|  |  |  |
| --- | --- | --- |
|  | **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):* | 4 |
| *Integrante(s):* | Dávila Ortega Jesús Eduardo - No. Cuenta: 317199860  Díaz Hernández Marcos Bryan - No. Cuenta: 317027253  Pareja Avila Emiliano - No. Cuenta: 317081345  Vazquez 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:* | 31 de octubre del 2020 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_

**Objetivo de la práctica.**

Aplicar los conceptos básicos de la programación orientada a objetos en un lenguaje de programación.

**Introducción.**

Una de las principales características del lenguaje de programación Java es ser un lenguaje fuertemente tipado, esto significa que para cada variable que se declare será necesario indicar el tipo al que corresponde y la información que se almacenará en dicha variable deberá ser compatible con el tipo de dato indicado durante toda la ejecución, en ocasiones es necesario realizar una conversión de un tipo de dato definido a otro y para esto se puede realizar lo que se conoce como casteo, el casting o casteo es un procedimiento para transformar una variable primitiva de un tipo a otro, existen dos tipos de conversiones de tipo o casting en Java: conversión implícita y conversión explícita. En el caso del casting implícito, automáticamente podemos asignar una variable de un cierto tipo de dato, a otra variable de otro tipo de dato distinto, sin la necesidad de escribir código para que la conversión se lleve a cabo, más allá de la asignación en la nueva variable, ocurre cuando se realiza lo que se llama una conversión ancha (widening casting), lo que hace es colocar un valor pequeño en un contenedor grande, es decir cuando se hace la conversión de un valor que ocupa una cierta cantidad de bits en memoria a otro que ocupa más cantidad de bits en memoria o cuyo rango de valores es mayor, un ejemplo de esto es el siguiente fragmento de código:

* *byte x = 5;*
* *int y = x;*

En donde el valor almacenado en x (en este caso 5), el cual es del tipo byte y ocupa 8 bits, se convertirá int que ocupa 32 bits, dicha conversión se realizará al asignarle a y el valor de x.

En el caso del casting explícito sí es necesario escribir más código. Ocurre cuando se realiza una conversión estrecha (narrowing casting), es decir cuando se convierte un valor a otro más pequeño, en lo que a nivel de bits o rango de valores respecta, la sintaxis para realizar este tipo de conversión es: (tipo)valorAConvertir, donde el tipo es el tipo al que se quiere convertir, y valorAConvertir, el valor desde que se quiere convertir, un ejemplo de lo anterior es el siguiente fragmento de código:

* *int num1 = 100;*
* *short num2 = (short) num1;*

En donde se realiza un casteo del tipo de dato int al tipo de dato short, sin embargo hay que tener precaución al realizar esta conversión ya que se puede aplicar a tipos no compatibles, lo que puede derivar en pérdidas de información e incluso errores en ejecución, por ejemplo tomando como base el ejemplo anterior si se sustituye la primera línea int num1=100 por int num1=1000000, teniendo lo siguiente:

* *int num1 = 1000000;*
* *short num2 = (short) num1;*

En este caso el código compilaría bien, pero habría pérdida de datos, pues el 1000000 se sale del rango de short, que comprende desde -32768 a 32767, y al mostrar en pantalla el valor de num2 se obtendría 16960, esto debido a que el contenido de la variable num 1 (32 bits) es: 00000000 00001111 01000010 01000000, cuyo valor es 1000000 y al hacer un casting al tipo short (16 bits) se están tomando los 16 bits menos significativos es decir: 01000010 01000000, cuyo valor es 1960.

Otro ejemplo donde puede presentarse la pérdida de información al hacer casteo entre distintos tipos de datos es:

* *float x = 5.3F;*
* *int y = (int) x;*

Donde se realiza la conversión de float a int, pero en este caso se perderá la parte decimal del número con punto flotante al guardarlo en y, es decir solo se guardará 5 en y, similar a cuando se trunca un número con parte decimal.

Cuando se crean instancias en java, es común inicializar sus atributos de acuerdo al tipo de dato al que corresponden, una opción para realizar esto es mediante el operador punto por ejemplo, para un objeto de la clase Persona llamado persona1, que cuenta con el atributo entero edad, dicha información puede ser inicializada como: persona1.edad=10, pero esta información puede ser inicializada a la vez que se crea la instancia, esto con la ayuda de un método constructor. Un constructor es un elemento de una clase cuyo identificador coincide con el nombre de la clase correspondiente y que tiene por objetivo inicializar a un objeto cuando se crea, todas las clases tienen constructores, ya sea que uno mismo defina alguno o no, porque Java proporciona automáticamente un constructor predeterminado, en este caso al crear un objeto con el constructor por defecto los atributos no inicializados tomarán sus valores predeterminados, que son cero, null y false, para tipos numéricos, tipos de referencia y booleanos, respectivamente. Además pueden existir varios constructores en una misma clase (sobrecarga de constructores), diferenciándose entre sí por el número y tipo de parámetros con los que están definidos.

Cuando existe una jerarquía de clases, al momento de instanciar un objeto de la clase o clases hijas se invocan los métodos constructores de las clases padres, iniciando por la clase de mayor jerarquía hasta la de menor jerarquía. En java la palabra reservada ‘this’ se utiliza generalmente para invocar atributos dentro de la misma clase donde es empleada cuando existe ambigüedad entre atributos y los parámetros de algún método (ocultamiento), de igual manera cuando en una clase se tienen varios constructores podemos usar this() para desde un determinado constructor poder hacer uso de otro constructor distinto definido en la clase, para llamar a otro constructor con this() siempre hay que hacerlo desde la primera línea del ámbito del constructor del cual se hace la llamada, de lo contrario se tendrá un error al momento de compilar, y de igual manera que cuando se llama a un método constructor en la creación de una instancia, dependiendo del número y tipo de argumentos proporcionados en this() se llama al constructor correspondiente en la clase, por ejemplo si en un constructor que recibe dos argumentos de tipo int se tiene en la primera línea *“this();”*, se dice que dicho constructor está llamando al constructor sin parámetros, el cual debe estar definido en la clase para evitar tener un error al momento de compilar el código.

**Ejercicio 1**

* **Análisis previo**

Para desarrollar la clase de Mascota, primero tuvimos que buscar los elementos básicos de una mascota, por ello nos basamos en los atributos que tiene un pou, de manera que desarrollamos una versión que tuviera atributos modificables por cada método.

Como se menciona los métodos tenían que modificar los atributos de nuestra mascota, por lo que se crearon varios de estos, que modelan diferentes situaciones dentro de la ejecución del programa, por ejemplo está la acción de jugar como un método, en el cual se tiene que los valores de energía, hambre, sed, se incrementan o disminuyen por la ejecución de este método.

De forma equivalente cada método termina modificando los atributos de forma que cada llamado de estos reduzca en una determinada cantidad las proporciones de los valores, y otros métodos, puedan volverlas a un estado de normalidad, es decir que si por ejemplo la energía llega a cero el método dormir llene de nuevo la energía y la mascota pueda jugar de nuevo o pueda salir a pasear.

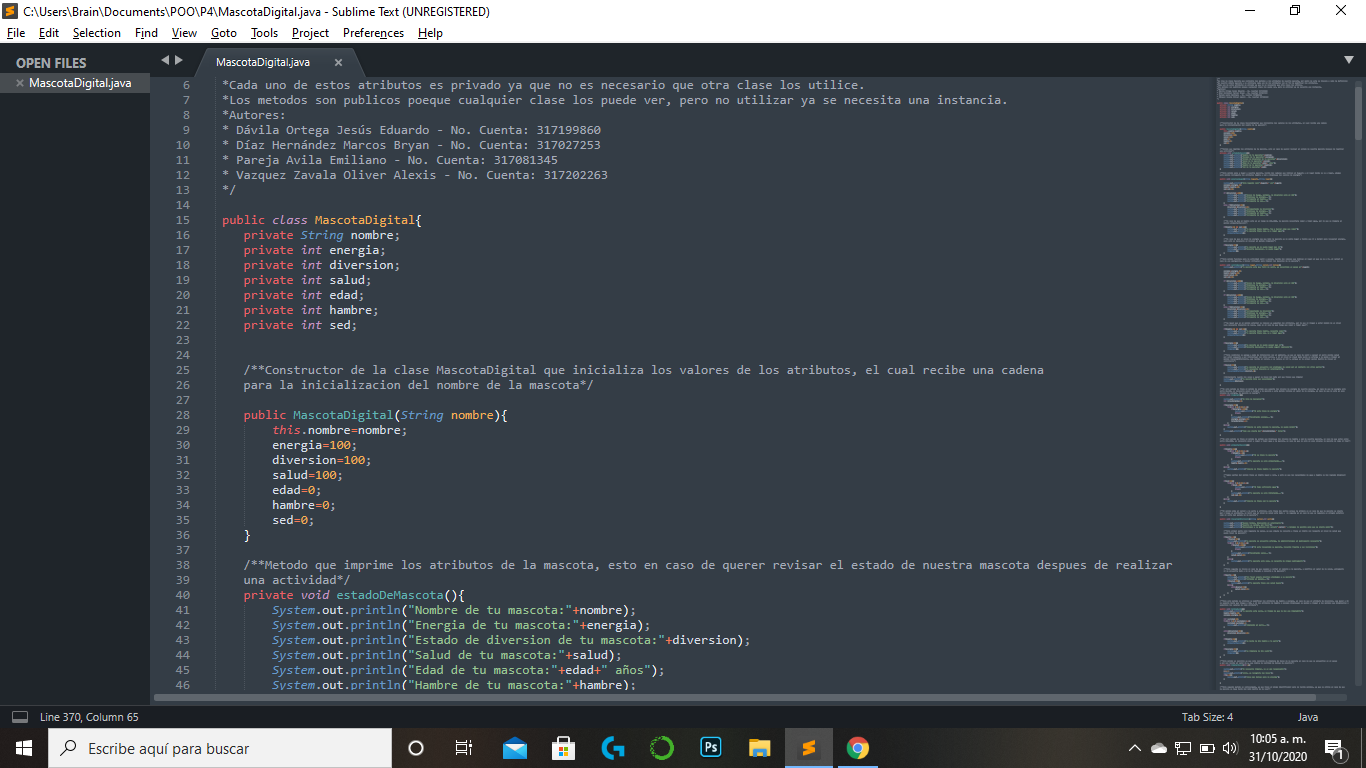
Igual se hicieron métodos con respecto a cada uno de los atributos, en este caso se tiene que cuando la mascota se siente mal, es necesario llevarla al veterinario, en este caso un veterinario virtual y que se diagnostique si está bien o no. Otro ejemplo es cuando sale de paseo, este método está más basado en la acción de jugar con la mascota pero en un entorno fuera de casa, siendo este método el modificador de la salud, la energía, la diversión.

El análisis previo fue el ver qué tipo de método tendrían que relacionarse con los atributos y como los modificarían, esto con un toque de realismo, pero sin olvidar que la mascota puede revivir y envejecer, ya que es virtual, simplemente no se le aplican métodos que no estén relacionados con los atributos.

* **Desarrollo**

Lo primero que se desarrolló en el código fue la clase, comenzando por los atributos, los cuales tendrían que ser privados, y al estar desarrollado el programa dentro de una misma clase, no hay problema con el acceso a estos, lo siguiente fue el crear un constructor el cual inicializa los valores de nuestros atributos, ya que es una mascota virtual esta tiene atributos definidos desde el momento en que se crea, para esto nos basamos en el pou.

Los atributos se inicializan con un valor de 100 en el caso de la energía, de la salud, y con cero en el caso del hambre, sed, y edad. La edad se incluye como un avance en la experiencia de la mascota, simulando el crecimiento en años de una mascota normal.



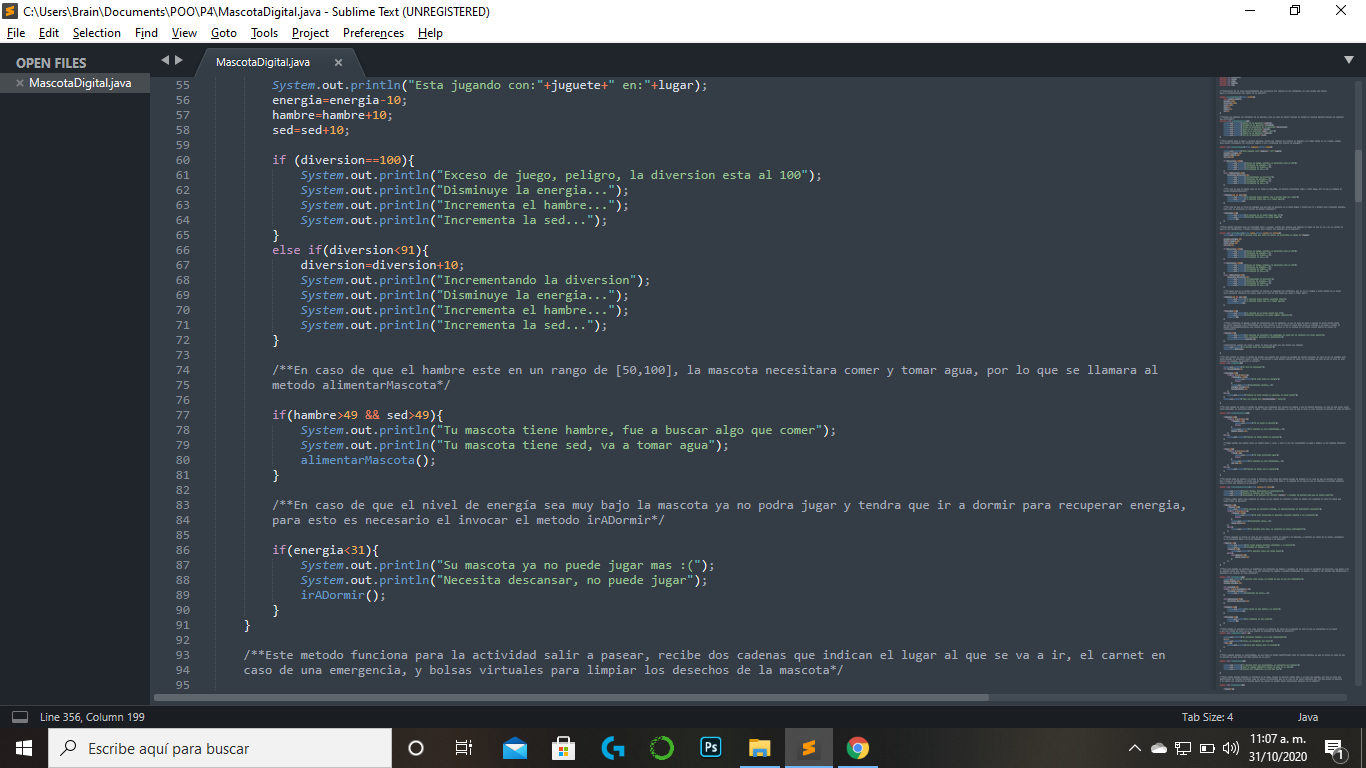
El primer método que se define es impresor del estado de la mascota, es decir, que va a imprimir los valores de los atributos, esto es para poder verificar el estado de la mascota después de cada acción, funciona como una consulta similar a la que se encuentra en los juegos, por el hecho de que los valores de los atributos de la mascota cambian tras ciertas acciones.

De igual forma dentro de los mismos métodos existen condiciones que en caso de que un atributo como la energía se encuentren debajo de cierto nivel no se puedan realizar más acciones y en automático se manda a dormir a la mascota, de igual forma los métodos se pueden aplicar individualmente como el de mandar a dormir, o comer, pero si el atributo hambre o energía de la mascota se encuentra por encima de un nivel, la mascota no dormirá o no comerá, debido a que no se encuentra con los niveles necesarios para que lo haga, y en ese caso se tiene la consulta para verificar los valores.

Los siguientes dos métodos son modificadores de los atributos, jugar, salir a pasear, e ir al veterinario. Los dos primero modifican los atributos de energía, diversión, hambre, salud, sed. Los cuales disminuyen de 10 en diez o incrementan en la misma proporción, y para el paseo se tiene que los atributos cambian de valor de 30 en 30, cada vez que se ejecuta el método.

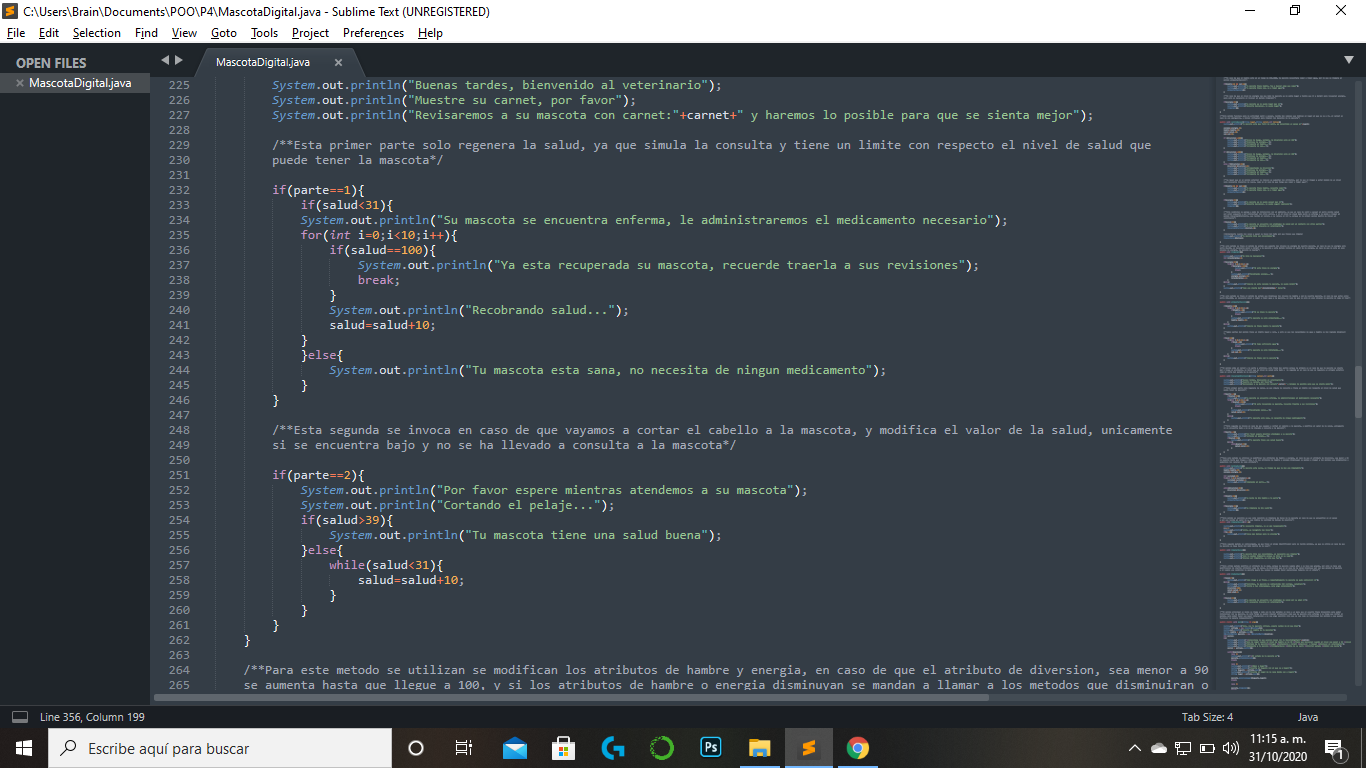
Se colocan if, con la intención de que los valores no se salgan de un rango de [0,100], dando como límite esos valores y en caso de que los valores entren en rangos de [0,30] para la energía, y la salud, se mandan los métodos que llenan estos valores, por medio de una visita al médico o durmiendo, estos métodos llenan por completo el atributo y permiten que se siga ejecutando el código.

Para el caso de hambre y sed, se tiene que si los valores están en un rango de [50,100], se mandan a llamar a los métodos que vacían los atributos de hambre o de sed, y para la diversión se tiene que al jugar esta incrementa pero no se modifica en otros métodos. Por ello se tiene que la diversión no varía mucho en nuestra mascota.



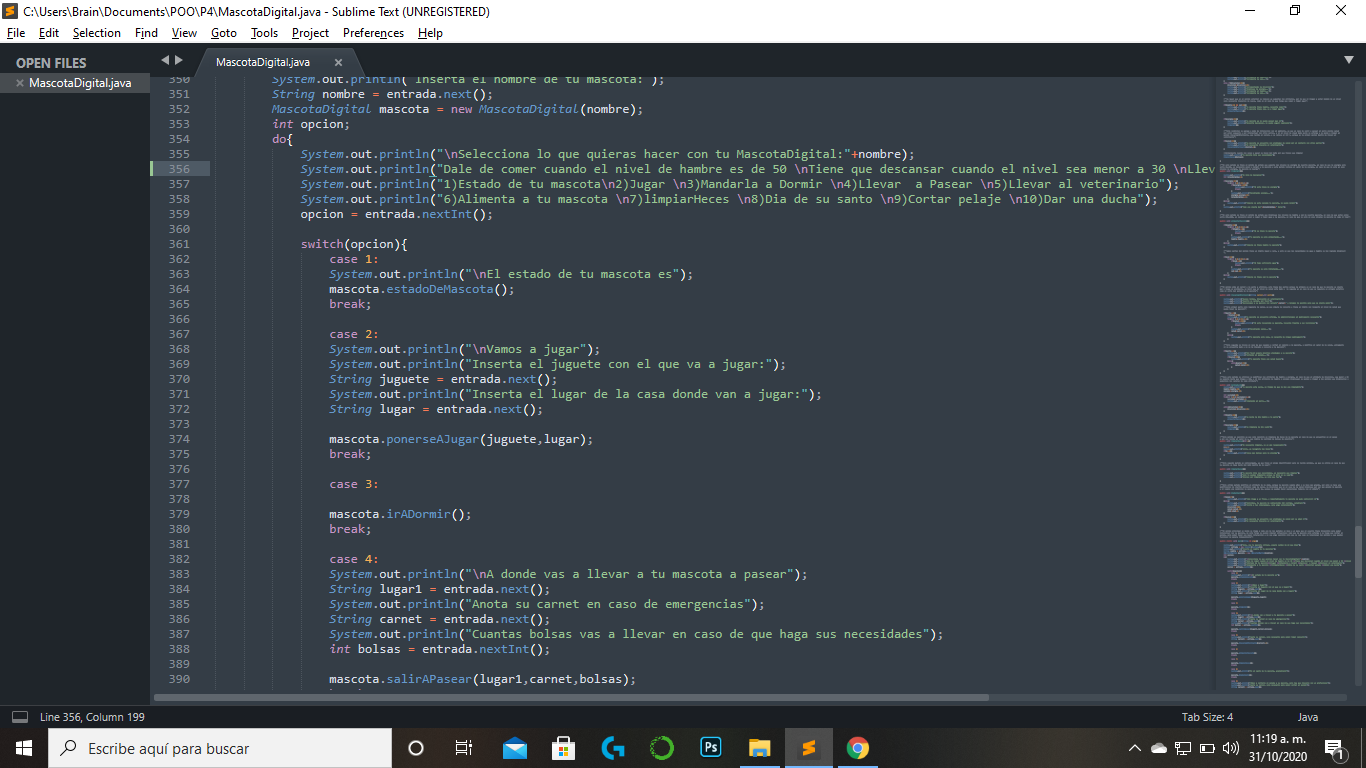
Los métodos que llenan los atributos se caracterizan por ir llenando de con un ciclo for el atributo y en cada iteración ir agregando 10, con el límite de 100, para la energía y para la salud, en el caso del hambre y sed se tiene que el límite es 0 y se van restando de 10 unidades, porque el hambre funciona de forma inversa a lo que es la energía. En estos métodos de ir a Dormir y alimentar se tiene la condición que se mencionaba antes, que si el valor del atributo no está dentro del rango, no llena o vacía el atributo, solo indica que el nivel o el valor de ese atributo no es adecuado para que la mascota duerma o se alimente.

El método de llevar al veterinario funciona como un método que restablece el valor de salud, en caso de que la mascota esté bajo de esta, y también funciona para cortar el pelaje a nuestra mascota, este método hace referencia a la característica de poder personalizar nuestra mascota pero con un toque de realidad. Y este método modifica la salud únicamente.

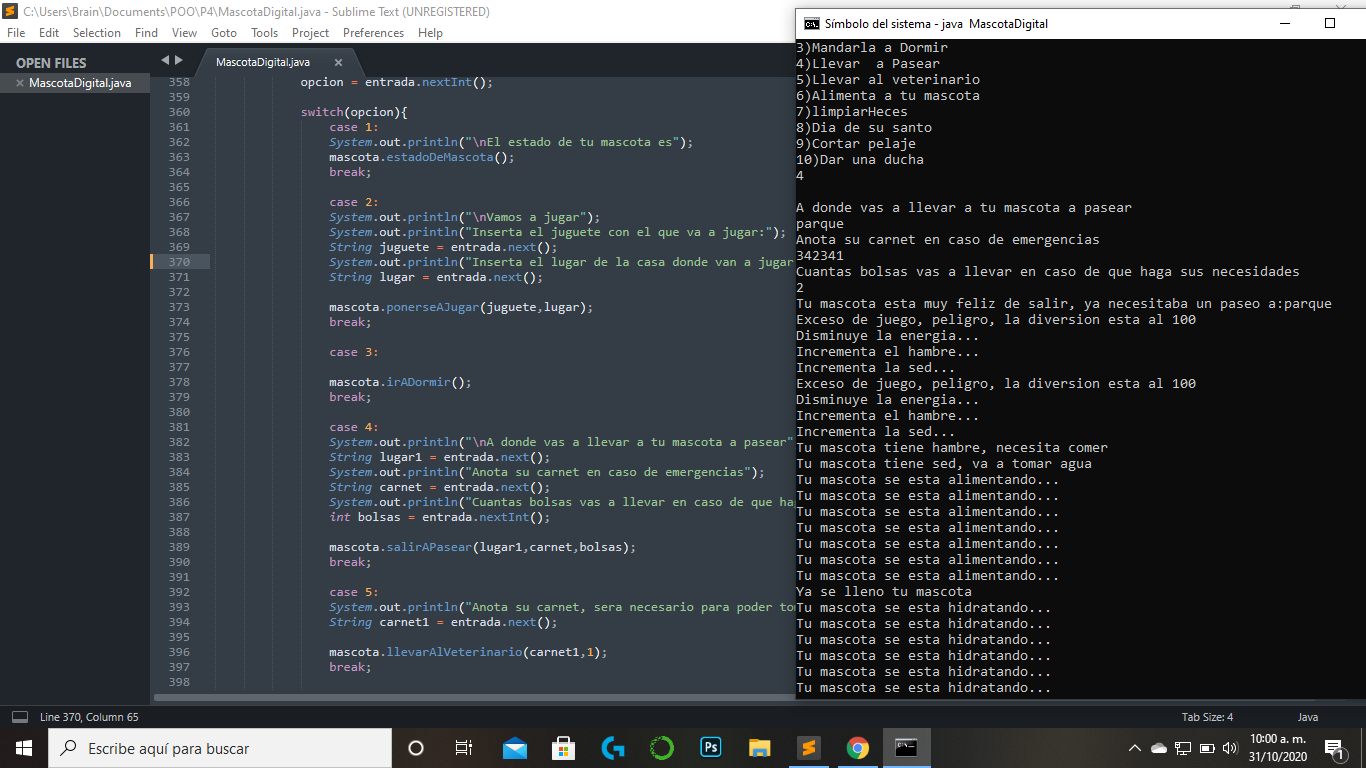


Los demás métodos como bañar al perro, limpiar heces, y cumpleaños, no modifican los atributos, excepto el primero que modifica hambre y energía, y para esto se aplican las mismas condiciones del if, para que cuando se encuentren los atributos bajo cierto nivel o rango, se llenan o vacían.

Como última parte se creó un menú, con un do-while, para la reiteración de los valores y los métodos que se pueden aplicar a la mascota, además de unas pequeñas instrucciones que le dan un detalle interesante a la interacción con el usuario.

****

* **Ejecución del programa**

****

**Ejercicio 2**

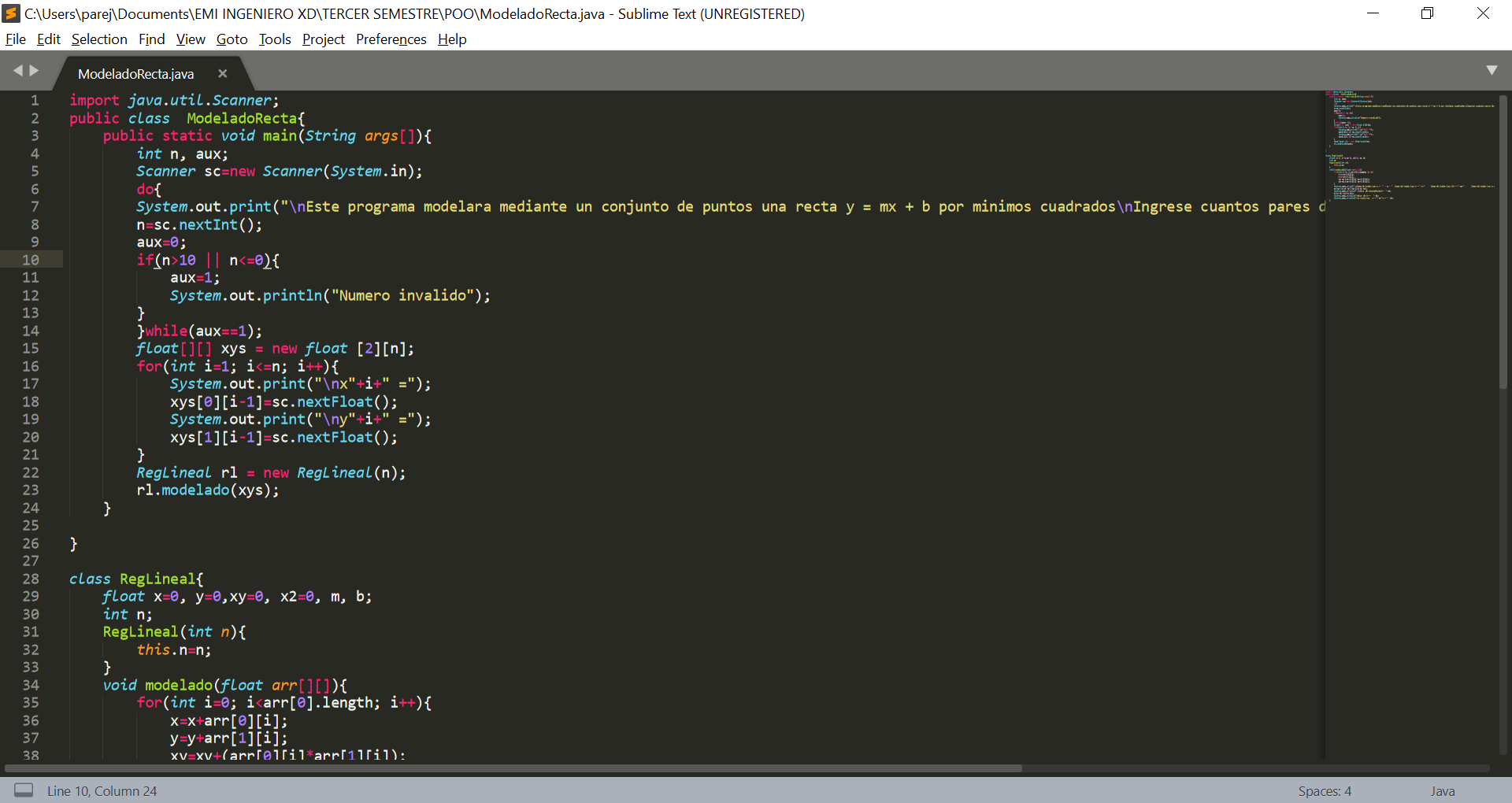
* **Análisis previo.**

Para este ejercicio fue necesario, primero investigar cómo hacer el modelado de una recta a partir de un conjunto de pares ordenados (x,y) dados, en nuestras clases de primeros semestres en materias como Fundamentos de Física o Química, en base a ciertos resultados que obteníamos en los laboratorios, se nos pedía que a partir de esos datos hiciéramos gráficas lineales sobre cómo se comportaba una variable respecto a otra, y para hacer estos modelos lineales, los profesores nos pedían que utilizáramos el método de mínimos cuadrados. Entonces recurrimos a las prácticas en donde se nos pedían hacer estos modelados lineales y observamos estas fórmulas:

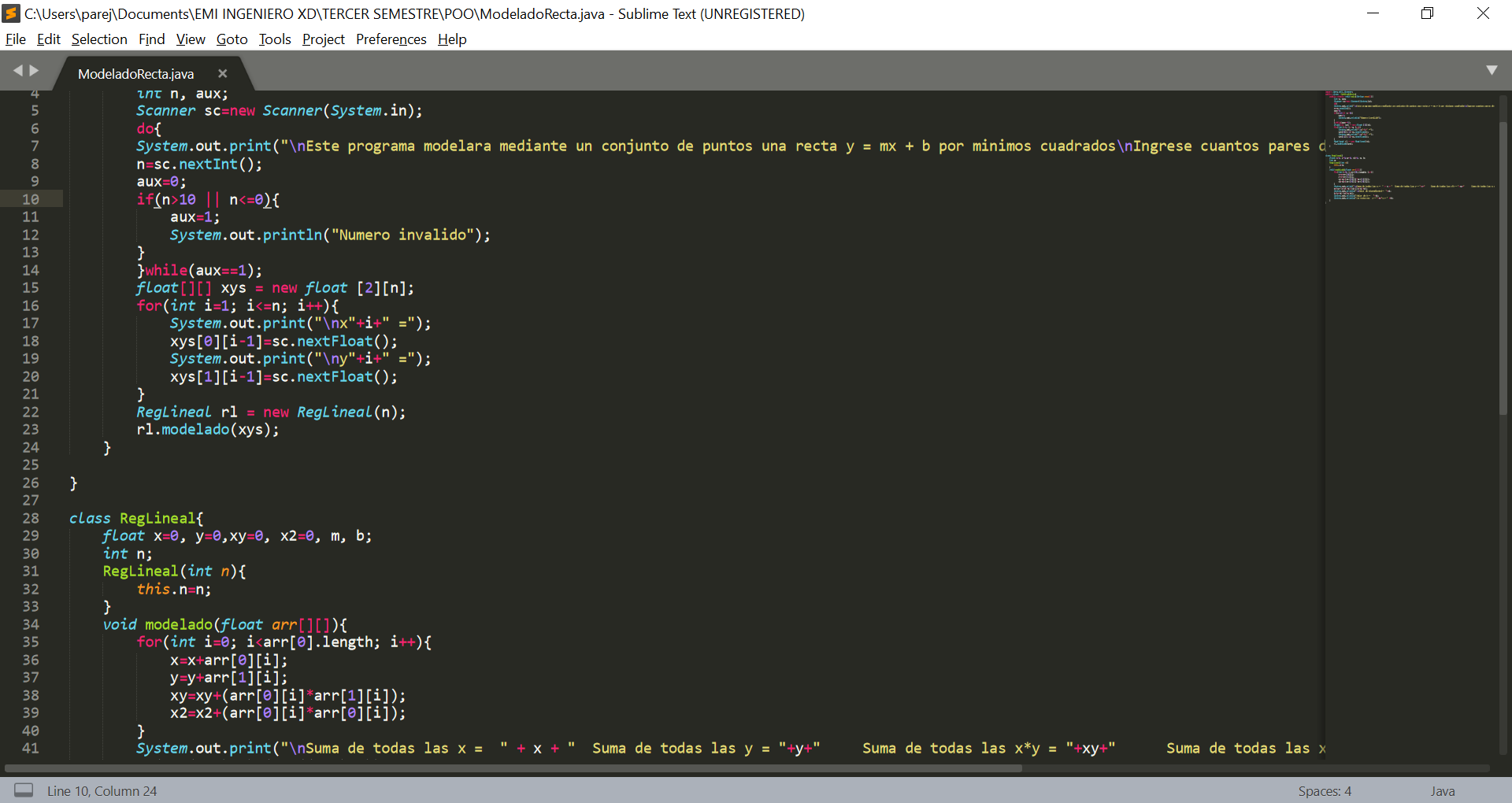
Entonces, sabiendo que una recta es igual a y= mx +b donde “m” es la pendiente de la recta y “b” es la coordenada donde la recta corta el eje y, ya tenemos la fórmula.

Pero también hay que tomar en cuenta las restricciones que se nos dieron como indicaciones en la práctica, en la cual el usuario nos tiene que dar por lo menos 1 par de puntos para hacer el modelado y también no más de 10 puntos. Entonces estableceríamos ciertas restricciones también en nuestro programa en donde el usuario no ingresará menos de 1 par ordenado, ni más de 10. También para guardar todos los pares ordenados (x,y), crearíamos un arreglo multidimensional de 2xnpares de puntos ya que en un arreglo guardaríamos todas las xs y en otro todas las ys. También en base a las especificaciones para la práctica, debíamos crear una clase que recibiera todo este arreglo multidimensional y dentro de él, con sus métodos y atributos calcular la ecuación de la recta.

* **Desarrollo**

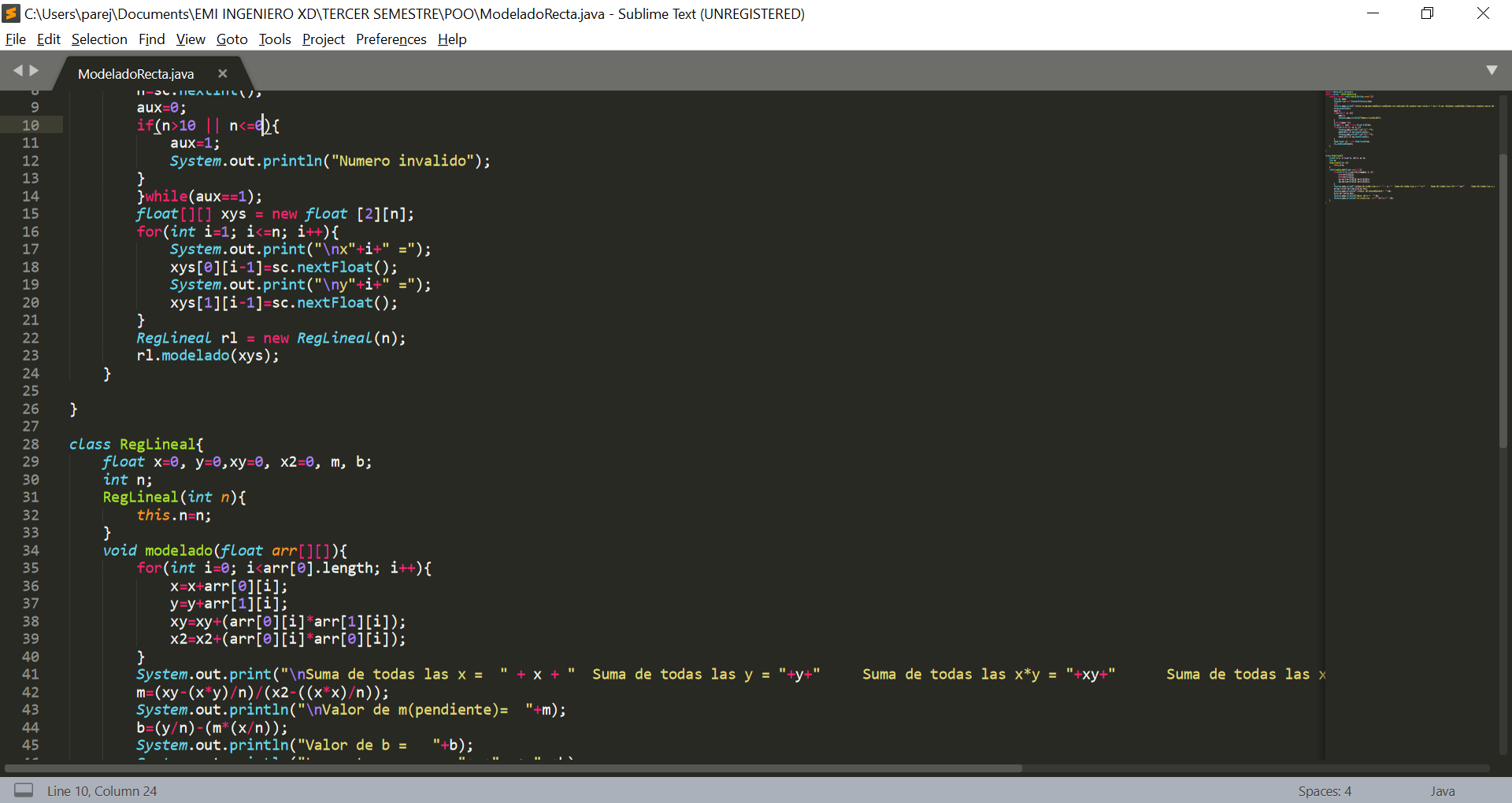
**En la clase principal:**

Primero, importamos la utilería Scanner para hacer las lecturas de las xs y ys, después ya creada nuestra clase de prueba y nuestro método main, creamos 2 variables enteras, una llamada n, que recibe el número de pares de puntos que nos ayudarán a modelar la recta, y otra llamada aux, que nos ayudaría en caso de que el usuario ingresará un valor inválido de números de puntos. Después de crear nuestro objeto sc de la clase Scanner para hacer las lecturas, en una estructura de repetición do-while, ingresamos el siguiente bloque de instrucciones, primero indicamos al usuario lo que hace nuestro programa y que indique cuántos pares de puntos tiene, después, con ayuda del objeto de la clase Scanner leemos el valor ingresado por el usuario y este valor se guarda en la variable n,, e igualamos la variable aux a 0, y después en una estructura de control de flujo if, en la condición se verifica que si n es mayor de 10 o menor o igual a 10, aux valdrá 1 e imprime la pantalla que el valor ingresado es inválido. Este ciclo se repetirá mientras aux sea igual a 1, o sea que mientras se haya ingresado un valor inválido el ciclo se repetirá. Si el valor entra en un rango de 1 a 10, entonces se crea un arreglo bidimensional de floats llamado xys, de 2xn, y luego en un ciclo for, se hacen todas lecturas de todos los pares (x,y)., donde el índice 0 de xys, guardará todas las x y el índice 1 todas las ys.



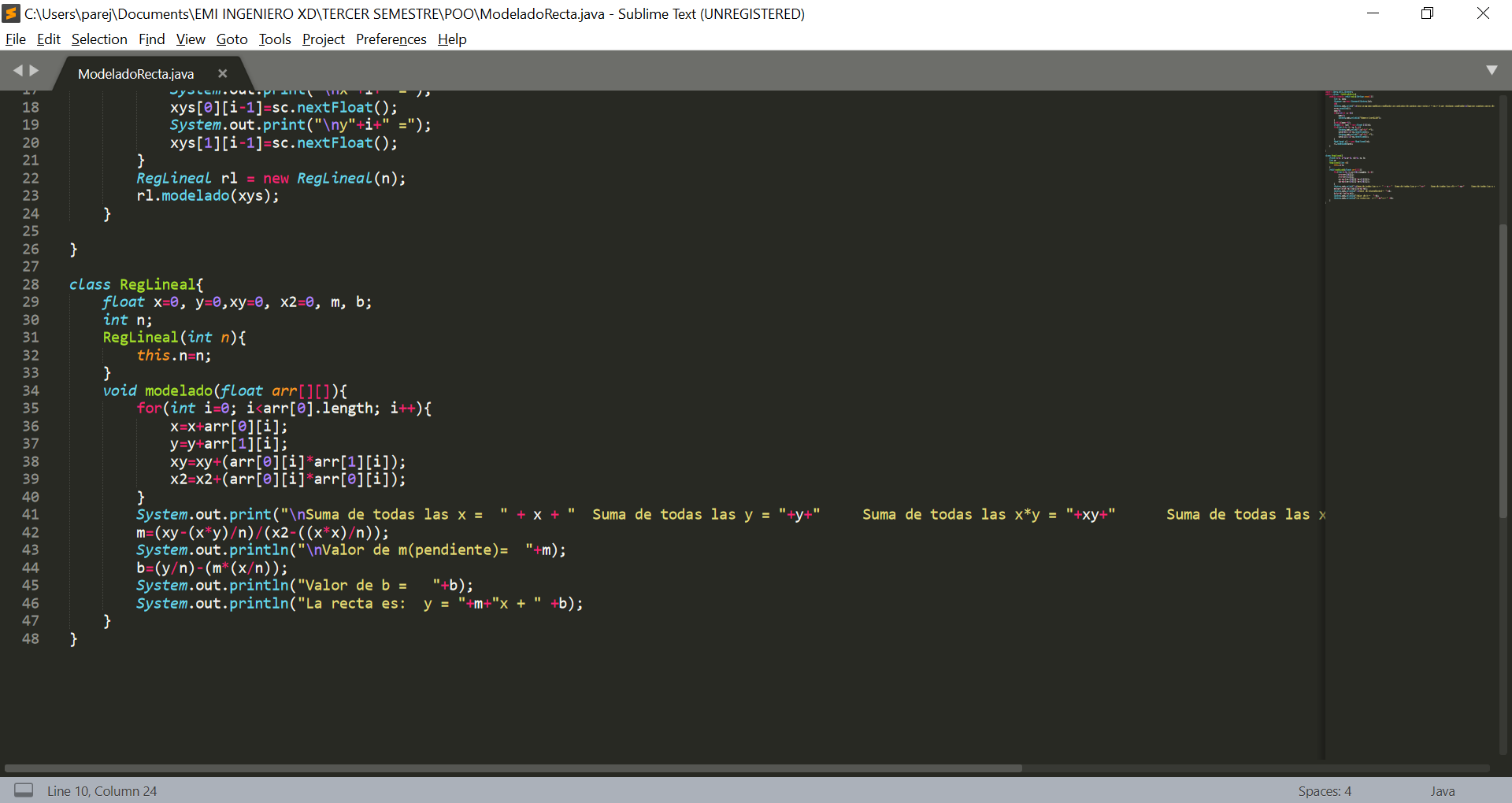
Después, creamos un objeto de la clase RegLineal, llamado rl y mediante su método constructor le asignamos como parámetro la n o el número de pares de puntos. Y finalmente del objeto rl, se hace una llamada a su método llamado modelado y se le asigna como parámetro el arreglo bidimensional de xs y ys. Y este método de la clase RegLineal, nos ayudará a modelar la recta que necesitamos.

**En la clase RegLineal:**

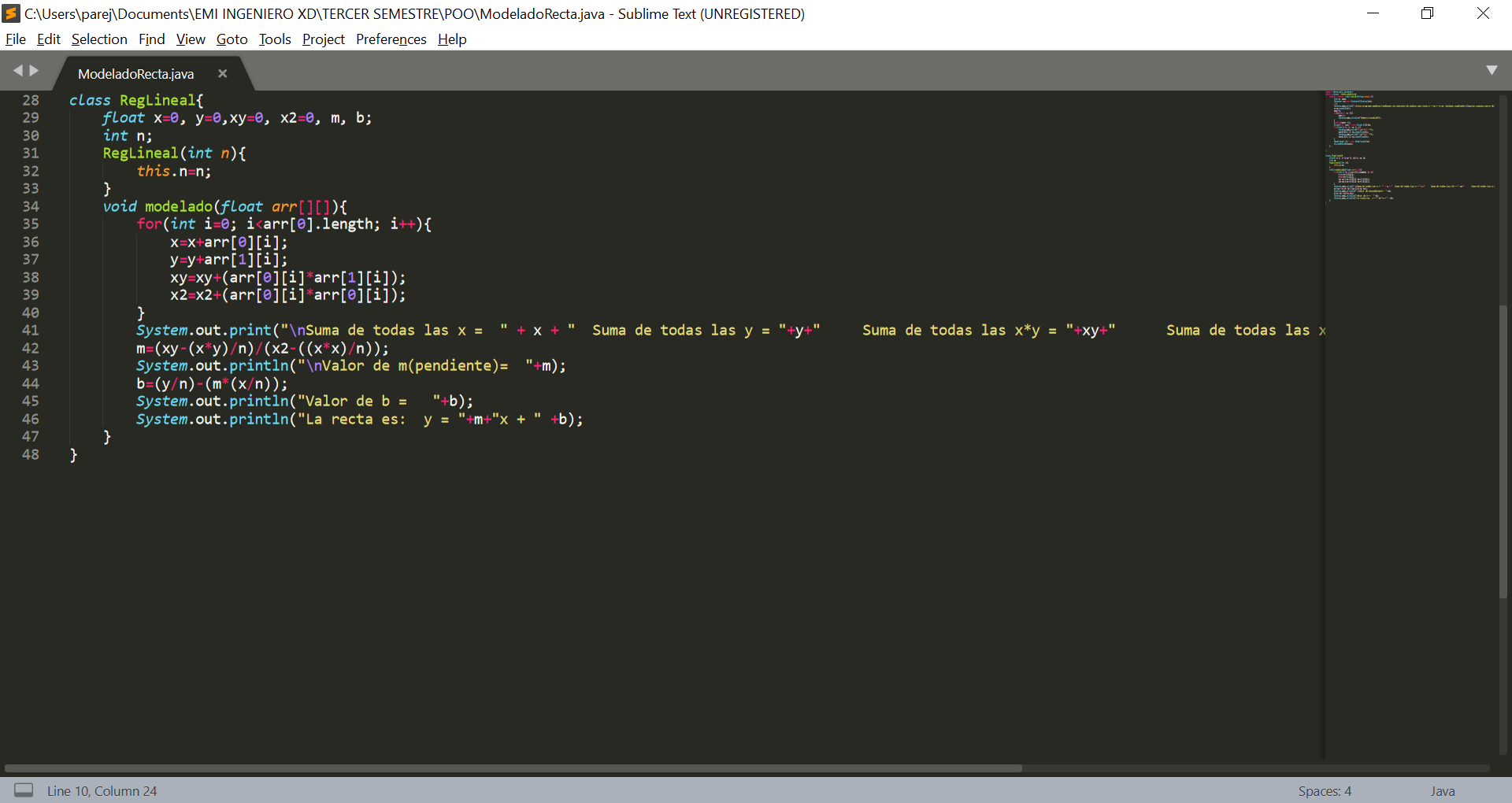


Dentro de la clase RegLineal definimos 7 atributos, 6 de ellos de tipo float y uno de tipo entero, el primero llamado x, almacenará dentro de ella la suma total de todas las x de todos los pares de puntos, y almacena la suma total de todas las ys de todos los pares de puntos, xy almacena la suma total de los productos de todos los pares ordenados (x,y), x2 almacena la suma total de todas las xs elevadas al cuadrado de todos los pares de puntos, cabe resaltar que de la variable x a la x2, todos los atributos están igualados a 0, ya que al momento de hacer la suma total se necesita un valor inicial de la variable., m será el valor de la pendiente de la recta y b será la intersección con el eje y de la recta.

Finalmente, n es el número de pares ordenados con los que contaremos para modelar la recta. Luego para cumplir con las especificaciones de la práctica, definimos un método constructor que recibe un parámetro entero llamado igual que el atributo n de la clase, entonces aquí provocamos el ocultamiento, pero para deshacerlo,con ayuda de la palabra reservada this, lo aplicamos al atributo n de la clase RegLineal. Entonces el atributo n del objeto, tomará el valor de la n que se le está pasando como parámetro al método constructor.



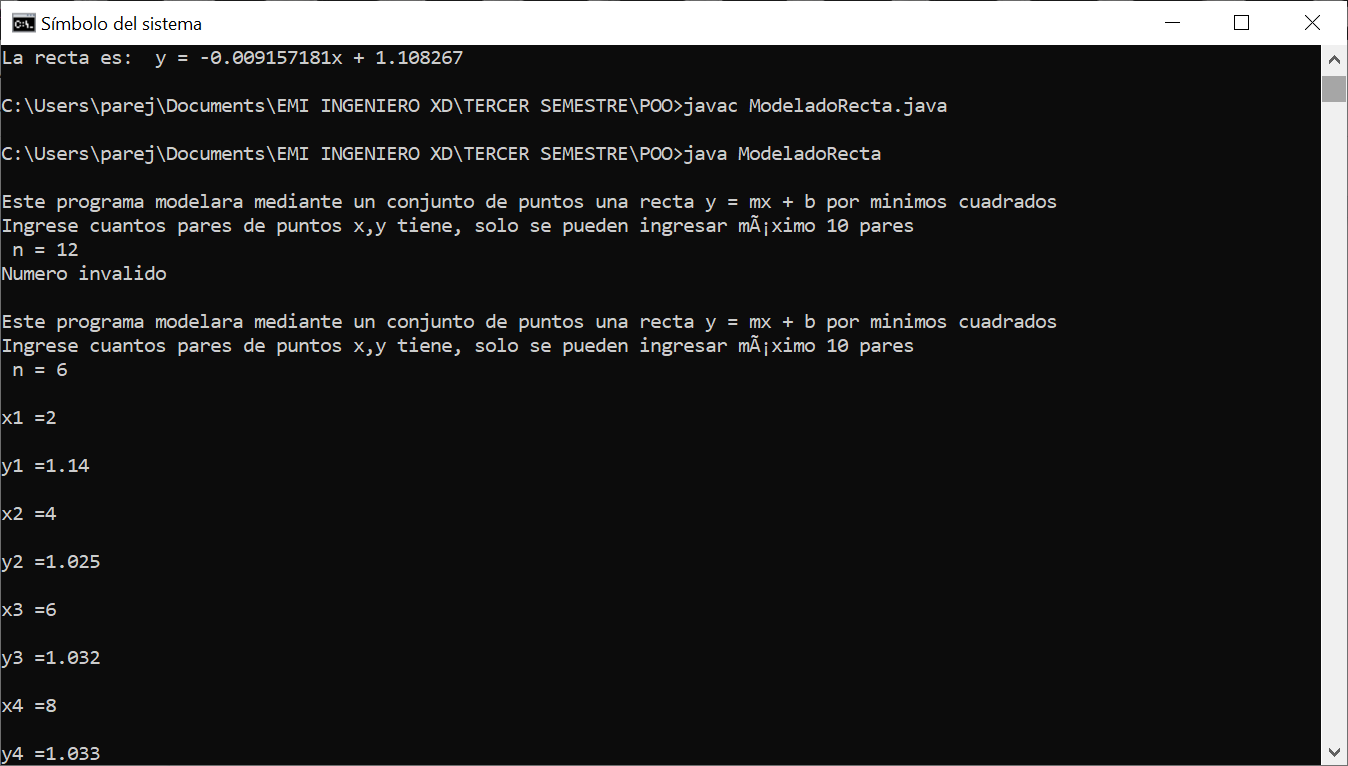
Después, dentro de la misma clase RegLineal, definimos el método que nos ayudará a modelar la recta a partir de un conjunto de pares ordenados, este método es de tipo void, se llama modelado y recibe solo un parámetro que es un arreglo multidimensional de floats, que en este caso será el arreglo que incluye todas las xs y ys leídas en la clase principal. Luego, en un ciclo for hacemos todas las sumas totales, primero la suma total de todas las x que se almacena en el atributo x, después la suma total de todas las ys, que se almacenan en y, enseguida la suma total de todos los productos xy y esta suma se almacenará en xy, y por último la suma de todas las xs de todos los pares de puntos elevadas al cuadrado, cuyo valor final se almacena en el atributo x2.

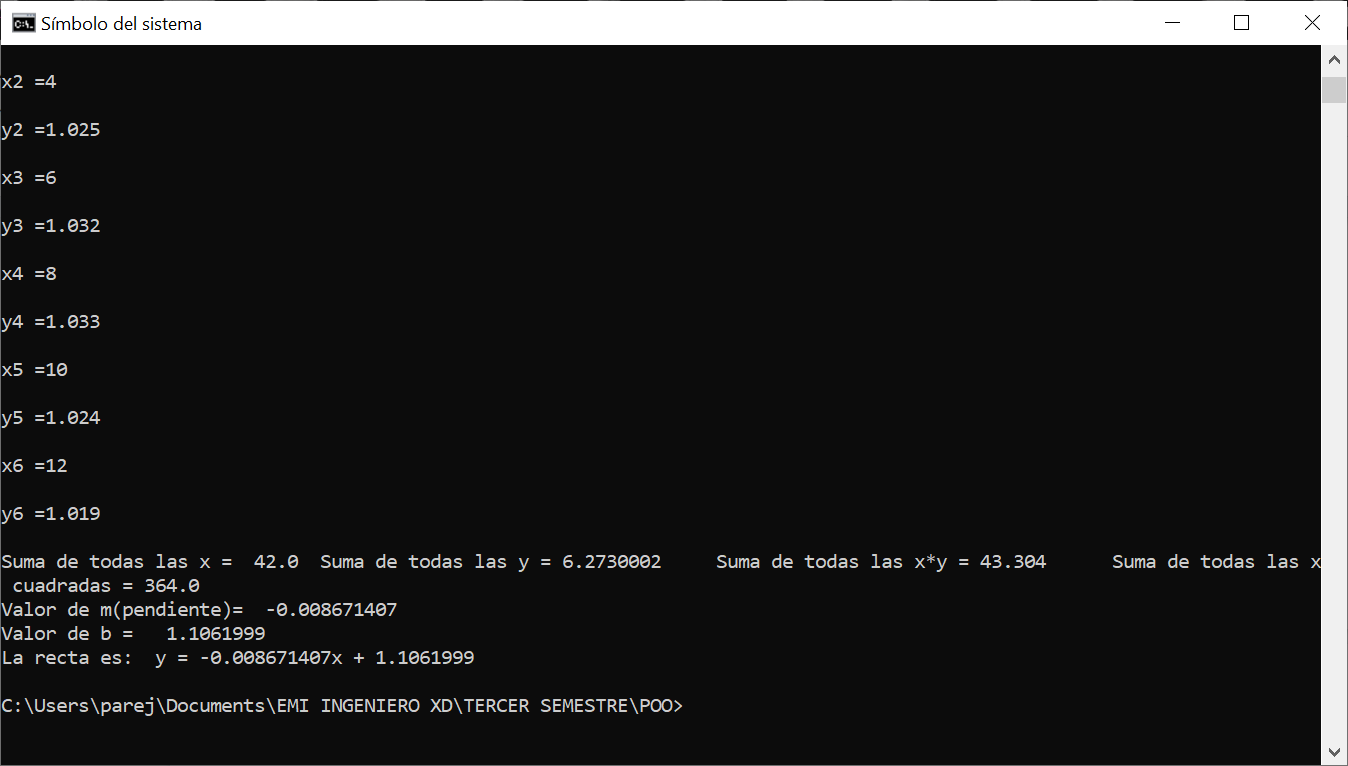


Ya que se calcularon todas las sumas totales, para cumplir con los requisitos de la práctica, indicamos el valor de los atributos del objeto de la clase RegLineal. Después ya aplicando las fórmulas del método de Mínimos Cuadrados, calculamos la pendiente de la recta, que es igual a la diferencia de la suma total del producto de todas las xs y ys de todos los pares ordenados, menos el producto de la suma de todas las xs por la suma total de todas las ys, entre el número total de pares ordenados, todo el valor anterior se divide entre la suma total de todas las xs elevadas al cuadrado, menos la suma total de las xs, elevada al cuadrado entre el número de pares ordenados, y todo este valor se almacena en el atributo m.

Luego se imprime a pantalla el valor de la pendiente y para terminar todo el proceso del modelado de una recta por el método de mínimos cuadrados, se calcula la intersección de la recta con el eje y, y esta es igual a la suma de todas las ys entre el número total de pares ordenados, menos el producto de la pendiente por la suma total de todas las xs entre el número total de pares ordenados, el valor que se obtenga se guarda en la variable b y se imprime a pantalla. Y para que termine el programa se imprime la ecuación de la recta modelada a partir de un conjunto de pares ordenados.

* **Ejecución del programa**





**Conclusiones**

* Dávila Ortega Jesús Eduardo:

Durante la realización de la práctica se pudo poner en práctica un poco el paradigma orientado a objetos y algunas de las características de este mismo, ya que la realización de algunos de los programas te motivaban a pensar en cómo modelar las cosas esto se ve sobre todo en el programa de la mascota, esto es así debido a que tenemos que pensar en las características que hacen que una mascota se a una y sus comportamientos o cosas que podemos hacer que haga la mascota, y esto se ve facilitado en el paradigma orientado a objetos gracias a todo lo que ofrece este tipo de paradigma.

* Díaz Hernández Marcos Bryan:

En esta práctica se han implementado los conceptos fundamentales del paradigma, por medio de distintas clases, métodos, objetos, los cuales modelan una mascota y la regresión lineal, mostrando que es posible modelar entes u objetos, mediante métodos que modifican los atributos de cada uno de estos. Con el propósito de comprender los elementos más básicos de la programación orientada, creamos los programas con las condiciones indicadas y con estos logramos cumplir el objetivo de implementar las partes esenciales de la programación orientada a objetos en Java.

* Pareja Ávila Emiliano:

Se cumplieron los objetivos de la práctica, ya que implementé en un programa, todo el paradigma orientado a objetos, con la creación de clases, abstracción, ocultamiento, así como definir dentro de las propias clases atributos y métodos que pueden tener algún valor de retorno o no. También, a partir de la creación de clases, se cómo instanciar objetos de esas clases en otra, también, puedo sobrescribir o sobrecargar el método constructor de un objeto asignando parámetros, puedo evitar el ocultamiento de un atributo con ayuda de la palabra reservada this y se acceder a los atributos y métodos de un objeto en base a sus modificadores de acceso. Entonces, con todos estos aspectos, puedo aplicar el paradigma orientado a objetos en cualquier programa.

* Vázquez Zavala Oliver Alexis:

Los objetos son la unidad fundamental dentro del paradigma orientado a objetos, los cuales contienen una serie de datos y tareas determinadas y esta práctica me ha ayudado a poder visualizar y utilizar este y otros de los conceptos básicos de la programación orientada a objetos en java, tales como clases, atributos, métodos, sobrecarga, entre otros, en el desarrollo de los ejercicios se implementó la creación y el uso de clases, se crearon instancias, se usó la sobrecarga de métodos, se ocupó y modificó el método constructor y se estudió el fenómeno de ocultamiento y cómo solucionarlo, todo esto de manera clara, aspectos que es muy importantes tener claros al momento de desarrollar un programa bajo el paradigma orientado a objetos.

**Bibliografía**

Oracle (s.f) “Chapter 5 Coversions and Promotions” <https://docs.oracle.com/javase/specs/jls/se7/html/jls-5.html#jls-5.1>

<https://sites.google.com/site/pro012iessanandres/java/conversion-entre-tipos-primitivos-casting>

-<https://programandoointentandolo.com/2018/10/this-y-super-java.html#:~:text=This%20en%20java,alg%C3%BAn%20otro%20m%C3%A9todo%20u%20objeto>