
String numeroAString2 = String.valueOf(-782);
System.out.println(numeroAString2); // El resultado será "-782"
```

Y si queremos transformar un String a entero tendremos que utilizar la clase Integer, no el tipo de dato primitivo de entero para realizar esta acción.

```
String a = "45";
//La variable String debe ser numérica o si no habrá un error
int numero = Integer.parseInt(a);
```

Ejercicio guiado: Utilizando String y formato

Queremos escribir texto para el destinatario de una encomienda. Para ello debemos pedir al usuario los distintos campos que se requerirán:

```
Scanner sc = new Scanner(System.in);
String nombre = sc.nextLine(); //Carmen
String apellido = sc.nextLine(); //Silva
String direccion = sc.nextLine(); //Los aguiluchos
int numeroDireccion = sc.nextInt(); //43
String ciudad = sc.nextLine(); //Concepción
int telefono = sc.nextInt(); //562264895;
```

Para crear el formato de la etiqueta podemos hacerlo de varias formas:

```
String etiqueta = String.format("DE:%s %s\nDirección: %s %d\nCiudad: %s\nContacto:%d\n", nombre, apellido, direccion, numeroDireccion, ciudad, telefono);
System.out.println(etiqueta);
```

Recordemos que si tenemos dos String podemos concatenarlos usando el operador +:

```
String etiqueta2 = String.format(
    "DE:%s %s\n"
    + "Dirección: %s %d\n"
    + "Ciudad: %s\n"
    + "Contacto:%d\n",
    nombre, apellido, direccion, numeroDireccion, ciudad, telefono);
```

O bien, escribirlo directamente en `System.out.printf()`;

```
System.out.printf(
    "DE:%s %s\n"
    + "Dirección: %s %d\n"
    + "Ciudad: %s\n"
    + "Contacto:%d\n",
    nombre, apellido, direccion, numeroDireccion, ciudad, telefono);
```

En los tres casos obtendremos el mismo resultado:

```
DE:Carmen Silva
Dirección: Los aguiluchos 43
Ciudad: Concepción
Contacto:562264895
```

Importante

Como podemos ver, todos los tipos de datos primitivos se escriben con minúscula, en cambio los de referencia a objetos, con mayúscula, porque son referencias a clases.

Ejercicio propuesto (2)

Siguiendo la lógica del ejercicio propuesto (1), prepararemos una hoja de calificaciones para el estudiante, la cual servirá como base para calcular el promedio final utilizando lo aprendido en cuanto a formato.

Requerimientos

Crear y mostrar en pantalla, una hoja de calificaciones que tendrá la siguiente información:

- Nombre completo del estudiante.
- Asignatura.
- Nombre del Docente.
- Nota 1.
- Nota 2.
- Nota 3.

Operaciones Aritméticas

Competencias

- Hacer uso correcto de los operadores aritméticos de tal manera que se puedan implementar en un código Java para realizar cálculos matemáticos simples y complejos.

Introducción

Las operaciones aritméticas no son nada más que la implementación de cálculos matemáticos en lenguaje Java. Esto es como traducir fórmulas matemáticas al lenguaje en sí.

A continuación aprenderemos a construir aplicaciones del tipo de calculadora, donde el usuario ingresa valores y le entregamos resultados.

Motivación

¿Por qué debemos aprender operaciones matemáticas si estamos interesados en crear aplicaciones web o videojuegos?

Los operadores aritméticos se ocupan todo el tiempo ya sea para calcular el total de un carro de compras o cambiar la posición de un personaje en un videojuego.

En Java existen herramientas que nos permiten, entre otras cosas, sumar, restar, asignar valores, comparar, entre otros, gracias a los operadores.

Operadores aritméticos

Las operaciones aritméticas nos permiten realizar operaciones matemáticas sobre los números:

Operador	Nombre	Ejemplo	Resultado
+	suma	2+3	5
-	resta	2-3	-1
*	multiplicación	2*4	8
/	división	12/3	4
%	módulo o resto	5/2	1

Tabla 2. Operadores aritméticos.

Fuente: Desafío Latam.

Operaciones con variables

El proceso es exactamente igual si guardamos los valores en variables

```
int a = 2;  
int b = 3;  
System.out.println(a+b);
```


Creando una calculadora

Esto nos permite que el usuario ingrese los valores, y nosotros operar sobre ellos.

```
Scanner sc = new Scanner(System.in);  
int a = sc.nextInt(); //2  
int b = sc.nextInt(); //3  
System.out.printf("a + b es igual a %d \n", a+b);  
System.out.printf("a * b es igual a %d \n", a*b);  
//a + b es igual a 5  
//a * b es igual a 6
```

Precedencia de operadores

Un concepto muy importante que debemos conocer es el de precedencia, es decir, saber en qué orden se realiza un grupo de operaciones.

En el lenguaje Java y todos los lenguajes, la precedencia sigue las mismas reglas aritméticas de las matemáticas convencionales, es decir, primero los paréntesis con multiplicaciones o divisiones, etc. Por ejemplo:

```
10 - 5*2  
10 - 10  
0
```

Orden de las operaciones

Veamos una tabla simplificada de precedencia. Esta tabla está ordenada de mayor a menor prioridad, esto quiere decir que la operación de exponenciación tiene mayor precedencia que la suma.

Operador	Nombre
Math.pow(a,b)	Potencia o elevado a..
*, /, %	Multiplicación, división, resto o módulo
+, -	Suma, resta

Tabla 3. Precedencia de operadores.
Fuente: Desafío Latam.

Cuando dos operaciones tienen el mismo nivel de prioridad entonces se resuelven de izquierda a derecha.

Operaciones y paréntesis

Al igual que en matemáticas, los paréntesis cambian el orden en que preceden las operaciones. Dando prioridad a las operaciones que estén dentro de los paréntesis.

```
System.out.println((10 - 5)*2); // 10  
System.out.println(10-5*2); //0
```

¡Los paréntesis sí importan!

Operaciones con números enteros y decimales

Si dividimos números enteros nos encontraremos con una sorpresa.

```
System.out.println(5/3); // = 1
```

Esto es muy común en todos los lenguajes de programación, para obtener la respuesta que esperamos necesitamos ocupar otro tipo de dato, el float.

Float

El tipo de dato primitivo asociado a los números decimales que se llamaba float.

Enteros y float

La división entre entero y float, o float y entero da como resultado un float.

```
System.out.println(5.0f/3.0f);
```

O bien podríamos tener dos variables de tipo float.

```
float a = 5.0f;  
float b = 2.0f;  
System.out.println(a/b);
```

La división entre floats también es un float. En Java podemos transformar un entero a float usando cast, esto será especialmente útil cuando estemos trabajando con variables que contengan enteros.

```
int a = 1;  
int b = 2;  
/*a esto se le llama cast, es una transformación de un tipo de datos a otro  
tipo.*/  
System.out.println((float)a/b);
```

Ejercicio guiado: Calculando la velocidad de un automóvil

Calcular la velocidad de un auto en km/h teniendo la distancia y el tiempo, cabe destacar que puedes usar decimales para la entrada de datos.

Lo primero que debemos hacer es declarar las variables:

```
float distancia;  
float tiempo;
```

Después solicitamos la entrada de los datos:

```
Scanner sc = new Scanner(System.in);  
  
    System.out.println("Ingresa la distancia en km: ");  
    distancia = sc.nextFloat();  
  
    System.out.println("Ingresa el tiempo en horas: ");  
    tiempo = sc.nextFloat();
```

Creamos la variable para la realización del cálculo y su resultado:

```
// Calcular  
float velocidad = distancia / tiempo
```

Finalmente, mostramos el resultado por pantalla:

```
System.out.printf("La velocidad es: " +velocidad+ " km/h");
```

Ejercicio propuesto (3)

Continuando con el ejercicio propuesto (2), prepararemos una hoja de calificaciones para el estudiante, agregando como dato, el promedio final de las notas del estudiante.

Requerimientos

Crear una hoja de calificaciones que tendrá la siguiente información:

- Nombre completo del estudiante.
- Asignatura.
- Nombre del Docente.
- Nota 1.
- Nota 2.
- Nota 3
- Promedio de Notas

Casting en Java

Competencias

- Identificar los tipos de datos de una variable para comprender formas de transformar desde un tipo de dato a otro.
- Aplicar transformaciones de tipos de datos de una variable.

Introducción

Existen ocasiones en que, en nuestro algoritmo, necesitamos transformar desde un tipo de datos a otro, o bien, desde una clase objeto a otra clase objeto. Para esto usamos los casting o cast que nos permiten hacer esta transformación siempre y cuando se pueda.

Implementación de un casting

El casting es un procedimiento para transformar una variable primitiva de un tipo a otro, o transformar un objeto de una clase a otra clase siempre y cuando tengan relación entre sí.

Para realizar esto, se coloca el tipo de dato al cual queremos convertir antecediendo al elemento que deseamos su transformación

```
System.out.println((int) 4.5); // 4  
System.out.println((float) 4); // 4.0
```

Casting en String

Si consideramos que es un String, más allá de que un String define un tipo **“cadena de caracteres”**, podemos inferir que un String puede entonces ser un texto, número, float o cualquier cosa. A esto se le llama un dato **alfanumérico**. En base a lo anterior, entonces podemos inferir que no es posible transformar un String a un entero, a un float o a cualquier otro tipo de datos, ya que un String puede contener, por ejemplo “1.23P”, lo cual por tener una letra P ya no es un dato numérico.

No obstante a lo anterior si sabemos que el dato que contiene un String corresponde a algún otro tipo de dato conocido, se pueden hacer las transformaciones que deseemos. Por ejemplo:

“1.2” → Se puede convertir a un float o a un double, ya que 1.2 puede serlo.

“10” → Se puede convertir a un entero.

Las transformaciones de datos que tenemos en el ejemplo anterior se llaman **Parser o Parseo**. El Parser es un método que tienen todas las clases Java que corresponden a tipos primitivos. Estas clases son:

- Integer.
- Double.
- Float.

Cada una de estas clases contiene un método que transforma un String en su tipo de dato primitivo, por ejemplo, Integer en un int, Double en un double. El siguiente fragmento de código muestra los tres tipos de parse que tenemos para cada clase:

```
//Variables de tipo String
String stringFloat = "1.2";
String stringDouble = "2.4";
String stringInteger = "23";

//Variables numéricas
float datoFloat;
double datoDouble;
int datoInteger;

//Cada clase tiene su propio parse
datoFloat = Float.parseFloat(stringFloat);
datoDouble = Double.parseDouble(stringDouble);
datoInteger = Integer.parseInt(stringInteger);

System.out.println("Esto es un float: " + datoFloat);
System.out.println("Esto es un double: " + datoDouble);
System.out.println("Esto es un integer: " + datoInteger);
```

Ejercicio guiado: Cálculo de área rectángulo

Dado el siguiente esquema, calcular su área y perímetro.

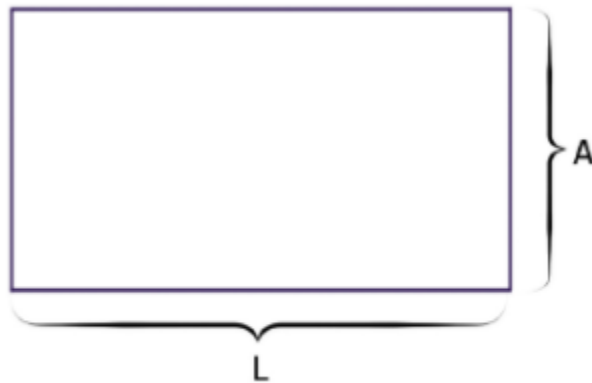


Imagen 23. Lados de un rectángulo.

Fuente: Desafío Latam.

Paso 1: Para resolver el problema, una buena técnica es identificar los pasos claves que debo realizar para llegar a la solución.

- Se necesita recibir dos datos numéricos en la entrada, correspondiente a cada lado del rectángulo.
- Cada dato debe ser mayor a cero, en el caso contrario dar al usuario un mensaje, por ejemplo, “El dato ingresado no es correcto. Ingrésele nuevamente”.
- En caso de que los datos sean correctos, calcular el área con la siguiente fórmula:

$$\text{área} = L * A$$

1. Creación del proyecto en eclipse

a. Ir a File -> New -> Java Project

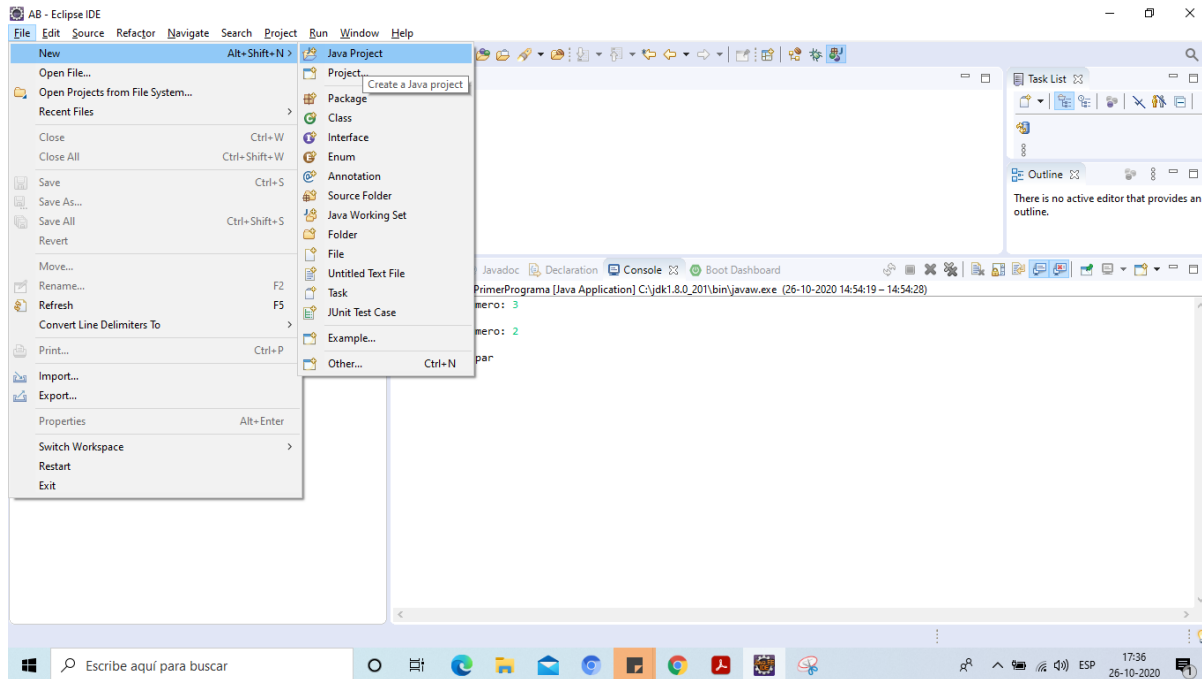
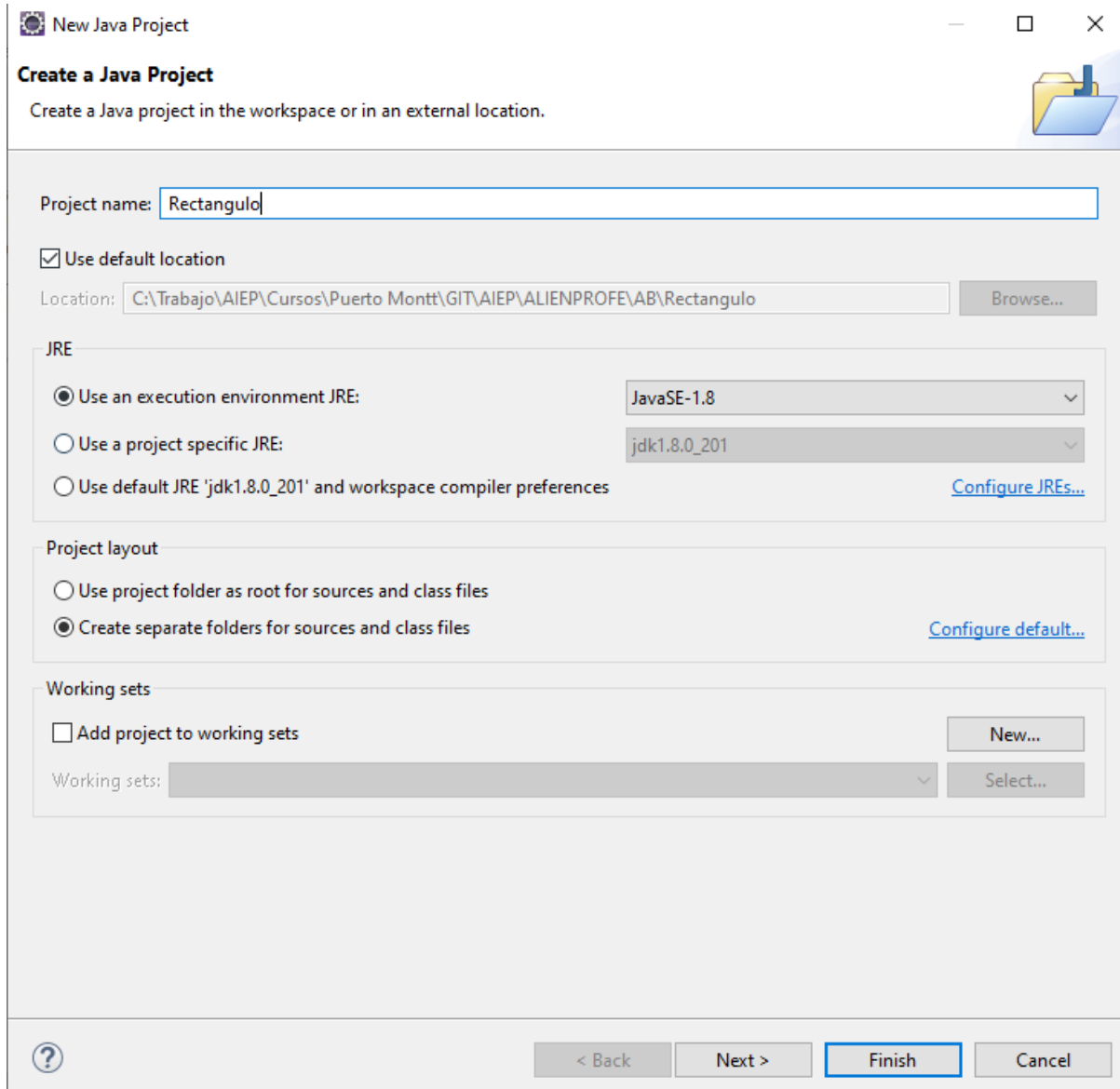


Imagen 24. Crear un nuevo proyecto en Eclipse IDE.
Fuente:Desafío Latam.

b. Crear el proyecto “Rectángulo”.



New Java Project

Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location

Location: [Browse...](#)

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE 'jdk1.8.0_201' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

[? < Back](#) [Next >](#) [Finish](#) [Cancel](#)

Imagen 25. Ingreso de datos del proyecto.
Fuente: Desafío Latam.

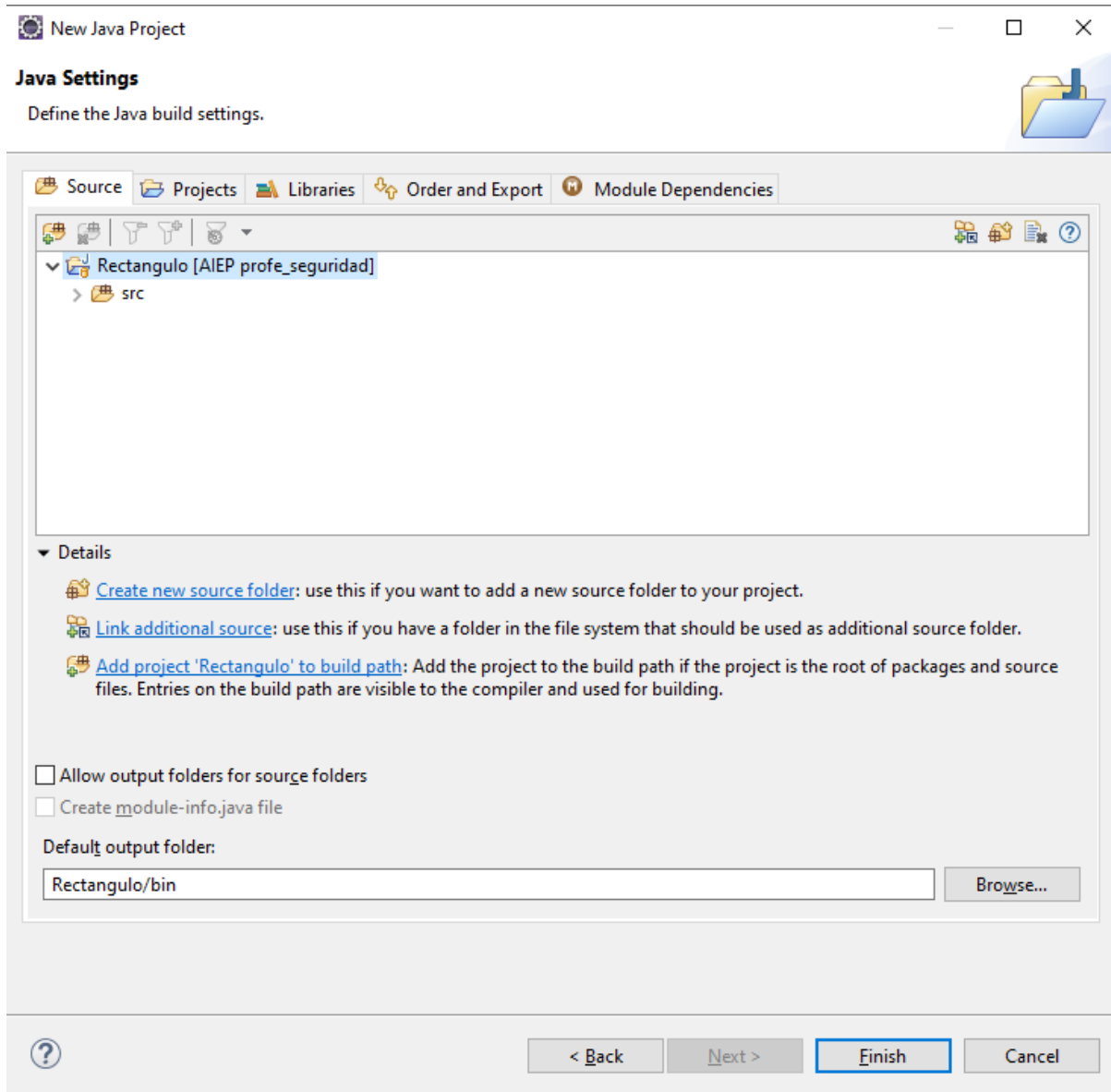


Imagen 26. Creación del proyecto. Se elige la estructura por defecto.
Fuente: Desafío Latam.

c. Crear el package cl.desafiolatam. clic derecho en carpeta src -> New -> Package

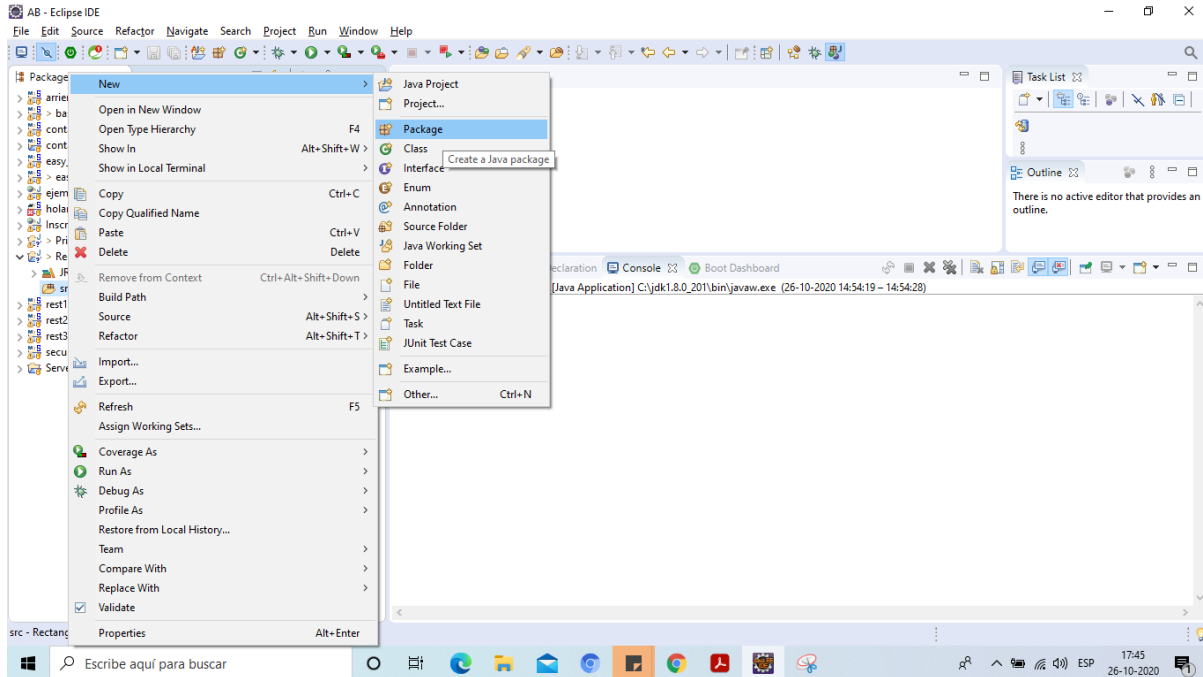


Imagen 27. Creación de un package o paquete de software.

Fuente: Desafío Latam.

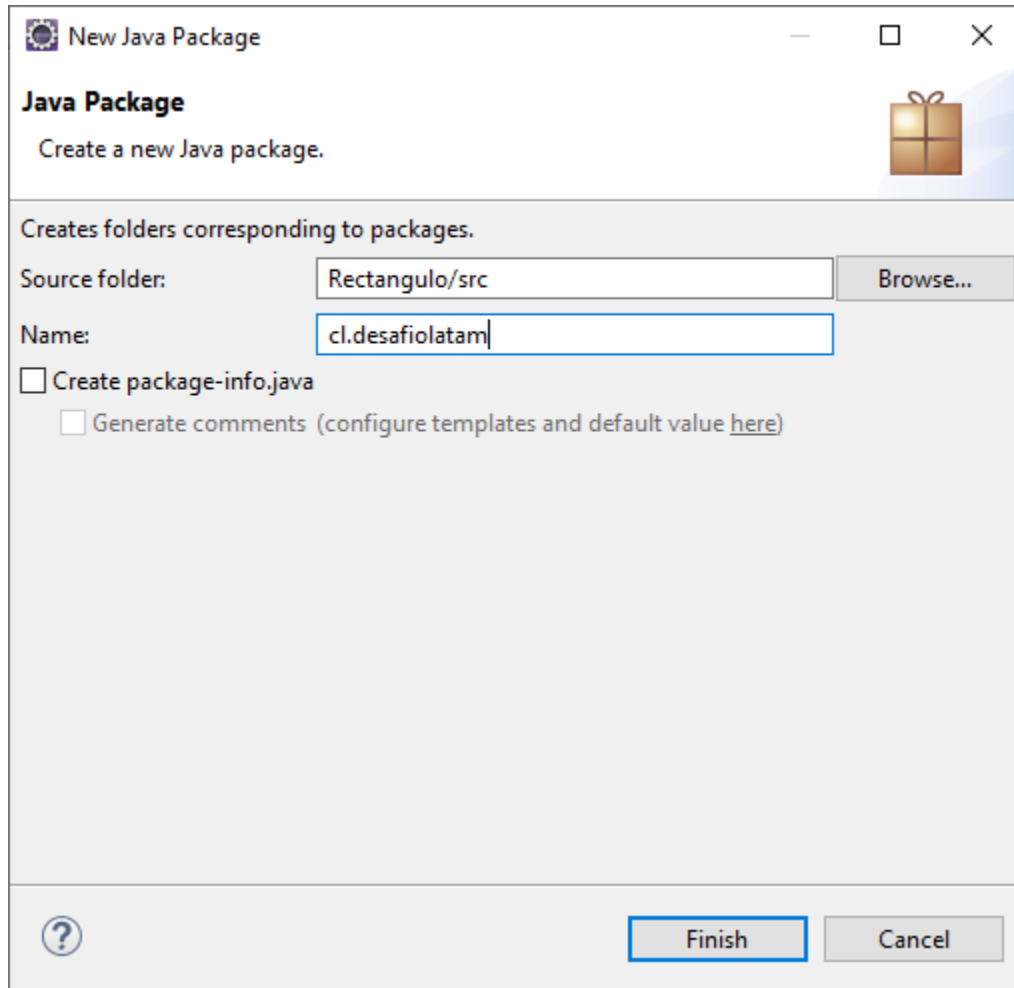


Imagen 28. Se ingresa el nombre del package y se finaliza.
Fuente: Desafío Latam.

- d. Crear la clase principal `AreaRectangulo.java`.
Clic derecho sobre el package → New → class. (Recordar hacer clic en public static void main)

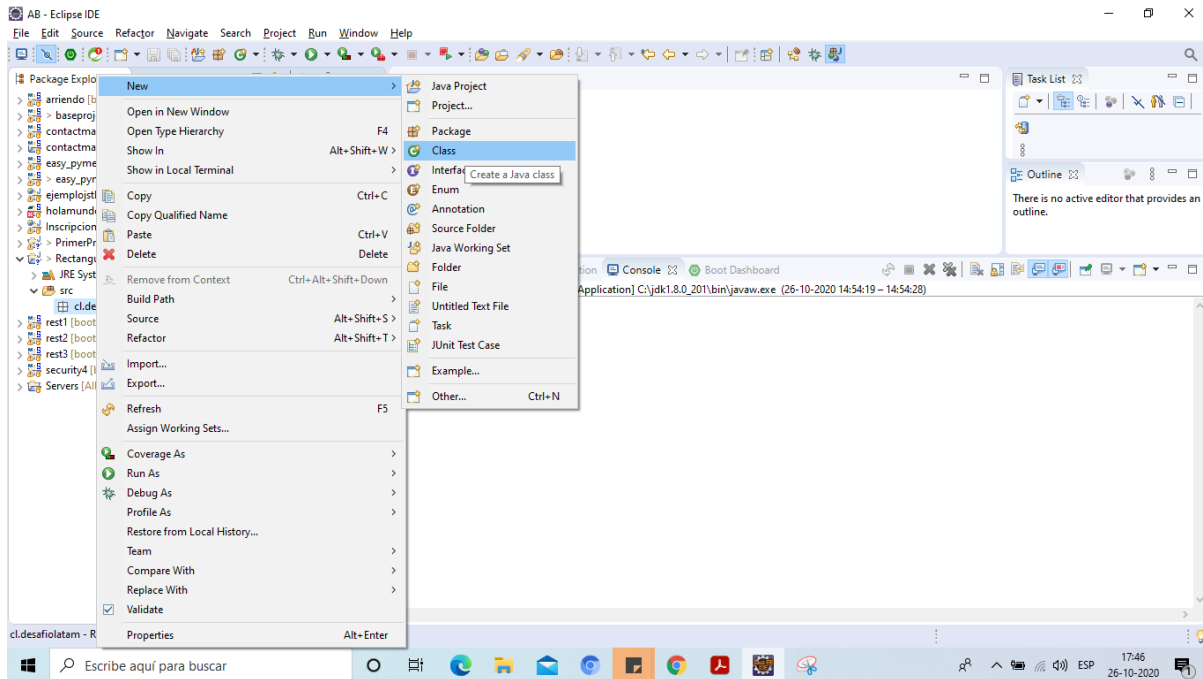


Imagen 29. Creación de una clase en eclipse IDE.
Fuente: Desafío Latam.

New Java Class

Java Class
Create a new Java class.

Source folder: Rectangulo/src [Browse...](#)

Package: cl.desafiolatam [Browse...](#)

☐ Enclosing type: [Browse...](#)

Name: AreaRectangulo

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object [Browse...](#)

Interfaces: [Add...](#) [Remove](#)

Which method stubs would you like to create?

☒ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

[?](#) [Finish](#) [Cancel](#)

Imagen 30. Ingresar nombre de la clase y seleccionar main.
Fuente: Desafío Latam.

2. Leer los datos: En la clase creada "AreaRectangulo" se deben leer los datos ingresados por el usuario. Considerar que los datos que se ingresen no pueden ser negativos ni tampoco igual a cero.

```
package cl.desafiolatam;
import java.util.Scanner;
public class AreaRectangulo {
    public static void main(String[] args) {
        double base = 0.0;
        double altura = 0.0;
        Scanner sc = new Scanner(System.in);
        do{
            System.out.printf("Ingrese la base: ");
            base = sc.nextDouble();
            if(base <= 0) {
                System.out.println("Dato inválido");
            }
        }while(base <= 0);

        do{
            System.out.printf("Ingrese la altura: ");
            altura = sc.nextDouble();
            if(altura <= 0) {
                System.out.println("Dato inválido");
            }
        }while(altura <= 0);
    }
}
```

3. Calcular Área

- a. Se agregan las siguientes líneas al algoritmo. La primera es la declaración de la variable que guardará el resultado del cálculo. La segunda, será la que realizará el cálculo correspondiente. La tercera, la que imprimirá el resultado por pantalla.

```
double area = 0.0;  
  
area = base * altura;  
  
System.out.println("El área del rectángulo es: " + area);
```

b. El código del algoritmo completo, quedaría como sigue:

```
public class AreaRectangulo {  
  
    public static void main(String[] args) {  
        double base = 0.0;  
        double altura = 0.0;  
        double area = 0.0;  
        Scanner sc = new Scanner(System.in);  
        do{  
            System.out.printf("Ingrese la base: ");  
            base = sc.nextDouble();  
            if(base <= 0) {  
                System.out.println("Dato inválido");  
            }  
        }while(base <= 0);  
  
        do{  
            System.out.printf("Ingrese la altura: ");  
            altura = sc.nextDouble();  
            if(altura <= 0) {  
                System.out.println("Dato inválido");  
            }  
        }while(altura <= 0);  
  
        area = base * altura;  
  
        System.out.println("El área del rectángulo es: " + area);  
    }  
}
```

Ejercicio propuesto (4)

La fórmula de Pitágoras nos permite calcular el largo de la hipotenusa de un triángulo rectángulo a partir de los largos de los catetos. Crearemos un programa donde el usuario introduzca los valores de ambos catetos y entreguemos como resultado el largo de la hipotenusa.

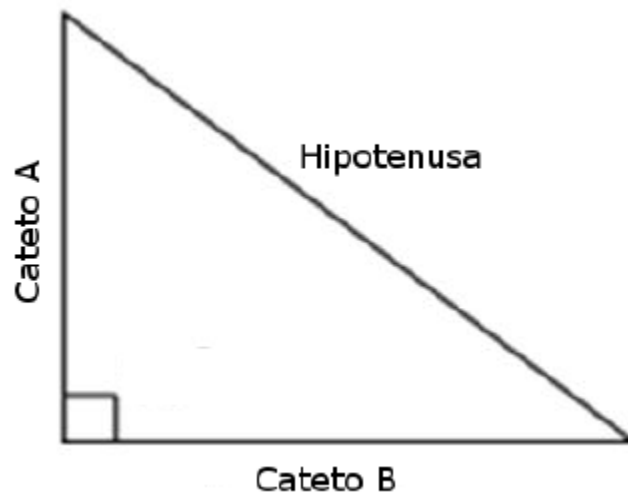


Imagen 31. Gráfica de un triángulo rectángulo.
Fuente: Desafío Latam.

La fórmula para calcular la hipotenusa es la siguiente:

$$hipotenusa = \sqrt{a^2 + b^2}$$

Solución ejercicio propuesto (1)

Requerimientos:

1. Ingresar 3 notas entre 1 y 7
2. Calcular el promedio de notas en base a la siguiente función:

$$\text{promedio} = \text{nota1} + \text{nota2} + \text{nota3} / \text{cantidad De Notas (3 en este caso)}$$

3. Si el promedio es **menor a 4**, mostrar por pantalla que el alumno está **reprobado**.
4. Si el promedio es **igual o mayor a 4**, mostrar por pantalla que el alumno está **aprobado**.

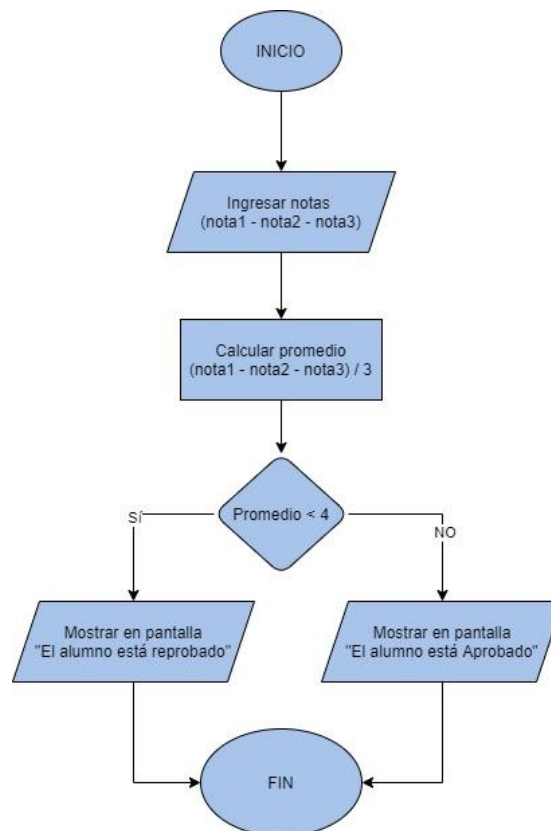


Imagen 32. Solución ejercicio propuesto (1).

Fuente: Desafío Latam.

Pseudocódigo

```
Algoritmo CalcularPromedio
  Leer nota1
  Leer nota2
  Leer nota 3

  promedio = (nota1 + nota2 + nota3) / 3

  si promedio es menor a 5 entonces
    Mostrar " El alumno está reprobado "
  si no
    Mostrar " El alumno está Aprobado "
FinCalcularPromedio
```

Solución ejercicio propuesto (2)

Para definir esta hoja de calificaciones debemos pedir al usuario que ingrese los datos que se requieren, luego de que el usuario ingrese estos datos, deben almacenarse en variables para ser mostrados en el orden que indica el enunciado.

```
package calificaciones;
import Java.util.Scanner;
public class Calificaciones{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String estudiante = sc.nextLine();
        String asignatura = sc.nextLine();
        String docente = sc.nextLine();
        int nota1 = sc.nextInt();
        int nota2 = sc.nextInt();
        int nota3 = sc.nextInt();
        System.out.printf(
            "Estudiante:%s %s\n"
            + "Asignatura: %s\n"
            + "Docente: %s\n"
            + "Nota 1: %d\n",
            + "Nota 2: %d\n",
            + "Nota 3: %d\n",
            estudiante, asignatura, docente, nota1, nota2, nota3);
    }
}
```


Solución ejercicio propuesto (3)

Para resolver este ejercicio integraremos el código del ejercicio propuesto (2).
Le agregamos la fórmula para calcular el promedio.

```
float promedio = (nota1 + nota2 + nota3) / 3;
```

Y mostraremos en pantalla todos los datos. Quedando finalmente de la siguiente manera:

```
package calificaciones;
import Java.util.Scanner;
public class Calificaciones{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String estudiante = sc.nextLine();
        String asignatura = sc.nextLine();
        String docente = sc.nextLine();
        int nota1 = sc.nextInt();
        int nota2 = sc.nextInt();
        int nota3 = sc.nextInt();

        float promedio = (nota1 + nota2 + nota3) / 3;

        System.out.printf(
            "Estudiante:%s %s\n"
            + "Asignatura: %s\n"
            + "Docente: %s\n"
            + "Nota 1: %d\n",
            + "Nota 2: %d\n",
            + "Nota 3: %d\n",
            + "Promedio: %d\n",
            estudiante, asignatura, docente, nota1, nota2, nota3, promedio);
    }
}
```

Solución ejercicio propuesto (4)

1. Se deben identificar los pasos a seguir para esto se podría hacer un diagrama de flujos

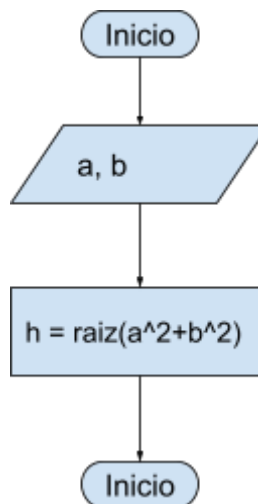


Imagen 33. Diagrama de flujo para el cálculo de la hipotenusa.

Fuente: Desafío Latam.

Nótese que para representar un elevado o una potencia se usa el operador $^$, sin embargo, en lenguaje Java utilizaremos el método pow de clase Math

2. Se debe crear el proyecto en eclipse, el package correspondiente y la clase Main, donde crearemos el siguiente algoritmo en lenguaje Java.

```
Scanner sc = new Scanner(System.in);
int a = sc.nextInt();
int b = sc.nextInt();
float h = (float)Math.sqrt(Math.pow(a, 2) + Math.pow(b, 2)); // retorna
tipo double
System.out.println(h);
```

3. El código completo de la clase quedaría como el siguiente:

```
package cl.desafiolatam;

import Java.util.Scanner;

public class Hipotenusa {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
        float h = (float)Math.sqrt(Math.pow(a, 2) + Math.pow(b, 2)); //
retorna tipo double
        System.out.println(h);
    }
}
```