TECMILENIO

P.O.O

CHRISTIAN SANABRA

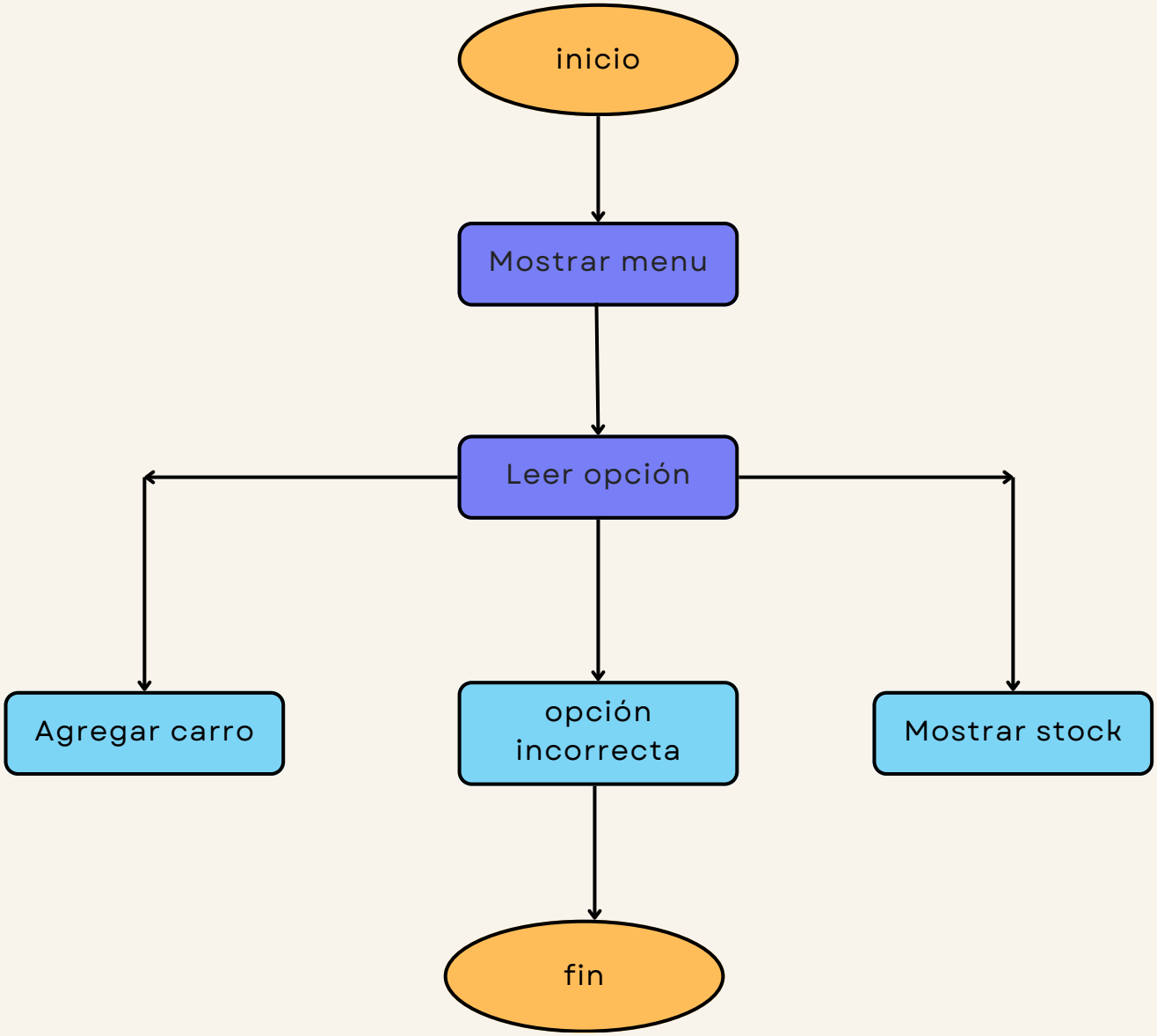
ACTIVIDDAD I

TECMILENIO

Diagrama
DE FLUJO

En este documento presentamos la solución a un ejercicio propuesto por nuestro profesor, cuyo objetivo es desarrollar un sistema para registrar y consultar automóviles a través de un menú interactivo. Para ello, implementamos el paradigma de programación orientada a objetos, creando objetos sencillos y utilizando la línea de comandos como interfaz de interacción con el usuario. Asimismo, incorporamos diagramas UML para representar de forma clara y estructurada la lógica de nuestro programa.

CLI


inicio
Mostrar menu
Leer opción
Agregar carro
opción incorrecta
Mostrar stock
fin

TECMILENIO

CLI

MOSTRAR MENU

Practicamente nos enseña nuestras opciones, tales como lo son Agregar Carro, Ver Stock y Salir.

LEER OPCION

Aqui es donde el usuario elige la opción que va a utilzar del menu que se le muestra.

INICIO/FIN

Muestran tanto el inicio del proceso como el termino del mismo, ya sea recopilando la información que necesitamos o mostrandonos un mensaje de finalización del proceso

AGREGAR CARRO

Si se elige esta opción damos paso a preguntar a nuestro usuario las características del carro por agregar, siendo estas el número de puertas, marca, color y modelo.

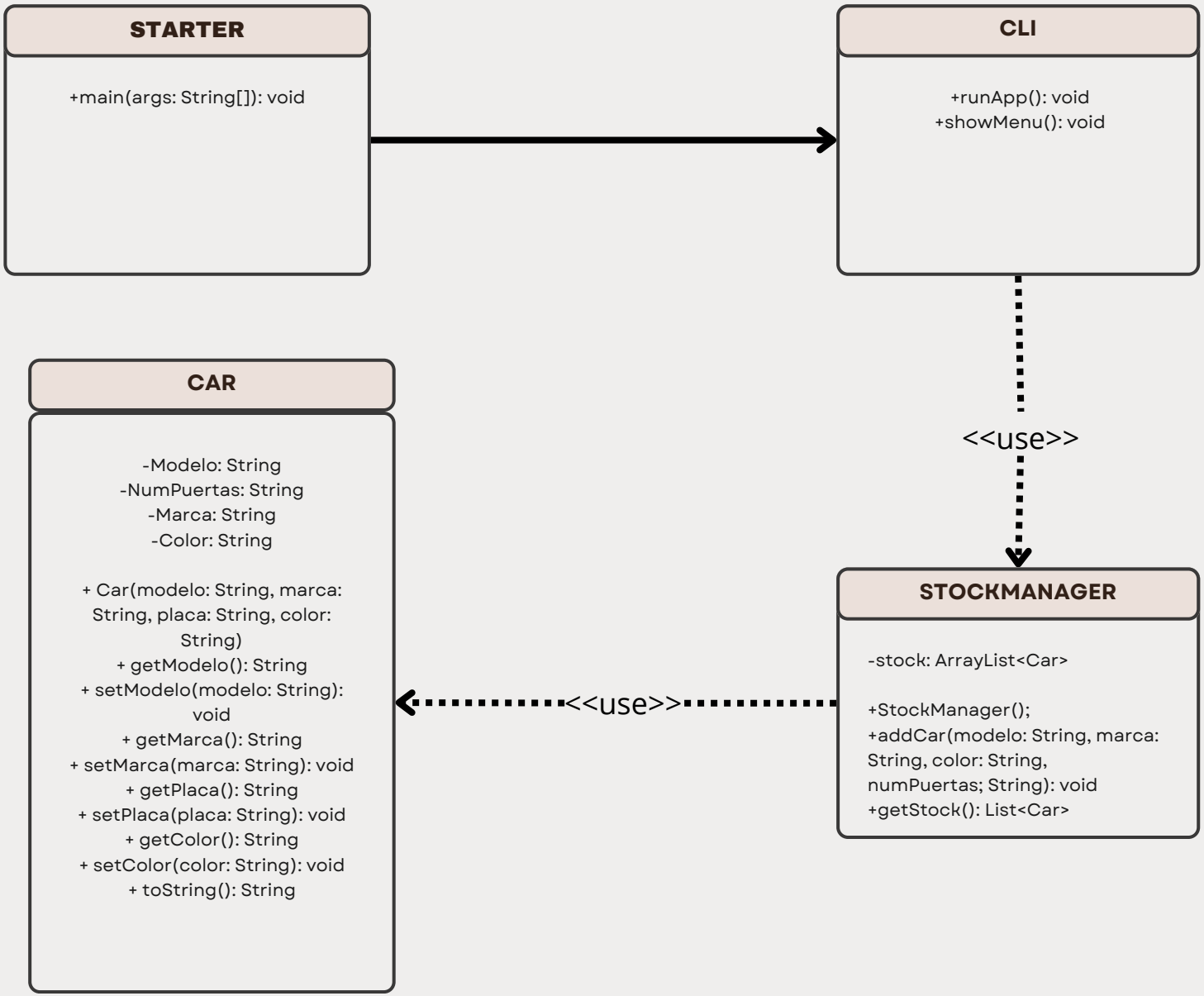
OPCION INCORRECTA

SI se elige algo que no concuerde con lo pedido por la aplicación, se nos arroja un mensaje de error, por ejemplo con el número de puertas si es fuera del rango establecido (2-5) nos dará el m,ensaje de error.

MOSTRAR STOCK

En esta opcion mostramos una lista con el stock de catálogo de los carros con los que contamos, en el caso de no haber ingresado ninguno todavía, se nos muestra una lista vacia.

DIAGRAMA DE CLASES



El diagrama de clases que se muestra a continuación refleja la estructura que hemos implementado en nuestro código. En primer lugar, la clase Starter actúa como nuestra clase principal (main), desde la cual se ejecuta toda la aplicación. Esta clase depende de CLI, que se encarga de gestionar la interfaz de usuario, mostrando el diseño y las interacciones necesarias. A su vez, CLI se apoya en StockManager, utilizando métodos como addCar y getStock para agregar y mostrar los vehículos registrados.

Además, StockManager mantiene una dependencia con la clase Car, que representa a los automóviles y gestiona sus atributos, como modelo, placa, color y marca. Esta organización nos permite un procesamiento ordenado y centralizado de los parámetros relacionados con los vehículos.
Hemos estructurado nuestro trabajo en cuatro clases distintas, cada una con un propósito y conjunto de métodos específicos. Esta separación facilita la comprensión del código por parte de otros programadores y asegura que las funciones estén bien definidas y documentadas. Al utilizar las convenciones de nomenclatura estándar de Java, hemos mantenido la coherencia y claridad en todo el proyecto.

Un aspecto clave que aprendimos en este ejercicio es cómo lograr una interacción efectiva entre las clases para crear procesos más claros y reutilizables. En lugar de replicar métodos en cada clase, los definimos una sola vez y los reutilizamos siguiendo los principios de la programación orientada a objetos. Esto nos permitió desarrollar un código más modular, eficiente y fácil de mantener.