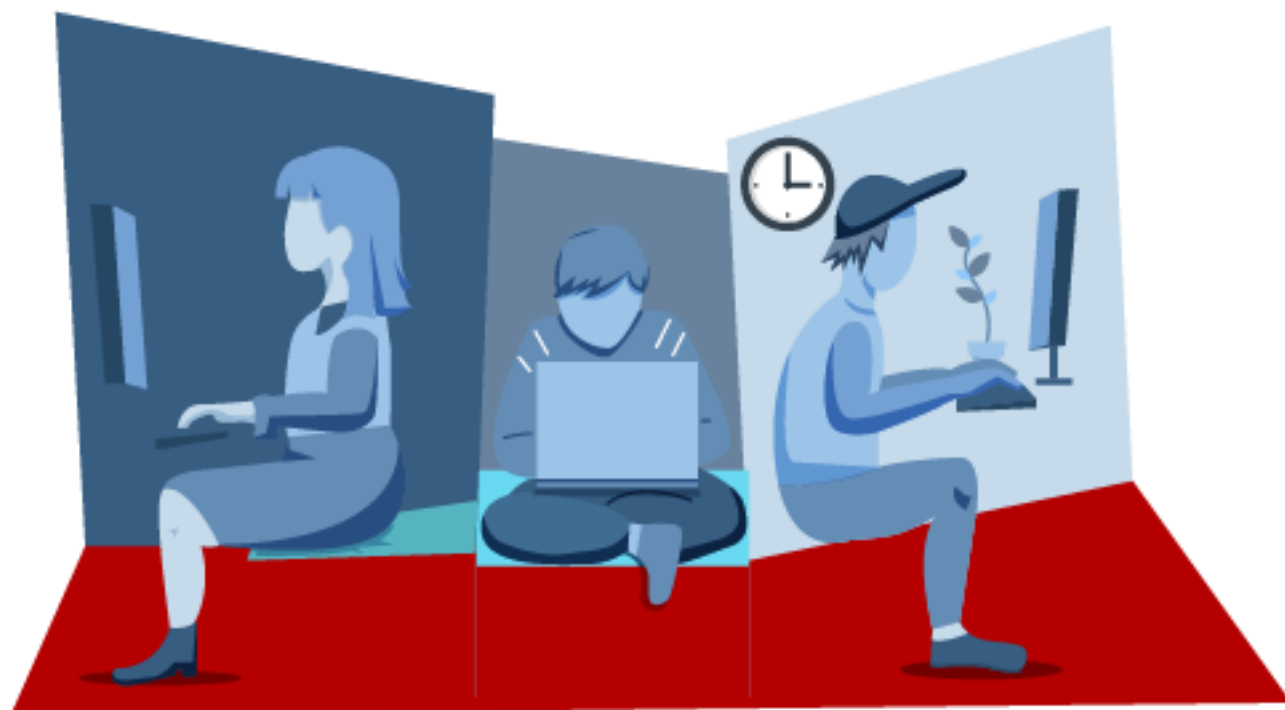




VehículoApp



**MÓDULO: PRO401-9524-225081-ONL-TALLER DE APLICACIONES MÓVILES**

**SEMANA: 3**

Docente: Iván Ayala

Estudiante: Karla Pesce, Jaime Codoceo y Sergio Molina

## Índice

<b><i>Introducción</i></b> .....	<b>3</b>
<b><i>Requerimientos</i></b> .....	<b>4</b>
Requerimientos funcionales (RF) .....	4
Requerimientos no funcionales (RNF) .....	4
<b><i>Desarrollo</i></b> .....	<b>5</b>
Entorno y prerequisites .....	5
Código Utilizado. ....	5
Proceso de compilación .....	7
Proceso de ejecución .....	8
Ejemplo de la ejecución.....	8
Documentación en GitHub .....	9
Historias de Usuario .....	11
Cronograma inicial: GitHub Project.....	12
<b><i>Conclusión</i></b> .....	<b>13</b>
<b><i>Bibliografía</i></b> .....	<b>14</b>

## Introducción

El desarrollo de aplicaciones en Java constituye una de las primeras aproximaciones al mundo de la programación estructurada y orientada a objetos. En esta actividad se busca comprender el proceso completo de creación, compilación y ejecución de un programa en Java sin utilizar un IDE, con el fin de fortalecer el dominio de la línea de comandos y afianzar la lógica básica.

El ejercicio práctico consiste en elaborar un programa sencillo que solicite al usuario datos específicos de un vehículo —marca, modelo, cilindrada, tipo de combustible y capacidad en pasajeros— utilizando la clase `Scanner` para la entrada de datos y variables de tipo `String` e `int` para su almacenamiento. Posteriormente, el programa debe mostrar en consola los valores ingresados, reforzando el ciclo entrada–proceso–salida fundamental en cualquier aplicación.

Más allá del código, la actividad promueve la documentación del proceso completo en un repositorio de GitHub, incluyendo requerimientos funcionales y no funcionales, historias de usuario y un cronograma de trabajo mediante GitHub Projects. Finalmente, se solicita la entrega de un informe que evidencie la práctica realizada, explique el proceso de compilación por consola y presente reflexiones sobre el aprendizaje adquirido.

## Requerimientos

En función de las instrucciones entregadas en la actividad, se definieron los requerimientos funcionales y no funcionales del sistema. Estos permiten especificar de manera clara lo que el programa debe realizar (funcionalidad) y las condiciones bajo las cuales debe operar (restricciones de calidad y entorno).

### Requerimientos funcionales (RF)

Código	Requerimiento funcional
RF1	Solicitar al usuario los siguientes datos: marca, modelo, cilindrada, tipo de combustible y capacidad en pasajeros.
RF2	Recibir los datos por teclado utilizando la clase Scanner de Java.
RF3	Mostrar en consola los valores ingresados por el usuario, respetando el formato establecido.
RF4	Permitir el uso de variables de tipo String e int para el almacenamiento de la información.

### Requerimientos no funcionales (RNF)

Código	Requerimiento no funcional
RNF1	Ejecutar sin un IDE, únicamente desde la línea de comandos, utilizando el compilador javac y la máquina virtual de Java (JVM).
RNF2	Comentar código línea por línea, con explicaciones breves y concisas que detallen la lógica utilizada.
RNF3	Contener README.md, evidencias gráficas, historias de usuario y cronograma de trabajo en repositorio
RNF4	Ser portable, pudiendo ejecutarse en cualquier equipo con JDK 8 o superior instalado.

# Desarrollo

## Entorno y prerequisites

Para la implementación del programa fue necesario contar con el Java Development Kit (JDK), versión 8 o superior, descargado desde el sitio oficial de Oracle. La instalación de este compilador permitió disponer tanto de la herramienta javac, utilizada para compilar los archivos fuente, como de la Java Virtual Machine (JVM), responsable de ejecutar el programa. De esta manera, se garantizó que el código pudiera ser procesado sin necesidad de un entorno de desarrollo integrado (IDE)

## Código Utilizado.

El programa fue desarrollado en un archivo denominado VehiculoApp.java. A continuación, se presenta el código, el cual solicita al usuario datos de un vehículo y posteriormente los muestra en consola. Cada línea del código se encuentra comentada para explicar su función.

```
// Importa la librería Scanner, necesaria para leer datos desde el teclado
import java.util.Scanner;

// Se define la clase principal del programa, debe llamarse igual que el archivo
(VehiculoApp.java)
class VehiculoApp {

// Método principal, punto de entrada de la aplicación
public static void main(String[] args) {

// Se crea un objeto Scanner llamado "sc" para capturar datos ingresados por el
usuario
Scanner sc = new Scanner(System.in);

// ===== ENTRADAS =====
```

```

        // Solicita al usuario ingresar la marca del vehículo y la almacena en una variable
tipo String
        System.out.print("Marca: ");
        String marca = sc.nextLine();

        // Solicita el modelo del vehículo y lo guarda en otra variable String
        System.out.print("Modelo: ");
        String modelo = sc.nextLine();

        // Solicita la cilindrada del vehículo (dato numérico), lo guarda en una variable tipo
int
        System.out.print("Cilindrada (cc): ");
        int cilindrada = sc.nextInt();

        // Limpia el salto de línea pendiente después de leer un número (evita errores al
leer un String después)
        sc.nextLine();

        // Solicita el tipo de combustible y lo guarda en un String
        System.out.print("Tipo de Combustible (gasolina/diesel/híbrido/eléctrico): ");
        String combustible = sc.nextLine();

        // Solicita la capacidad en pasajeros (dato numérico entero)
        System.out.print("Capacidad en pasajeros: ");
        int capacidad = sc.nextInt();

        // ===== SALIDAS =====

        // Muestra todos los datos ingresados en consola
        System.out.println("\nLa marca que ha ingresado es: " + marca);
        System.out.println("El modelo que ha ingresado es: " + modelo);

```

```
System.out.println("La cilindrada que ha ingresado es: " + cilindrada + " cc");
System.out.println("El tipo de combustible es: " + combustible);
System.out.println("Tiene una capacidad de " + capacidad + " pasajeros.");

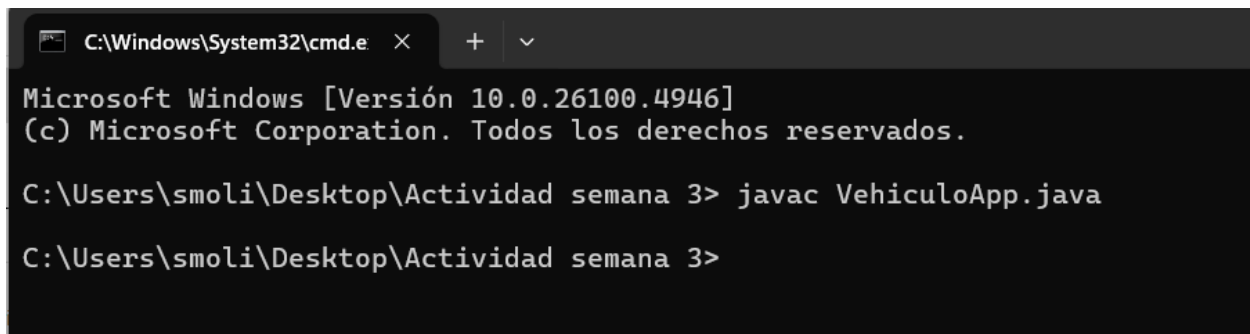
// Cierra el objeto Scanner para liberar recursos
sc.close();
}
}
```

## Proceso de compilación

Para compilar el programa se utilizó la línea de comandos (CMD), ubicándose en el directorio donde se guardó el archivo VehiculoApp.java. El comando ejecutado fue:

```
javac VehiculoApp.java
```

Este comando invocó al compilador de Java (javac), el cual transformó el código fuente en un archivo denominado VehiculoApp.class. Dicho archivo contiene el bytecode, un conjunto de instrucciones intermedias que pueden ser interpretadas por la JVM.



```
C:\Windows\System32\cmd.e  X  +  v
Microsoft Windows [Versión 10.0.26100.4946]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\smoli\Desktop\Actividad semana 3> javac VehiculoApp.java

C:\Users\smoli\Desktop\Actividad semana 3>
```

## Proceso de ejecución

Una vez compilado se ejecutó el siguiente comando:

```
C:\Users\smoli\Desktop\Actividad semana 3> java VehiculoApp
Marca:
```

Este comando cargó el archivo VehiculoApp.class en la JVM, permitiendo su ejecución en el equipo. El sistema solicitó al usuario los datos requeridos y posteriormente los imprimió en pantalla, confirmando el ciclo de entrada–proceso–salida.

## Ejemplo de la ejecución

```
Microsoft Windows [Versión 10.0.26100.4946]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\smoli\Desktop\Actividad semana 3> javac VehiculoApp.java

C:\Users\smoli\Desktop\Actividad semana 3> java VehiculoApp
Marca: Nissan
Modelo: Iq23
Cilindrada (cc): 2000
Tipo de Combustible (gasolina/diesel/híbrido/eléctrico): Híbrido
Capacidad en pasajeros: 2

La marca que ha ingresado es: Nissan
El modelo que ha ingresado es: Iq23
La cilindrada que ha ingresado es: 2000 cc
El tipo de combustible es: Híbrido
Tiene una capacidad de 2 pasajeros.

C:\Users\smoli\Desktop\Actividad semana 3>
```



## Documentación en GitHub

Con el fin de respaldar el trabajo y aplicar buenas prácticas de control de versiones, se implementó un repositorio en la plataforma GitHub, el cual centraliza toda la evidencia asociada a la actividad. Este repositorio contiene el archivo fuente VehiculoApp.java, comentado línea por línea.

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings		
main	VehiculoApp-Semana3 /	
Nicolas9317 docs: agregar historias de usuario		28647e8 - 2 minutes ago History
Name	Last commit message	Last commit date
Archivos.png	Add files via upload	19 minutes ago
Compilacion.png	Add files via upload	19 minutes ago
Creacion de clase VehiculoApp.png	Add files via upload	19 minutes ago
README.md	Update README.md	3 minutes ago
VehiculoApp.java	Add files via upload	19 minutes ago
historias_usuario.md	docs: agregar historias de usuario	2 minutes ago

Además, se incluyó un archivo README.md en el que se documentan los objetivos del proyecto, las instrucciones de compilación y ejecución desde la consola, así como un ejemplo de salida obtenido en pruebas reales. El README también incorpora un listado de requerimientos funcionales y no funcionales, debidamente diferenciados, y una sección con historias de usuario que permiten comprender la aplicación desde la perspectiva de distintos actores, junto con sus criterios de aceptación.

README.md

### VehiculoApp – Semana 3

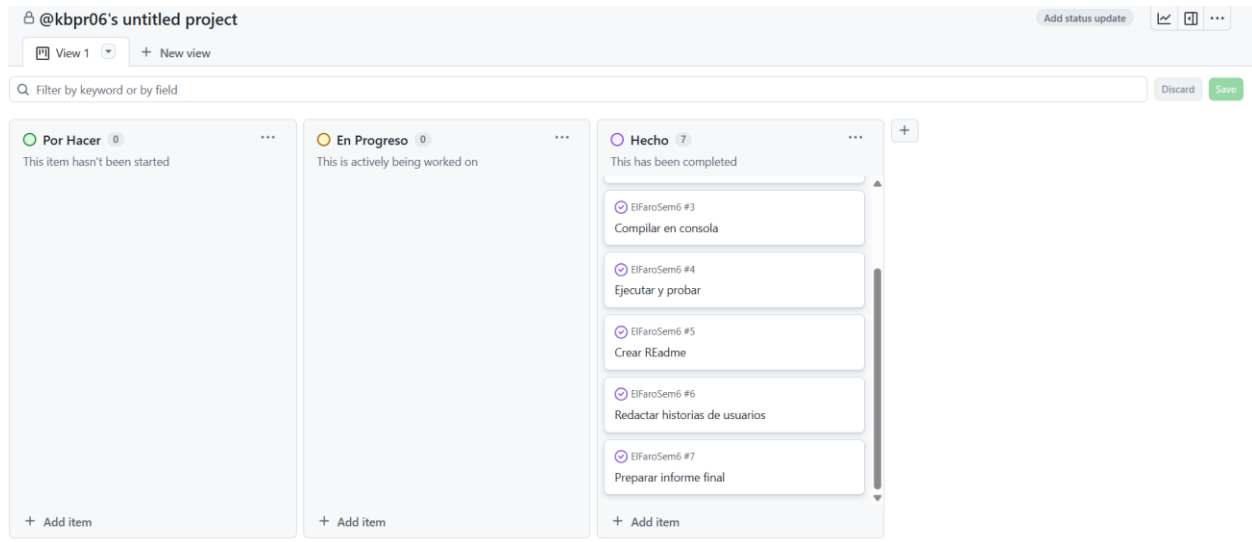
#### Descripción

Este repositorio contiene el programa en **Java** (sin uso de IDE) que solicita al usuario los siguientes datos de un vehículo:

- Marca
- Modelo
- Cilindrada (cc)
- Tipo de combustible
- Capacidad en pasajeros

El sistema recibe la información por teclado mediante la clase **Scanner** y luego la muestra en consola.

Para complementar, el repositorio contiene evidencias gráficas que ilustran el proceso de compilación, la ejecución en consola y los archivos generados por el compilador. Asimismo, se configuró un tablero de trabajo en GitHub Projects bajo el enfoque Kanban, donde se planificaron y gestionaron tareas como la escritura del código, la documentación del mismo, la recopilación de evidencias, la elaboración del README y la redacción del informe final.



En conjunto, esta documentación integrada en GitHub no solo refuerza la transparencia y organización del proceso, sino que también promueve el uso de herramientas profesionales que facilitan la gestión de proyectos de software de manera clara y ordenada.

## Historias de Usuario

En el marco de la actividad se elaboraron historias de usuario con el fin de representar los requerimientos del sistema desde la perspectiva de los actores involucrados. Estas historias permiten vincular las funcionalidades técnicas con necesidades concretas, estableciendo criterios de aceptación que garantizan el cumplimiento de los objetivos planteados.

- HU1: Como operador de ingreso, quiero registrar los datos de un vehículo en la consola para que el sistema los muestre en pantalla.  
Criterios de aceptación: se solicitan los cinco campos definidos (marca, modelo, cilindrada, tipo de combustible y capacidad en pasajeros), y se muestran en consola exactamente los valores ingresados por el usuario.
- HU2: Como docente evaluador, quiero que el estudiante compile y ejecute el programa por consola, para comprobar que domina el uso de los comandos javac y java.  
Criterios de aceptación: el archivo README incluye los comandos de compilación y ejecución, y el repositorio presenta capturas de evidencias que demuestran la ejecución correcta del programa.
- HU3: Como estudiante, quiero documentar el proceso completo en GitHub (código comentado, requerimientos, cronograma y evidencias) para organizar mi trabajo y cumplir con la rúbrica de evaluación.  
Criterios de aceptación: el repositorio contiene el código comentado línea por línea, un README estructurado con toda la documentación solicitada y un cronograma inicial elaborado en GitHub Projects.

## Cronograma inicial: GitHub Project

Con el objetivo de planificar y organizar las distintas etapas de la actividad, se elaboró un cronograma inicial utilizando la herramienta GitHub Projects, bajo la modalidad de tablero Kanban. Este tablero permitió distribuir las tareas en tres columnas principales (*To Do*, *In Progress* y *Done*), lo que facilitó la visualización del estado de avance y la asignación de prioridades.

Las tareas incluidas en el cronograma fueron las siguientes:

- Escribir el código fuente del programa VehiculoApp.java.
- Comentar línea por línea el código, explicando la lógica y el propósito de cada instrucción.
- Compilar y ejecutar el programa en consola mediante los comandos `javac` y `java`.
- Capturar evidencias gráficas del proceso de compilación y ejecución.
- Elaborar el archivo README.md con instrucciones, requerimientos, historias de usuario y evidencias.
- Documentar las historias de usuario en el repositorio.
- Preparar el informe final en Word/PDF para su entrega.

La utilización de GitHub Projects permitió llevar un registro ordenado de las actividades, reflejando tanto el trabajo planificado como el realizado, y garantizando una mejor gestión del tiempo y cumplimiento de la rúbrica de evaluación.

## Conclusión

El desarrollo de la actividad permitió aplicar los conceptos básicos de Java sin IDE, reforzando el uso de la consola para compilar y ejecutar programas. Se implementó un sistema sencillo que solicita y muestra datos de un vehículo utilizando la clase Scanner, cumpliendo con los requerimientos funcionales y no funcionales definidos.

Además, se fortalecieron buenas prácticas de documentación al crear un README.md estructurado, con instrucciones claras de compilación y ejecución, evidencias gráficas del proceso, así como historias de usuario y un cronograma de trabajo.

Con esta experiencia, se afianzaron competencias clave en programación estructurada, control de versiones con GitHub y gestión de proyectos mediante la organización de entregables.

## Bibliografía

- Oracle. (s. f.). Java SE Development Kit 8 Downloads. Recuperado el 1 de septiembre de 2025, de <https://www.oracle.com/cl/java/technologies/javase/javase-jdk8-downloads.html>
- Oracle. (s. f.). Scanner (Java Platform SE 8). Recuperado el 1 de septiembre de 2025, de <https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html>
- Oracle. (s. f.). Java Platform, Standard Edition 8 Documentation. Recuperado el 30 de agosto de 2025, de <https://docs.oracle.com/javase/8/docs/>