|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Ing. Guadalupe Lizeth Parrales Romay |
| *Asignatura:* | Programación Orientada a Objetos |
| *Grupo:* | 04 |
| *No de Práctica(s):* | 5 |
| *Integrante(s):* | Dávila Ortega Jesús Eduardo - No. Cuenta: 317199860  Díaz Hernández Marcos Bryan - No. Cuenta: 317027253  Pareja Ávila Emiliano - No. Cuenta: 317081345  Vázquez Zavala Oliver Alexis - No. Cuenta: 317202263 |
| *No. de Equipo de cómputo empleado:* |  |
| *No. de Lista o Brigada:* | 12,14, 30, 37 |
| *Semestre:* | 2021-1 |
| *Fecha de entrega:* | 7 de noviembre del 2020 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_

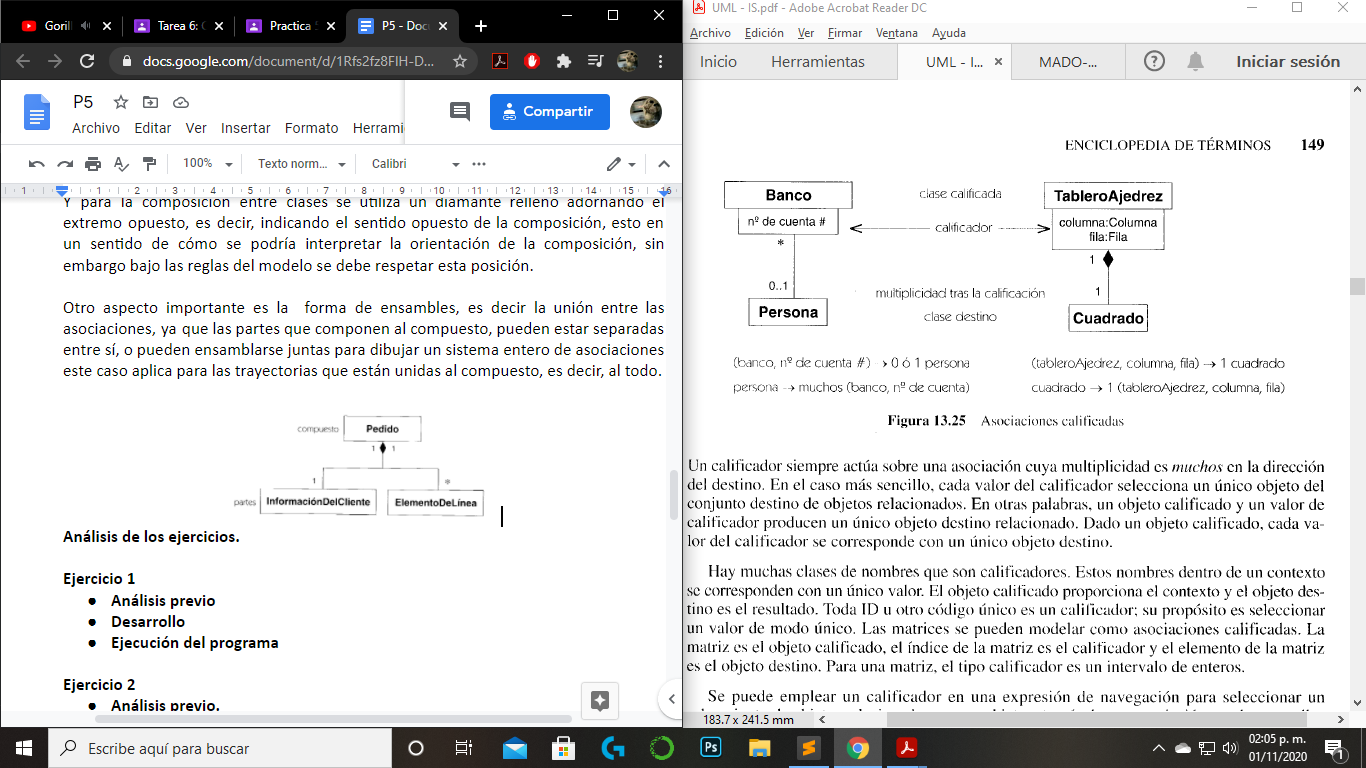
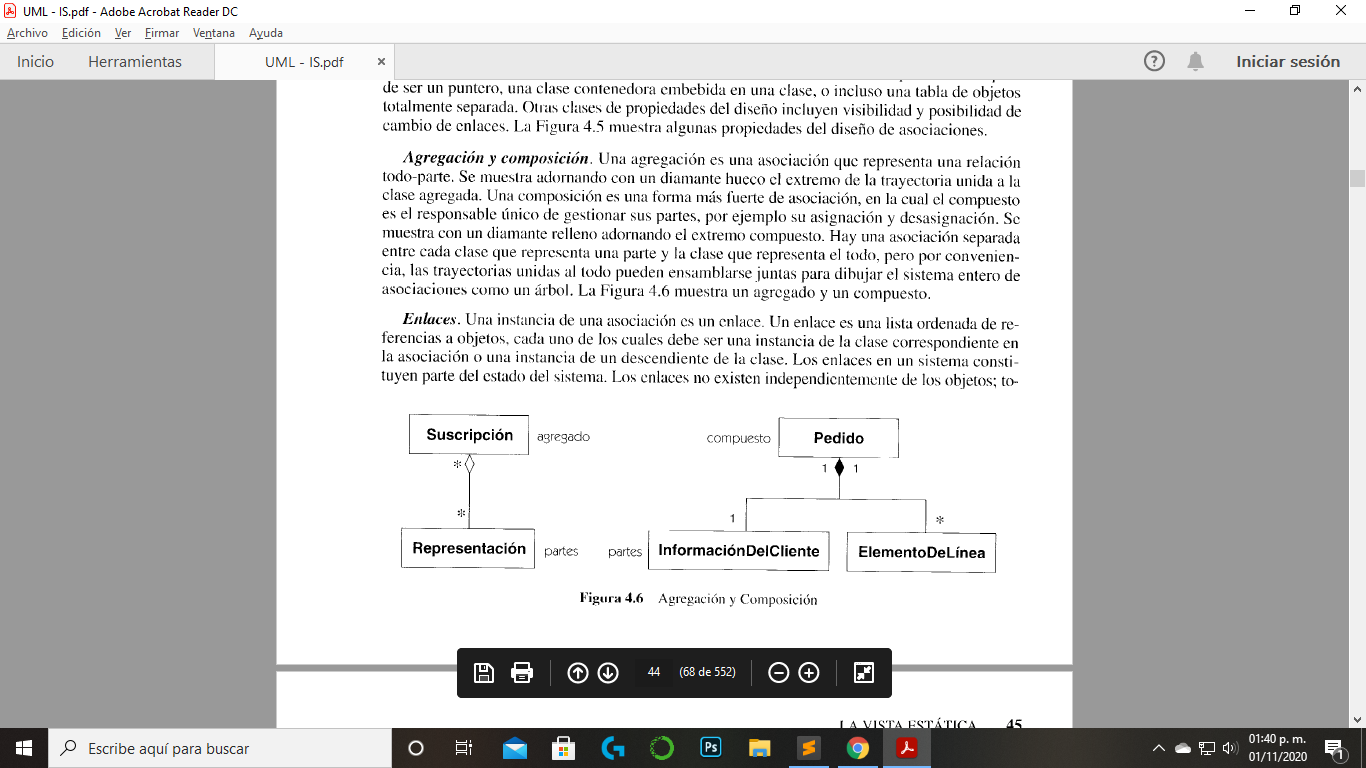
***Objetivo de la práctica.***

Aplicar el concepto de abstracción para el diseño de clases que integran una solución,

utilizando el encapsulamiento para proteger la información y ocultar la implementación.

***Introducción.***

En base al UML, se tiene que una composición es un tipo de asociación y una asociación describe las conexiones discretas entre objetos u otras instancias de un sistema. La composición es un tipo fuerte de asociación porque, se compone de dos elementos, el compuesto y las partes que lo componen, donde el compuesto puede gestionar las partes que lo componen, como la asignación y la ejecución.

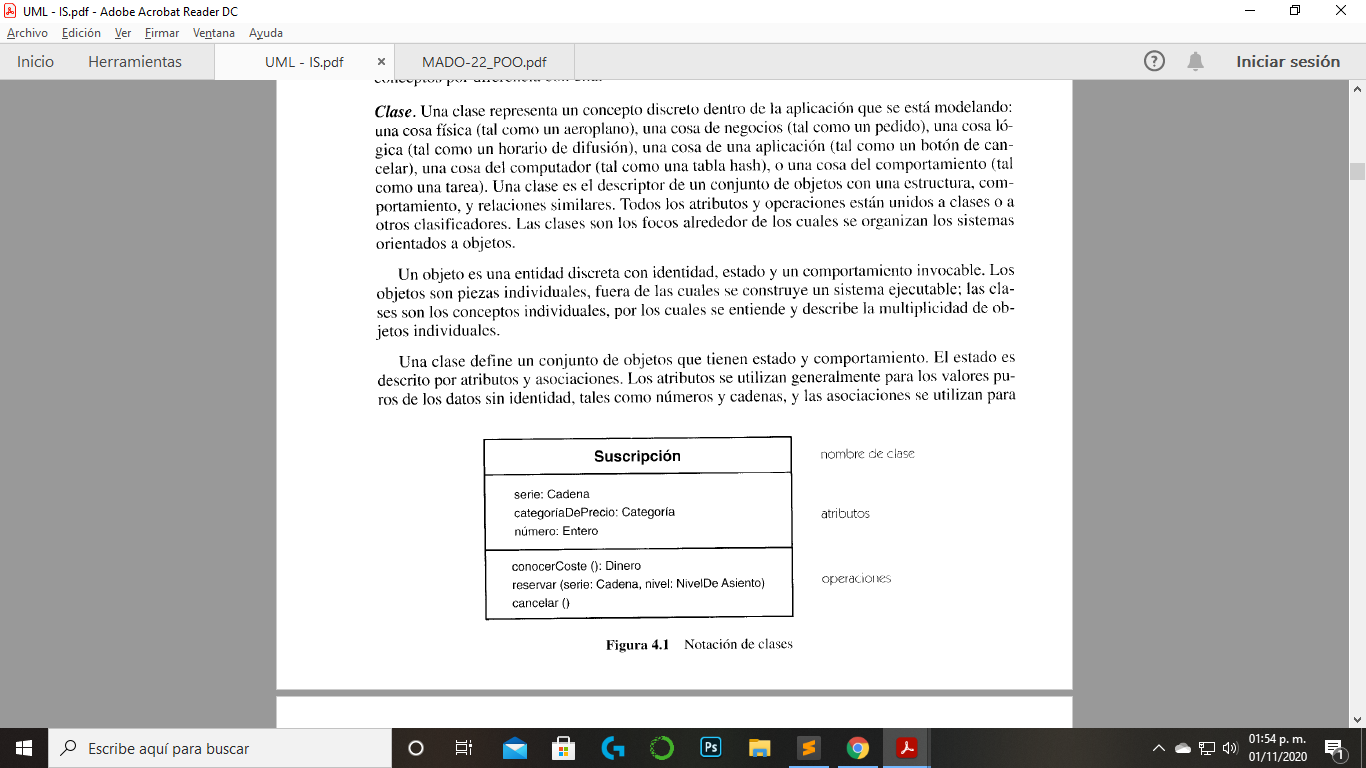


Como se muestra, el compuesto está formado por más elementos, que son las partes que lo componen y que este puede disponer para realizar sus acciones. Ahora dándole un enfoque hacia la programación se tiene que son las referencias a objetos de otras clases, las cuales se colocan como miembros de una clase.

El UML, no se define específicamente para la programación ya que el lenguaje unificado de modelado, surgió para la industria del software pero su implementación no queda ahí, sino que se convirtió en un estándar en la industria, aplicando o siendo utilizado en las industrias donde sea necesario, la construcción de modelos como condición previa al diseño y para la posterior creación de prototipos, si bien las distintas metodologías del software envuelven esto procesos, también las industriales, y en general industrias donde se crean productos, es necesario el crear los modelados.

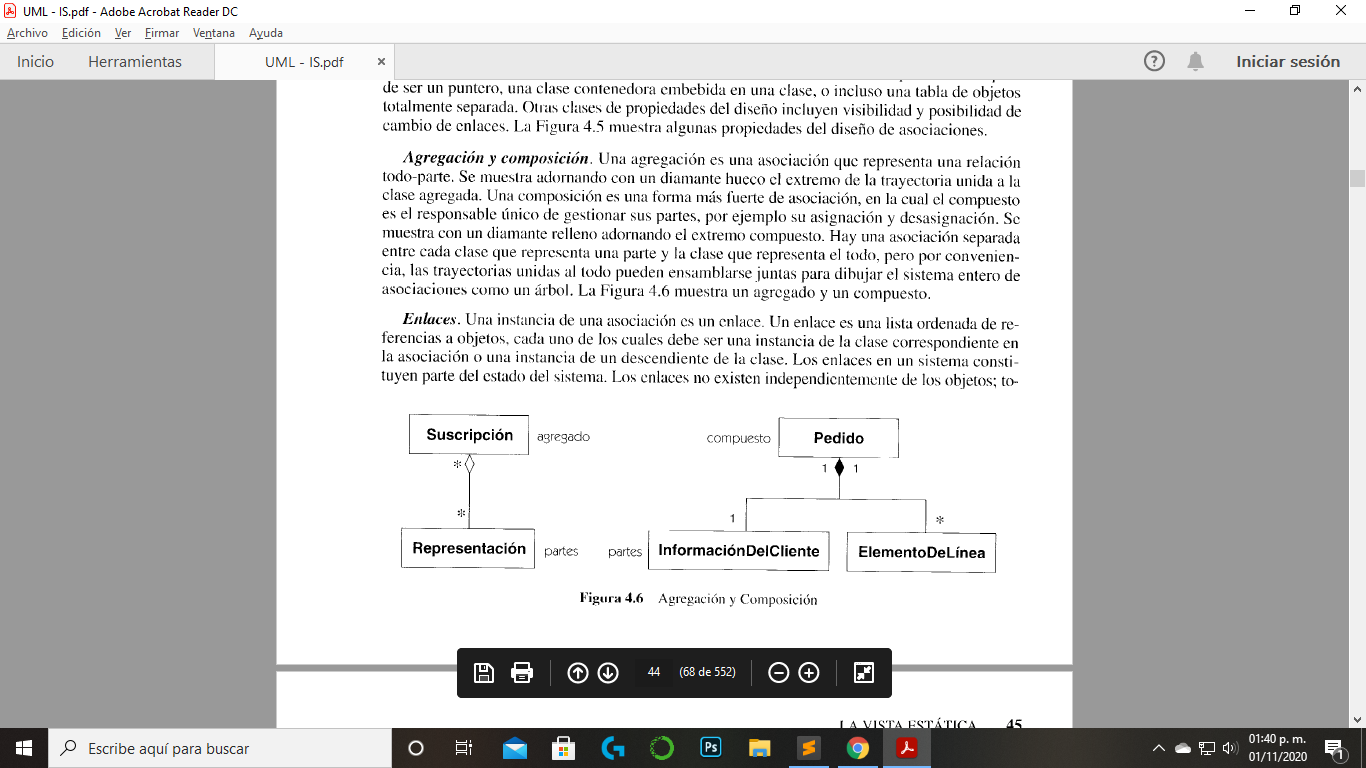
Teniendo el enfoque y el objetivo del UML claro, podemos entender que se aplica a la programación orientada, ya que este modelo contiene grandes niveles de abstracción, además de diferentes tipos de clasificadores.

Para nuestra práctica lo que utilizaremos será el clasificador: clase, y la composición, donde la clase se define como muestra la imagen:



Y para la composición entre clases se utiliza un diamante relleno adornando el extremo opuesto, es decir, indicando el sentido opuesto de la composición, esto en un sentido de cómo se podría interpretar la orientación de la composición, sin embargo bajo las reglas del modelo se debe respetar esta posición.

Otro aspecto importante es la forma de ensambles, es decir la unión entre las asociaciones, ya que las partes que componen al compuesto pueden estar separadas entre sí, o pueden ensamblarse juntas para dibujar un sistema entero de asociaciones este caso aplica para las trayectorias que están unidas al compuesto, es decir, al todo.



**Análisis de los ejercicios.**

**Ejercicio 1**

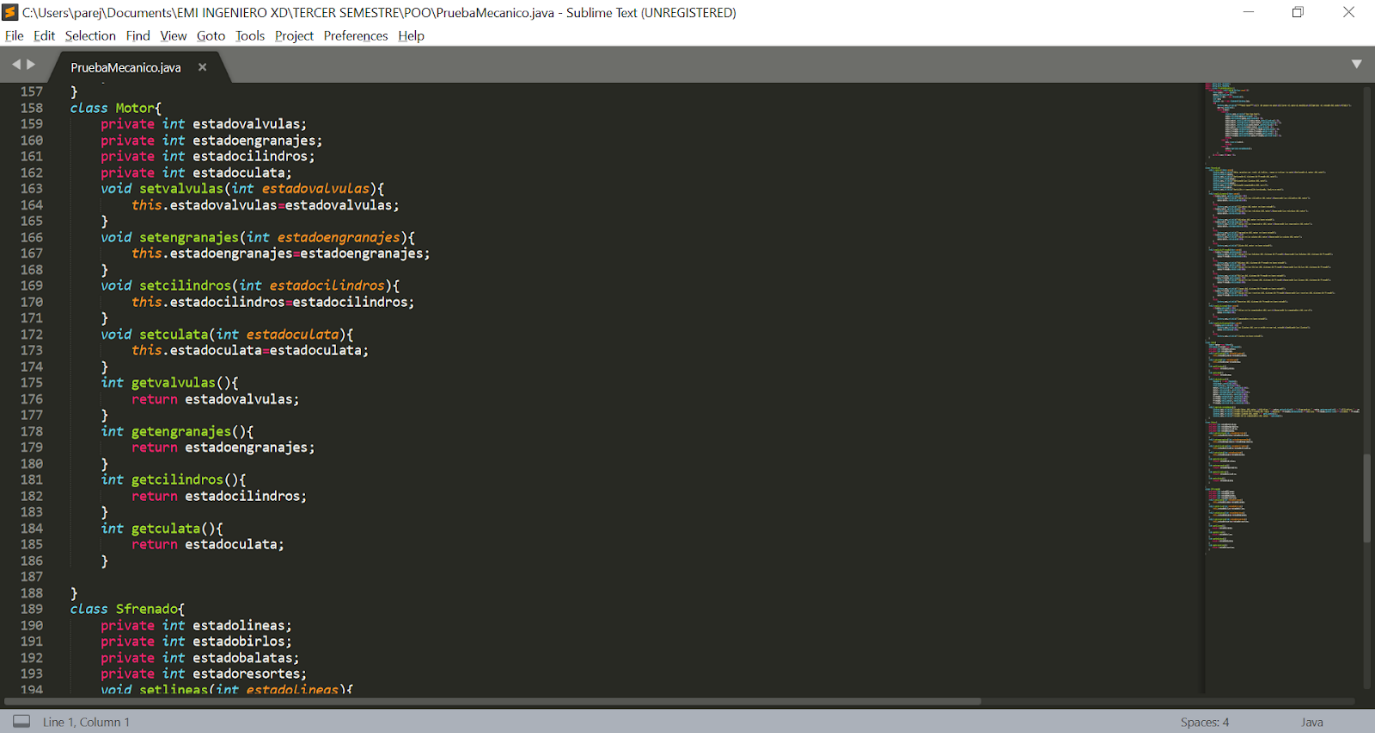
* **Análisis previo**

Lo primero que analizamos sobre este ejercicio es las partes del motor o del sistema de frenado de un automóvil, ya que se componen de varias partes, por ende investigamos las partes de estos y decidimos ingresar dentro de nuestro programa las partes más importantes, que en este caso fueron para el motor, las válvulas, los engranajes, la culata y los cilindros, por su parte para el sistema de frenado sus partes más importantes son las líneas, los birlos, las balatas y los resortes que mantienen a las líneas. También analizamos que un carro no solo se compone de estos 2 sistemas, son muchas más partes lo que lo componen y decidimos agregar 2 elementos fundamentales de los carros, que son las llantas, y la computadora del automóvil, ya que esta se conecta con el motor para hacer que este funcione. Después para el análisis de un mecánico, entendimos que esta persona es la única que puede cambiar los atributos de un automóvil a partir de los métodos que tenga, por tanto debe de recibir en sus parámetros un auto para poder realizar cambios dentro de él. Otra cosa que analizamos fue el hecho de que al auto se le debían dar valores iniciales, y que estos debían ser creados por un método auxiliar, puesto a que haciendo una abstracción a la vida real, solo una persona especializada puede saber en qué tan buen estado se encuentran las partes de un automóvil. Y comprendimos que los atributos de la clase automóvil debían de estar en constante cambio, por tanto debíamos hacer métodos o instrucciones que modifican los atributos del auto, para que así el mecánico que modelamos pueda hacer cambios.

* **Desarrollo**

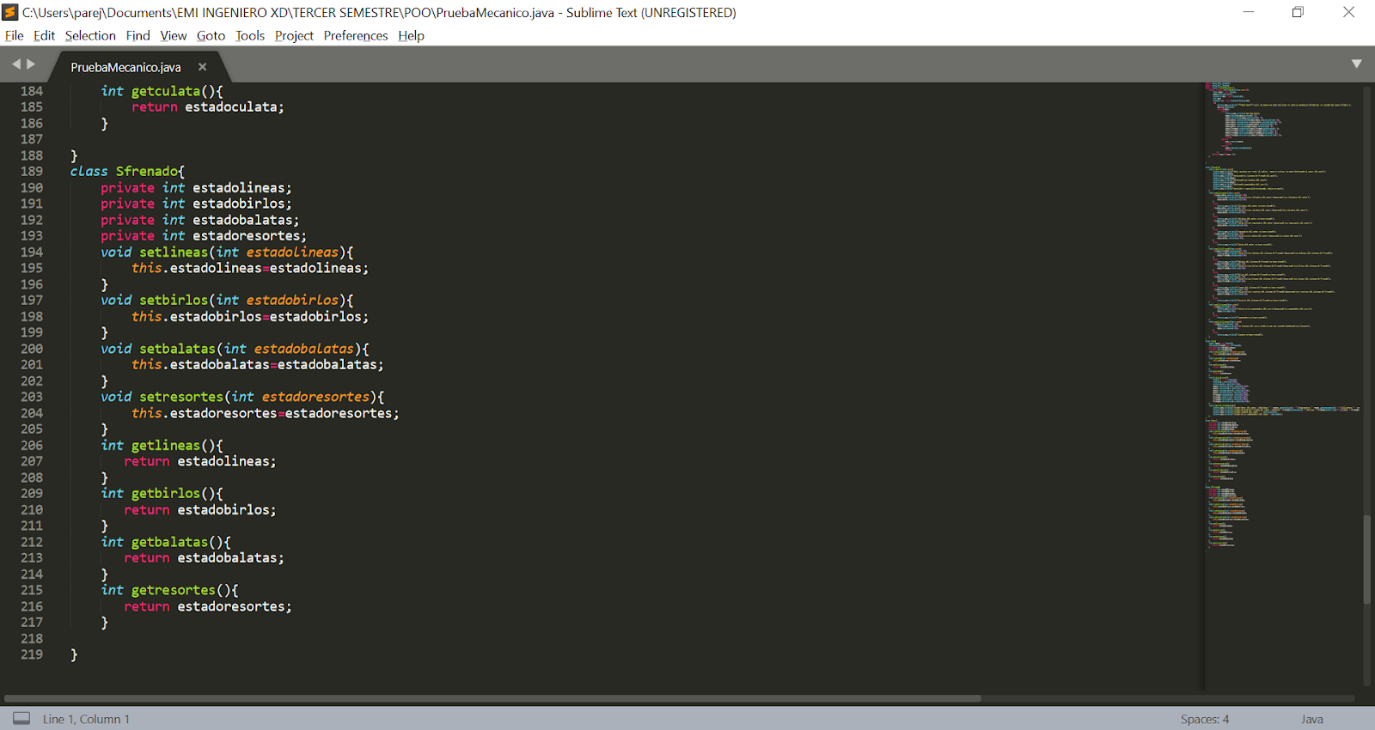
Primero desarrollamos las 2 clases base de las que se compone un automóvil que es el motor y un sistema de frenado, dentro de la clase motor, definimos 4 atributos de tipo entero, y de carácter privado, que son el estado de la culata, los engranajes, los cilindros y las válvulas del motor, decidimos declararlo como entero, basandonos al porcentaje de rendimiento de cada una de estas partes, por ejemplo si uno de estos atributos tiene un valor de 70, significa que dicho atributo está funcionando al 70% de su capacidad. Después de que declaramos sus atributos, instanciamos sus métodos setters, que básicamente, todos reciben un parámetro entero, que lleva por nombre el valor del atributo que se está modificando, que será el valor de uno de estos atributos y que para evitar el ocultamiento hacemos uso de la palabra reservada this. Después instanciamos los métodos getters de la clase motor que todas devuelven un entero, que será el valor de uno de los atributos de acuerdo con el nombre de su método getter.

* **Clase que modela al Motor del auto:**



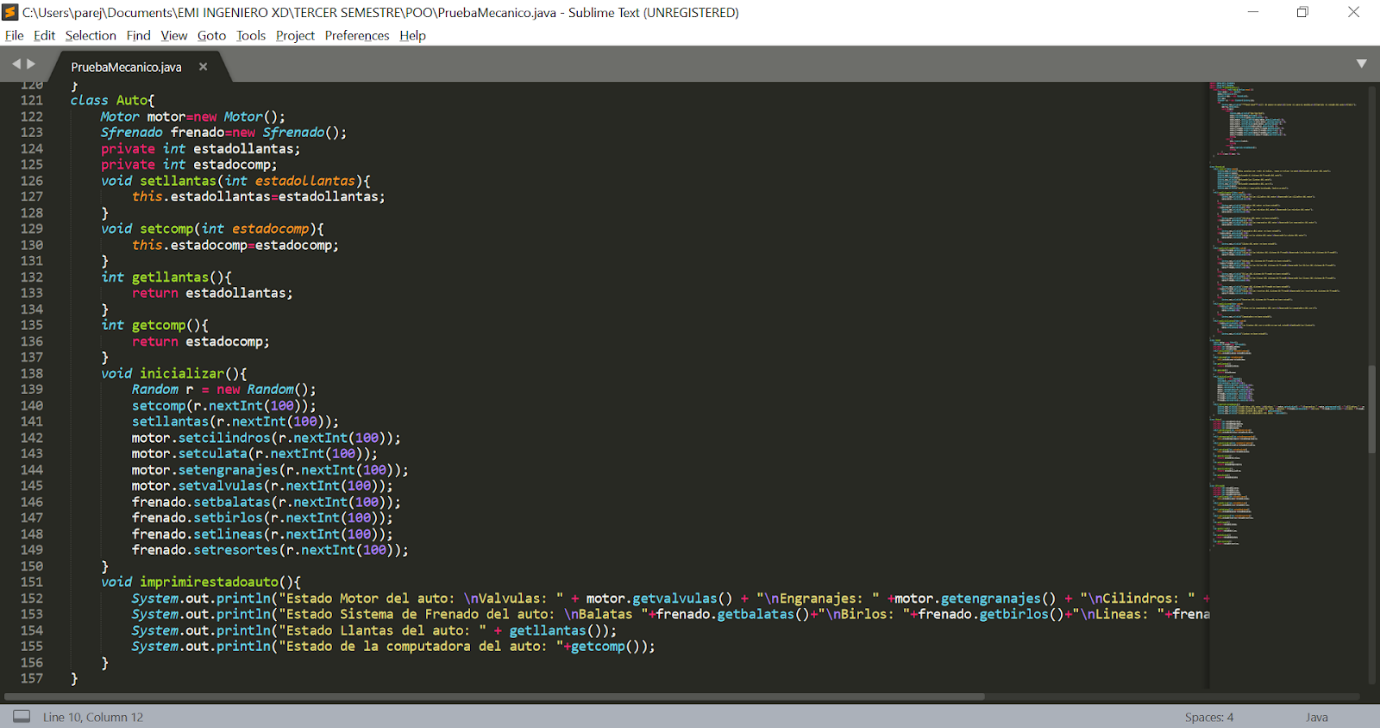
Después creamos la clase que modela un sistema de frenado, llamada Sfrenado, que siguiendo por la misma línea de la clase anterior, declaramos 4 atributos de tipo entero y de carácter privado, que también basa su valor de cada atributo de acuerdo con el porcentaje de rendimiento. Los atributos de esta clase son el estado de los birlos, las líneas, los resortes que sostienen a estas líneas y las balatas, entonces para poder modificar los valores de estos atributos instanciamos sus métodos setters, que todas son de tipo void y que reciben como parámetro un valor entero que su variable lleva por nombre el atributo que se desea modificar, entonces para poder hacer el cambio usamos la palabra reservada this. Y después para poder obtener el valor de unos de los atributos de esta clase, instanciamos los métodos getters, que todas devuelven un valor entero que es el valor del atributo que se deseé conocer.

* **Clase que modela al Sistema de Frenado del auto:**



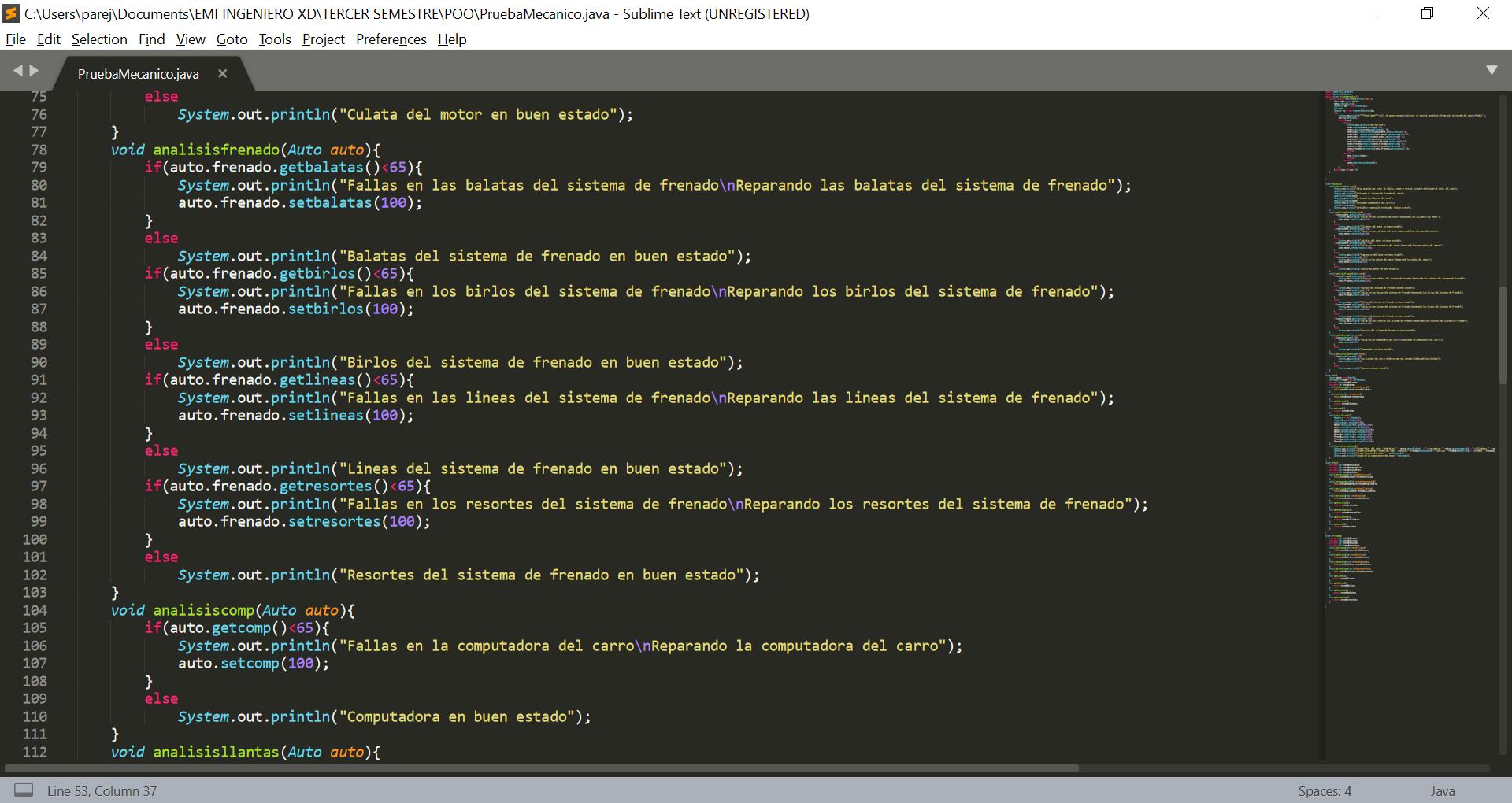
Entonces, para poder hacer uso de las clases que se crearon, creamos la clase principal que es la que modela a un automóvil, entonces dentro de esta clase, creamos un objeto de la clase Motor y otro de la clase Sfrenado, y también declaramos otros 2 atributos que son el estado de la computadora y de las llantas del auto, e instanciamos sus métodos setters y getters para poder modificar u obtener alguno de los atributos según la necesidad requerida. Donde para los setters, son métodos de tipo void que reciben como parámetro un valor entero, que será el nuevo valor del atributo y que como este parámetro, lleva por nombre el atributo, para evitar el ocultamiento usamos la palabra reservada this, y para los métodos getters, todos son de tipo entero y devuelven el valor de un atributo que se quiera conocer. Después instanciamos un método que nos ayudarían al momento de crear un auto, que es el método inicializar, debido a que un usuario no puede conocer, a menos que sea un experto, el estado de cada una de las partes de un auto, no podemos pedir que inicialice cada uno de los atributos de su auto, entonces para poder inicializarlos, instanciamos este método, donde dentro de él, primero se crea un objeto de la clase llamada Random, que nos ayudará a generar valores aleatorios, después haciendo uso de los diferentes métodos setters, modificamos el valor de los atributos del motor, los atributos del sistema de frenado, del estado de la computadora y de las llantas del auto, gracias al método nextInt, del objeto de la clase Random que creamos, ya que este método devuelve un número aleatorio en un intervalo que se le asigna como parámetro que en este caso fue 100, entonces el método devolverá un valor entero entre 0 y 100. Después, para poder conocer el valor de los atributos del auto, instanciamos un método llamado imprimirestadoauto, que con ayuda de los métodos getters de las clases Motor, Sfrenado, y Auto, imprime a pantalla el valor de todos los atributos que conforman a la clase Auto.

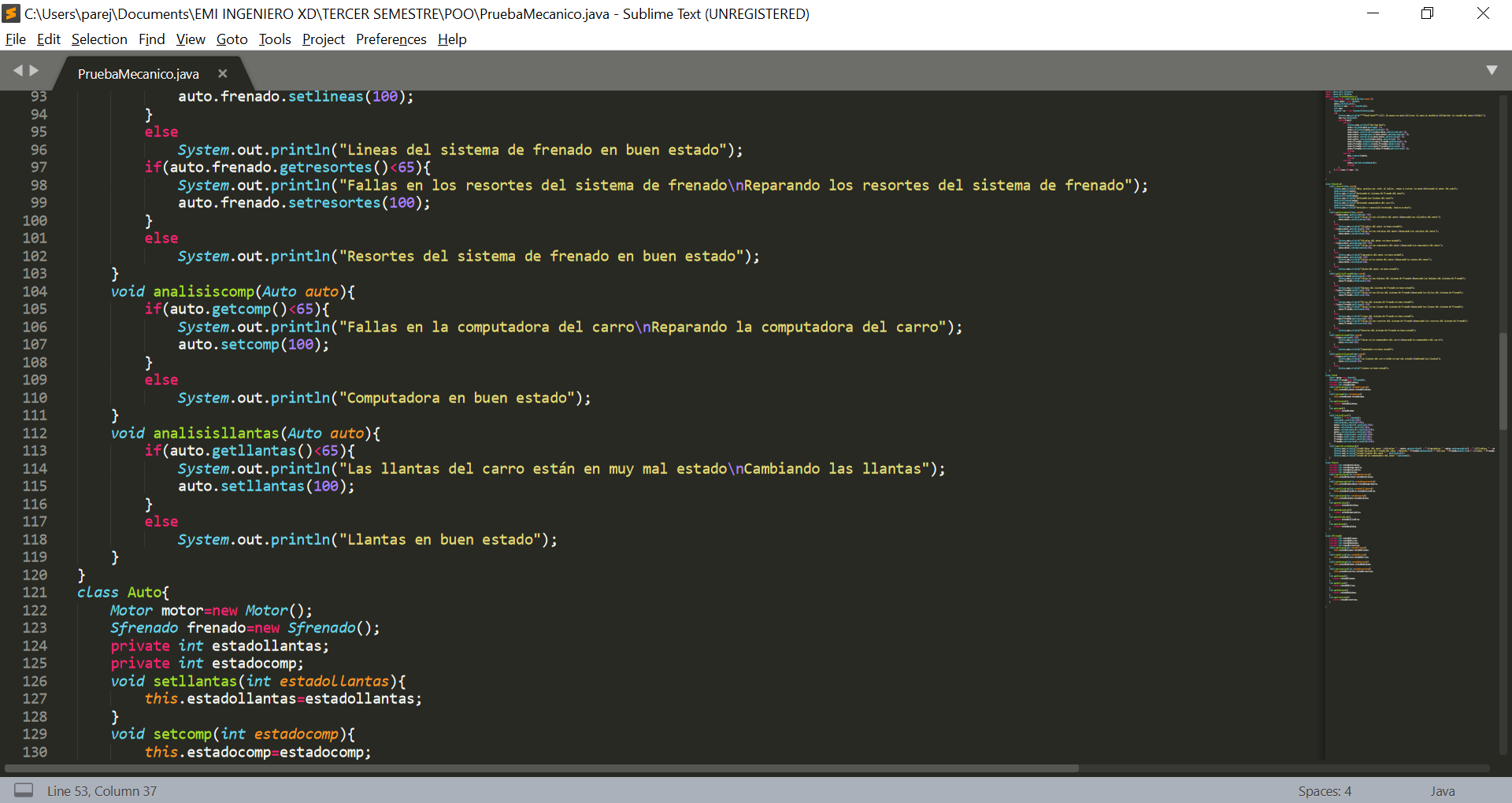
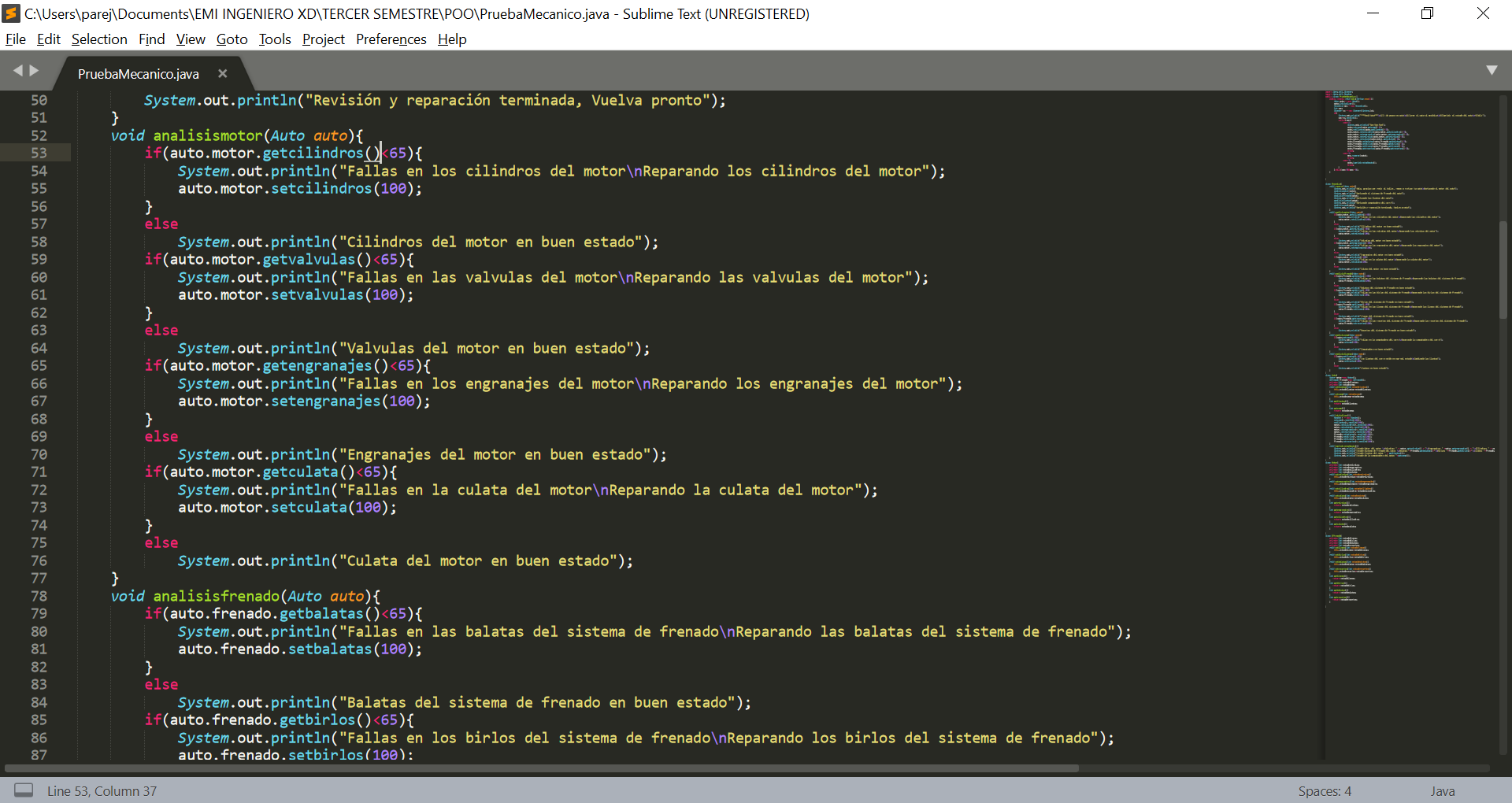
* **Clase que modela a un auto:**



Para poder hacer los cambios en los atributos del auto, necesitamos un mecánico, entonces creamos una clase llamada Mecánico, y dentro de ella creamos varios métodos, el principal se llama reparar que recibe como parámetro el auto al cual se le harán las reparaciones correspondientes, que hace la llamada a todos los demás métodos de la propia clase Mecánico, que son analisismotor, analisisfrenado, analisisllantas y analisiscomp; donde todos estos métodos también recibirán como parámetro un objeto de la clase auto, dentro del método analisismotor, se comparan los diferentes atributos del motor del auto, con el porcentaje mínimo de rendimiento que es 65%, estos valores de los atributos del motor del auto, los obtenemos gracias a los métodos getters declarados en la clase Motor, y en una estructura de control de flujo if, se hace una comparación y si el porcentaje de rendimiento de cada uno de estos atributos es menor a 65, se reporta que hay fallas en dicho atributo y su valor se cambia con los métodos getters igualmente declarados dentro de la clase Motor, a los cuales se les pasa como parámetro un 100, que será el nuevo valor del porcentaje de rendimiento del atributo del motor, en el método analisisfrenado, básicamente se hace los mismo que en el método anterior, se va comparando uno a uno los atributos del sistema de frenado del auto, y si el valor de porcentaje de rendimiento de alguno de estos atributos, que obtenemos con los métodos getters de la clase Sfrenado, es menor a 65, se cambia su valor de porcentaje de rendimiento del atributo, con los métodos setters declarados dentro de la clase Sfrenado a los cuales se les pasa como parámetros un 100, que será el nuevo valor de porcentaje del atributo del sistema de frenado que este fallando. Luego en el método analisiscomp, se obtiene con el método getcomp de la clase Auto el valor de porcentaje de rendimiento de la computadora del auto, y en una estructura de control de flujo if se hace una comparación y si este valor es menor a 65, se cambia el porcentaje de rendimiento de la computadora a 100, con el método setcomp de la clase Auto. Y para finalizar en el método analisisllantas, en una estructura de control de flujo if, se hace una comparación entre el valor del atributo llantas de la clase Auto que obtenemos con el método getllantas instanciado dentro de esta misma clase y 65, y si el valor del porcentaje de rendimiento de este atributo es menor a 65, se cambia con el método setllantas, al cual se le pasa como parámetro un número entero 100, que será el nuevo valor del atributo llantas de la clase Auto.

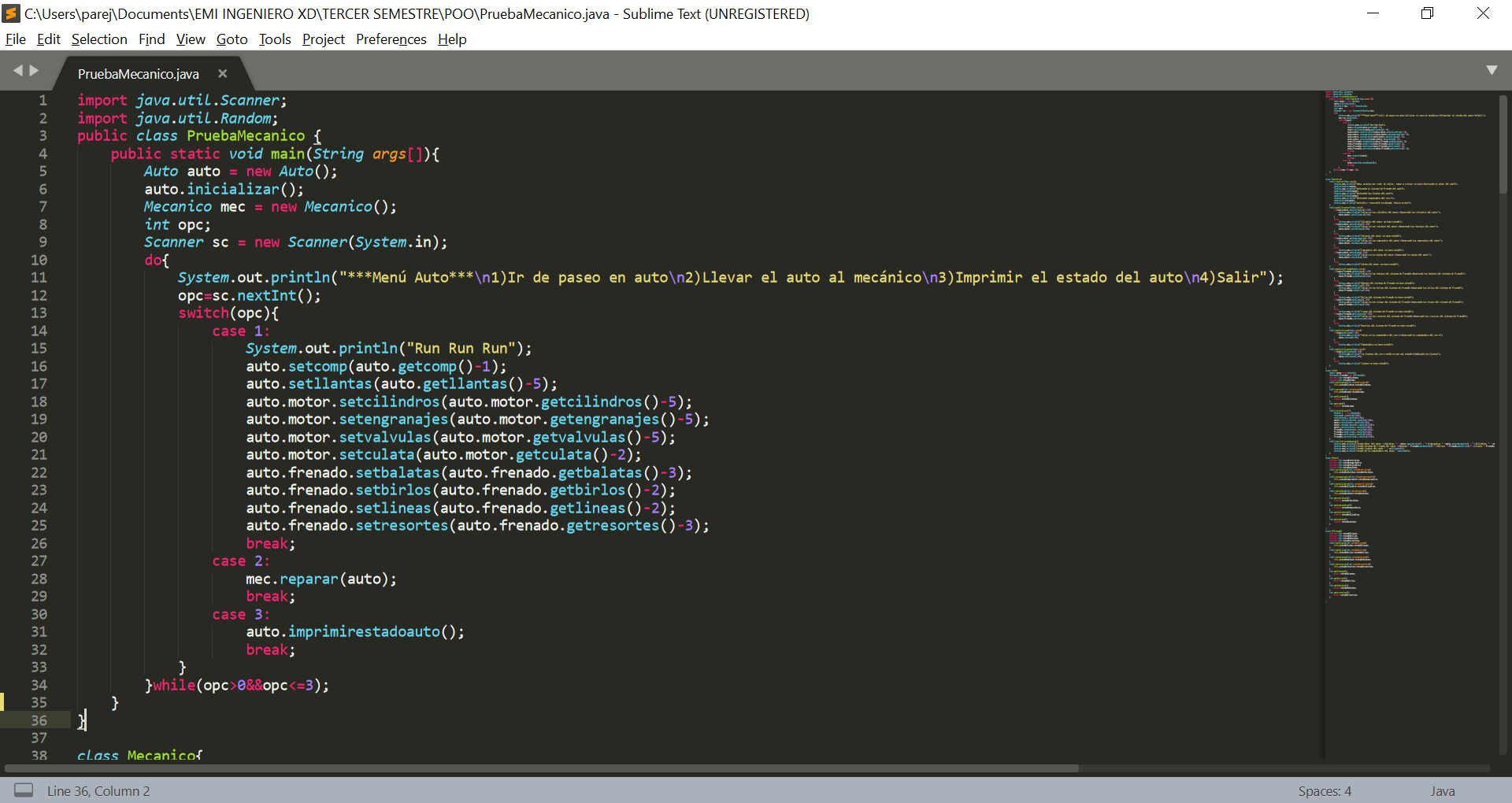
* **Clase que modela un Mecánico:**



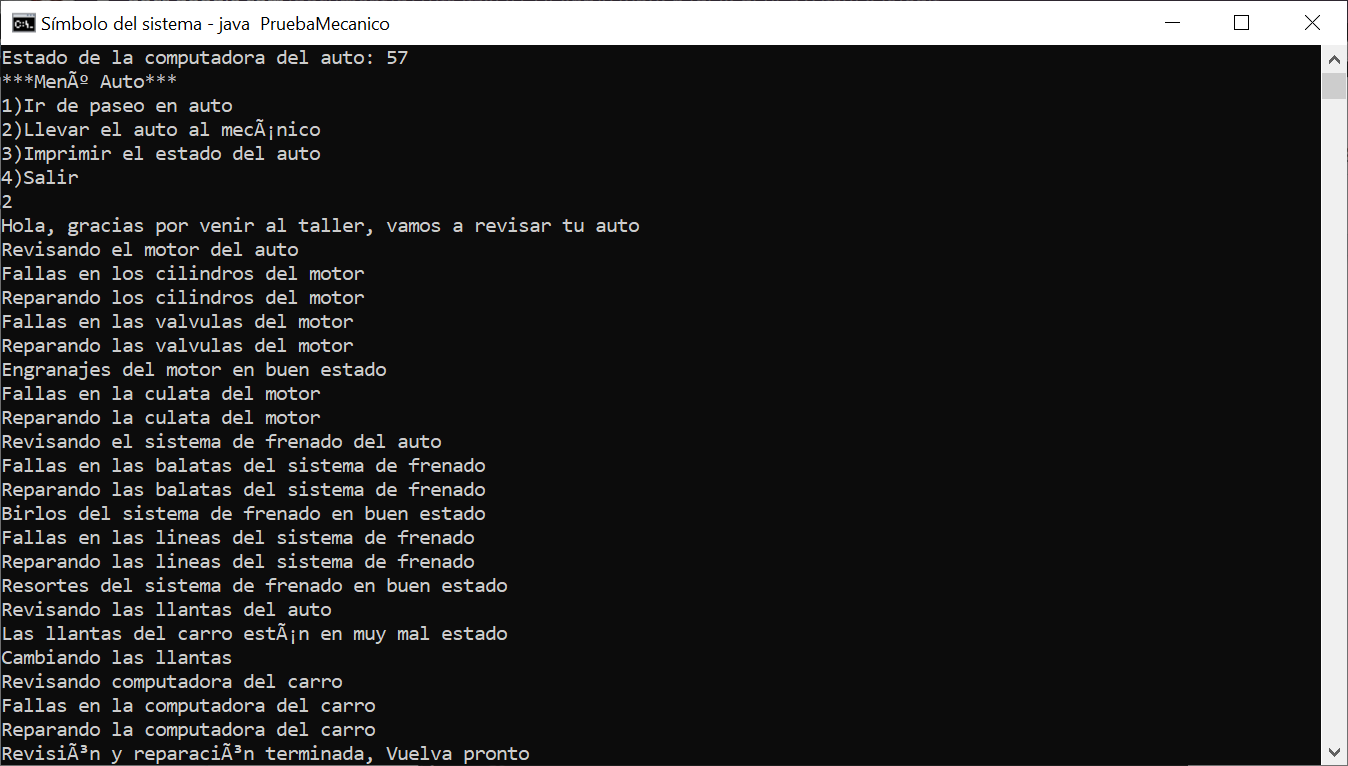


Y para finalizar, creamos nuestra clase de prueba llamada PruebaMecanico, en la cual dentro de su método main, primero creamos un objeto de la clase Auto, y llamamos al método de la clase Auto llamado inicializar, para que los valores de todos los atributos del auto se inicializaran con valores aleatorios, luego creamos otro objeto de la clase Mecanico, y declaramos una variable entera llamada opc que nos ayudaría al momento de elegir la opción de un menú para poder realizar alguna acción, y para que pudieramos leer la opción de un usuario desde el teclado, creamos un objeto de la clase Scanner, luego en una estructura de repetición do-while, hicimos nuestro menú, donde indicamos al usuario que si escoge la opción 1, se hace un viaje en un auto, si selecciona la opción 2 se hace una visita al mecánico, y si se escoge la opción 3 se imprimen a pantalla los valores del porcentaje de rendimiento de cada uno de los atributos del auto. Entonces se hace la lectura del teclado del usuario y el valor que ingrese se almacenará en la variable opc, y en una estructura switch, verificamos el valor de la variable opc es 1, “se hace un viaje en auto” pero en realidad, lo que aquí se hace es modificar el valor de los atributos del carro, entendiendo que cada que se hace un viaje en auto, este se desgasta o se va deteriorando alguna parte de su sistema, entonces con los métodos setters y getters de los atributos del auto, vamos obteniendo los valores de cada uno de estos con los getters, y restamos a su valor un número, y ese valor nuevo, con los métodos setters los reasignamos. Si el usuario escoge la opción 2, se hace la llamada al método reparar del objeto de la clase mecánico y se le pasa como parámetro el auto, y si escoge la opción 3 se llama al método imprimirestadoauto del auto que creamos.

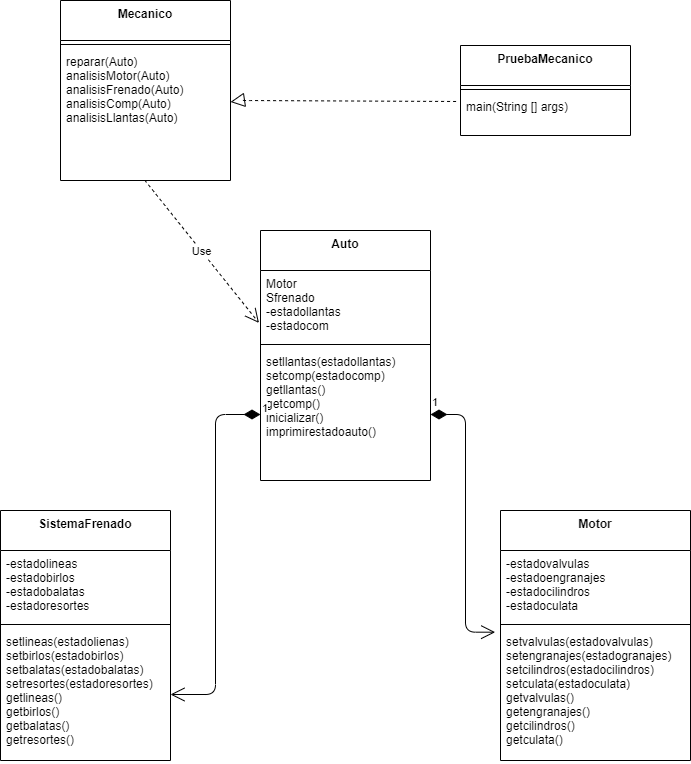
* **Clase de Prueba:**



* **Ejecución del programa**



* **Diagrama UML**

****

**Conclusiones**

* Dávila Ortega Jesús Eduardo:

En la realización de la práctica logre entender el concepto de abstracción y cómo se implementa en la programación orientada a objetos, ya que en la práctica al momento de modelar un automóvil se debía de pensar en sus características y cómo estas características iban a ser manejadas por otra clase llamada mecánico el cual es el intermediario al momento de reparar el automóvil, y esto se debió a como consideramos iba a interactuar el mecánico, y se puede ver claramente las relaciones que tienen las clases gracias al diagrama UML.

* Díaz Hernández Marcos Bryan:

En esta práctica logré comprender la abstracción que pueden las clases, y cómo esta afecta al procesamiento del programa, además de poder implementar el UML como medio de análisis de las clases y llevar a cabo la composición entre varias clases, ya que el desarrollo del programa requería utilizar estos conceptos, por lo que puedo concluir que se cumplió el objetivo de la práctica.

* Pareja Ávila Emiliano:

Se cumplieron los objetivos de la práctica, ya que a partir de la abstracción pude modelar algunas clases del programa, pero no solo eso, también algunos métodos que se componen de otros métodos. Gracias a la abstracción puedo llegar a modelar varias clases y así crear objetos que se componen de otros objetos, lo cual para el Paradigma orientado a Objetos es fundamental. También con la parte de la introducción sobre los diagramas UML, comprendo la composición en el Paradigma y como realizar estos diagramas.

* Vázquez Zavala Oliver Alex:

Con el desarrollo de esta práctica he podido captar bastante mejor los conceptos de abstracción y encapsulamiento en el paradigma orientado a objetos, así como su implementación en Java, aspectos que son muy importantes que tenga presentes al momento de desarrollar programas bajo este paradigma, creando clases en donde tome en cuenta sólo los aspectos más relevantes de una cierta entidad, protegiendo y manejando de forma correcta la información de dicha entidad, entendiendo todo esto gracias al ejercicio desarrollado y el uso que se tiene de los modificadores de acceso, métodos getters, setters y la composición en el programa.

**Bibliografía**

* James R. et al. (2000). El lenguaje unificado de modelado. Madrid: Pearson.