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

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Ing. Jorge Ángel Hernández López |
| *Asignatura:* | Programación Orientada a Objetos |
| *Grupo:* | 04 |
| *No. de práctica(s):* | 03 |
| *Integrante(s):* | Barrera Treviño José Gerardo  Muñoz San Agustin Victoria Monserrat  Olvera Bravo Cynthia Carolina  Leocadio Chávez Rodrigo  Velasco Garcia Santiago |
| *No. de lista o brigada:* | Brigada 01 |
| *Semestre:* | 2024-1 |
| *Fecha de entrega:* | 11 de septiembre de 2023 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_\_\_

# **Introducción**

La práctica se titula utilerías y clases de uso general donde se abordan los siguientes temas:

* Arreglos

Un arreglo es una estructura ordenada de elementos del mismo tipo, que son accesibles a través de un índice. Los datos o elementos que contiene pueden ser de tipo primitivo o referencias a objetos. Además, pueden ser unidimensionales (1) o multidimensionales (más de 1).

* Argumentos por línea de comandos

Se refieren a los valores que se pasan a una aplicación Java cuando se ejecuta desde la línea de comandos o la terminal. Estos argumentos son proporcionados por el usuario al momento de ejecutar el programa y se utilizan para personalizar la ejecución de la aplicación según las necesidades específicas de cada ejecución. Los argumentos por línea de comandos son una forma común de interactuar con programas Java y permiten que los programas sean más flexibles y reutilizables.

* API de java

Proporciona una interfaz Java con un conjunto de clases de programa de utilidad y métodos que puede utilizar para escribir código Java. Estas APIs cubren diferentes áreas de desarrollo de software, desde la manipulación de cadenas hasta la creación de interfaces de usuario y el acceso a bases de datos. Java es conocido por su rica colección de bibliotecas estándar y APIs, lo que lo convierte en un lenguaje versátil para una amplia gama de aplicaciones.

* Manejo de cadenas

Se refiere al proceso de trabajar con cadenas de caracteres en la programación, lo que implica realizar diversas operaciones y manipulaciones en texto o secuencias de caracteres. Las cadenas de caracteres son una parte fundamental en la mayoría de los lenguajes de programación y se utilizan para representar información de texto, como palabras, frases, nombres, números como cadenas, y mucho más.

1. Creación de cadenas: La creación de cadenas implica definir y asignar valores a variables de cadena. Esto se hace utilizando comillas simples ('') o comillas dobles ("") en la mayoría de los lenguajes de programación.
2. Concatenación: La concatenación es la operación de unir dos o más cadenas para formar una cadena más larga. Esto es útil para combinar texto o datos de múltiples fuentes.
3. Extracción de subcadenas: Puedes extraer partes específicas de una cadena utilizando índices o métodos específicos del lenguaje de programación. Por ejemplo, obtener los primeros cinco caracteres de una cadena.
4. Búsqueda y reemplazo: Puedes buscar una subcadena dentro de una cadena más grande y reemplazarla por otra. Esto es útil para realizar ediciones en el texto.
5. Comparación de cadenas: Es posible comparar cadenas para determinar si son iguales o cuál es mayor o menor en orden alfabético.
6. Transformación de mayúsculas y minúsculas: Cambiar el caso de las letras en una cadena, como convertir todo a mayúsculas o minúsculas.
7. Longitud de cadena: Obtener la cantidad de caracteres en una cadena.
8. División de cadenas: Separar una cadena en partes más pequeñas utilizando un delimitador específico, como un espacio en blanco o una coma
9. Formateo de cadenas: Crear cadenas de salida con un formato específico, por ejemplo, para mostrar números con un cierto número de decimales.

* Wrappers

Llamadas clases envolventes, permiten manejar a los datos primitivos como objetos, así como convertirlos en cadenas de caracteres a números. Como a los siguientes:

* Integer para int.
* Double para double.
* Boolean para boolean.
* Character para char.
* Byte para byte.
* Short: para short.
* Long para long.
* Float para float.
* Autoboxing

Permite convertir un dato primitivo a uno tipo envolvente. Esto quiere decir que es una característica que permite la conversión automática entre tipos de datos primitivos y sus correspondientes clases wrapper (clases que encapsulan tipos primitivos). Esta característica simplifica el código y hace que sea más conveniente trabajar con tipos primitivos cuando se requieren objetos wrapper y viceversa.

* Colecciones: hashtable, arraylist

Se utilizan para almacenar y manipular conjuntos de elementos, pero tienen diferencias significativas en su funcionamiento y uso.

En el caso de la **Hashtable** se utiliza cuando se necesita una estructura de datos que permita un acceso rápido a los valores a través de una clave única. Es útil en situaciones donde se requiere la sincronización, como en entornos multihilo, pero pueden afectar el rendimiento de las aplicaciones donde no se utilice la sincronización.

Para las **Arraylist** se utiliza cuando se necesita una lista de elementos en un orden específico, y la capacidad de la lista puede crecer o reducirse dinámicamente. Son muy utilizados para almacenar colecciones de datos en aplicaciones Java. Además, su eficiencia se basa en que exista la sincronización.

* Clases de utilerías: math, date y calendar

La clase **Math** se encuentra en el paquete *java.lang* y proporciona métodos estáticos para realizar operaciones matemáticas comunes, como funciones trigonométricas, exponenciación, raíces cuadradas, redondeo, valor absoluto, etc.

La clase **Date** se encuentra en el paquete *java.util* y se utiliza para representar y manipular fechas y horas. Sin embargo, Date ha quedado obsoleta en gran medida y se recomienda utilizar las clases LocalDate, LocalTime, y LocalDateTime de la API java.time introducida en Java 8 para un manejo de fechas y horas más moderno y flexible.

La clase **Calendar** se encuentra en el paquete *java.util* y se utiliza para trabajar con fechas y horas en un formato más versátil que Date. Proporciona métodos para obtener y configurar valores de año, mes, día, hora, minuto, segundo, etc. También es útil para realizar cálculos con fechas.

# **Desarrollo**

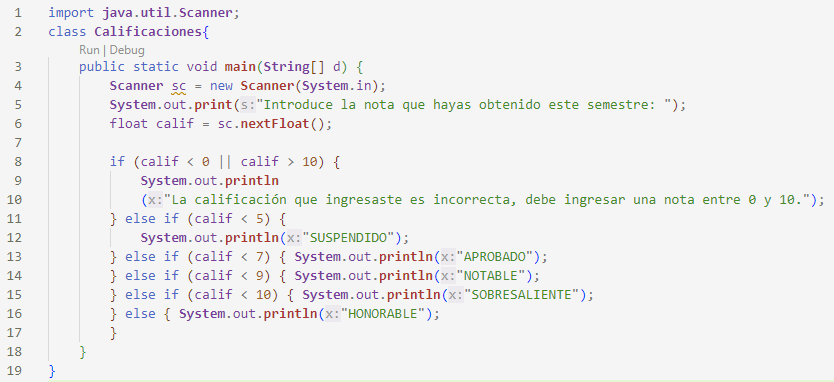
1. Realizar un programa que, dada la nota numérica de un alumno, imprima por pantalla la nota literal, sabiendo que menos de 5 es SUSPENDIDO, entre 5 y menos de 7 es APROBADO, entre 7 y menos de 9 es NOTABLE, entre 9 y menos de 10 SOBRESALIENTE, y un 10 HONORABLE. El programa deberá considerar el caso de que se introduzca un valor incorrecto por teclado.

Biblioteca usada para leer valores



Capturas del código del programa

En este primer ejercicio se puede observar que mediante el uso de condicionales se puede obtener una respuesta inmediata referente al valor que se le de como calificación. En el caso de Java es posible incluir un “*if”* unido a “*else if*” que definan la característica conforma la calificación. También el resultado lo define el operador lógico *OR* puesto dentro del if.



Compilación del código (ejemplo)



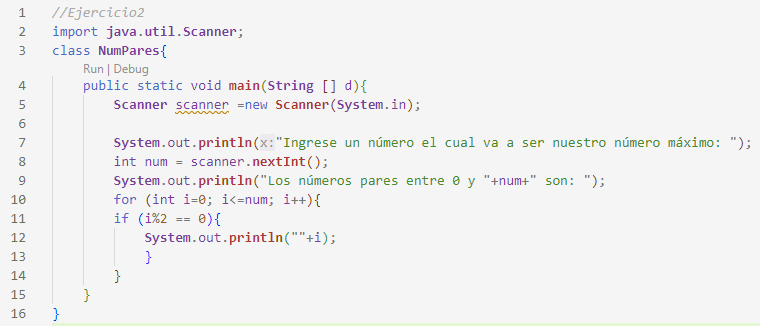
1. Realizar un programa que muestre los números pares comprendidos entre 0 y un valor introducido por teclado por el usuario.

Biblioteca usada para leer valores

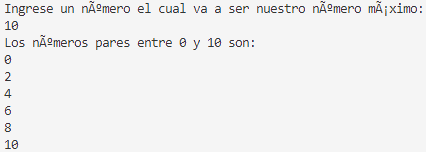


Capturas del código del programa

Usamos un Scanner para obtener la lectura del usuario. Además, utilizamos dos for, uno evalúa desde el 0 hasta el valor ingresado por el usuario y el otro evalúa los valores para solo imprimir los valores pares dentro del máximo y mínimo.



Compilación del código (ejemplo)



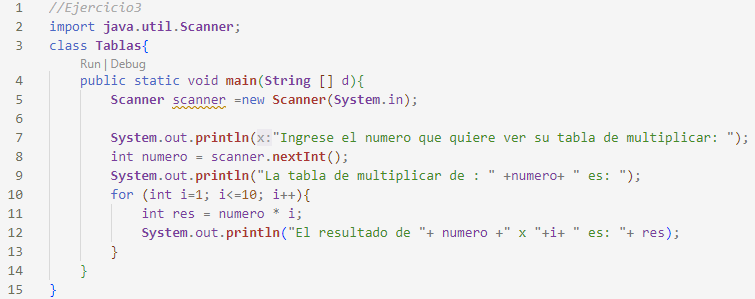
1. Realizar un programa que lea un número entero de teclado y muestre su tabla de multiplicar por pantalla.

Biblioteca usada para leer valores

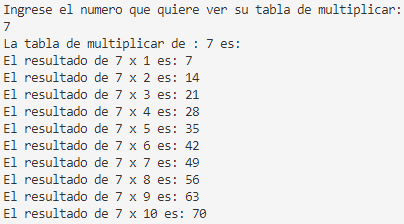


Capturas del código del programa

Nuevamente usamos un Scanner para obtener la lectura del usuario. Además, utilizamos un for que evalúa el número a multiplicar del 1 al 10 y así imprimir todos los resultados de esas multiplicaciones al valor ingresado.



Compilación del código (ejemplo)



1. Realizar un programa que permita calcular el perímetro y el área de figuras geométricas. Deberá de mostrar un menú principal como el siguiente:
2. Triángulo
3. Cuadrado
4. Círculo
5. Salir

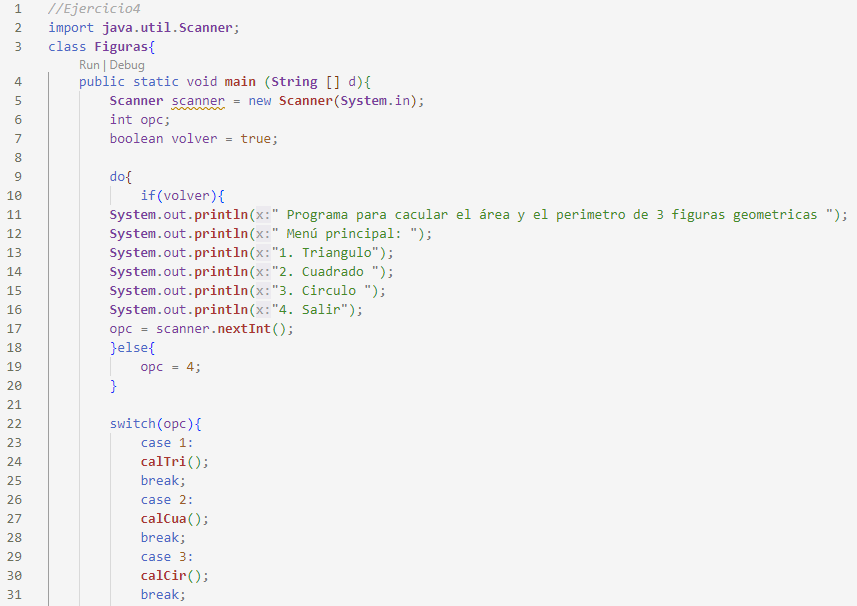
Biblioteca usada para leer valores



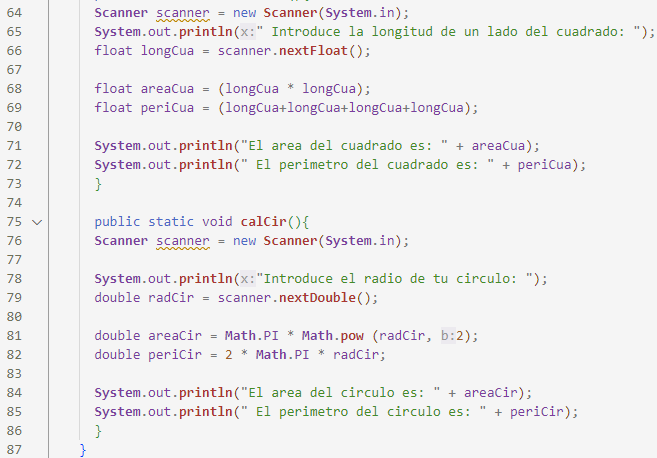
Capturas del código del programa

Este programa fue más complejo en el aspecto de que se utilizó un switch en el que se evaluando los 4 casos dados (tres figuras y un salir). En resumen, explicar este programa es basado en entender el funcionamiento de un switch, eliges una opción y conforme a las características de esa figura se van desenvolviendo los datos que se le piden al usuario. En este caso se denominan case.

También se puede observar que para el cálculo de las áreas y perímetros se encasillan en fórmulas predeterminadas, dando lugar a una respuesta rápida del programa. Por otro lado, las instrucciones dadas al usuario son más concretas y claras para evitar problemas en su compilación.

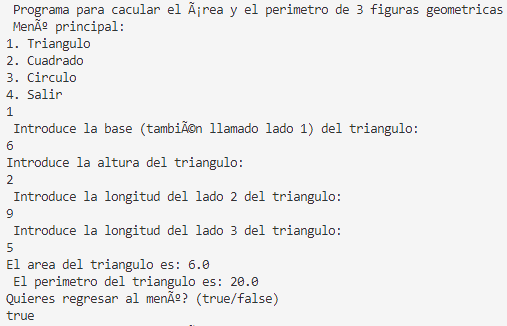






Compilación del código (ejemplo)

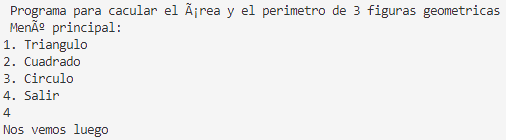
* Cálculo normal de una figura:



* Al terminar de calcular te pregunta si deseas intentarlo con otra o no:



* Opción salir:



# Conclusiones

Barrera Treviño José Gerardo: Durante la elaboración de estos programas pusimos en práctica varias cosas que ya veníamos conociendo desde Fundamentos de Programación, como lo son las estructuras de control “*if*”, “*if-else*”, “*switch*”, para realizar un óptimo programa. También utilizamos lo recién aprendido en Java que es la interacción con el usuario mediante la clase “*Scanner”.*

Por otro lado, para realizar el programa 4, tuve que investigar cómo se utilizaba a dentro del programa que realmente no tiene diferencia con Python ya que es lo mismo: “*Math.PI*” y así sin problema realizamos todos los ejercicios con éxito.

Muñoz San Agustin Victoria Monserrat: Esta práctica fue como un repaso de los conocimientos previos y aprendizaje de nuevas estructuras dentro de la programación. Aunque anteriormente hemos manejado diferentes lenguajes de programación, Java me parece aún un poco abstracto por el manejo de clases y sus maneras de desarrollar los diferentes casos. Sin embargo, esta práctica me hizo entender mejor el manejo de la consola y la importancia de la declaración de las clases, también el uso de los wrappers y bibliotecas en el mundo de Java.

Olvera Bravo Cynthia Carolina: Al momento de realizar la práctica pudimos poner en práctica diferentes estructuras que ya habíamos revisado anteriormente de manera teórica. Creamos arreglos y pudimos implementar cadenas en java, así observamos las diferentes operaciones que se pueden hacer con estas. Utilizamos conocimientos adquiridos ya en los semestres anteriores como son los ciclos, pero aplicados utilizando java como lenguaje realizando una investigación acerca de sus bibliotecas y las “APIs” para así poder implementarlas de manera eficiente y eficaz en el código. No tuvimos problemas mayores durante la realización de esta práctica.

Velasco Velasco Santiago: Durante la práctica utilizamos clases vistas en clase tales como Scanner y utilizamos estructuras ya conocidas como if, do while, switch, entre otras, pero ahora desde el paradigma orientado a objetos. Igualmente se trabajó con el manejo de cadenas y de múltiples tipos de datos primitivos. El cambio de un paradigma a otro se vio respaldado por la teoría proporcionada en clase. Todos los programas trabajan desde el main, lo cual nos hablas de que no hay mucha modularidad, sin embargo permite visualizar y entender de mejor manera el funcionamiento de cada programa. En general no tuvimos problemas significativos y los objetivos de la práctica fueron cumplidos satisfactoriamente.

# Leocadio Chávez Rodrigo: La realización de esta práctica no presentó grandes complicaciones, ya que los programas desarrollados no cuentan con un grado de dificultad mayor a los temas vistos en semestres anteriores más que poner en práctica estos conocimientos en el lenguaje Java. Tales como los arreglos y el uso de cadenas, sin embargo ahora con un enfoque diferente, relacionado a la materia, orientado a objetos. No se presentó ningún obstáculo mayor.

# Referencias

IBM (2023). “API de Java”. Recuperado el 10 de septiembre de 2023 de htpps ://www.ibm.com/docs/es/product-master/12.0.0?topic=code-java-api

OpenAI. (2023). "¿Qué son Argumentos por línea de comandos en Java?" [Mensaje en línea]. Recuperado el 10 de septiembre de 2023 de <https://www.openai.com>.

OpenAI. (2023). "¿Qué conforma el Manejo de cadenas Argumentos en Java?" [Mensaje en línea]. Recuperado el 10 de septiembre de 2023 de https://www.openai.com.

OpenAI. (2023). "¿Qué diferencia a las hashtable y arraylist?" [Mensaje en línea]. Recuperado el 10 de septiembre de 2023 de https://www.openai.com.

\*Otros obtenidos de apuntes y presentaciones del Classroom