EJERCICIOS JAVA

Contenido

CUADROS DE DIÁLOGO CON JAVA.SWING

Swing es una biblioteca de clases incluida en el paquete **javax.swing** en Java. Proporciona un conjunto de componentes gráficos y herramientas para crear interfaces de usuario (UI) en aplicaciones de escritorio. Con Swing, puedes diseñar y desarrollar interfaces gráficas de usuario interactivas y ricas en funcionalidades.

Dentro del paquete **javax.swing**, encontrarás clases para diversos componentes de la interfaz de usuario, como botones, etiquetas, cuadros de texto, paneles, marcos, etc. También encontrarás clases para manejar eventos de usuario, disposición de componentes, gráficos, ventanas, diálogos y más.

Cuando te aburras de la consola en estos ejercicios, también puedes solicitar datos al usuario y mostrar resultados de una manera más interactiva que simplemente utilizando la consola.

Cuadro de Diálogo de Entrada:

Puedes utilizar la clase `JOptionPane.showInputDialog()` para mostrar un cuadro de diálogo de entrada donde el usuario puede ingresar datos. Por ejemplo:

```
import javax.swing.JOptionPane;

public class CuadroDialogoEntrada {

    public static void main(String[] args) {

        String nombre = JOptionPane.showInputDialog(null, "Ingrese su nombre:");

        JOptionPane.showMessageDialog(null, "¡Hola, " + nombre + "!");

    }

}
```

Cuadro de Diálogo de Mensaje:

Puedes utilizar la clase `JOptionPane.showMessageDialog()` para mostrar un cuadro de diálogo de mensaje con información para el usuario. Por ejemplo:

```
import javax.swing.JOptionPane;

public class CuadroDialogoMensaje {

    public static void main(String[] args) {

        JOptionPane.showMessageDialog(null, "¡Bienvenido al programa!");

    }

}
```

INTRODUCCIÓN AL LENGUAJE

- Variables y Tipos de Datos Primitivos
- Operadores
- Estructuras de Control (if, else, switch)
- Bucles (for, while)

1. **Variables y Operadores:**

Declara una variable entera llamada `num1` y otra variable entera llamada `num2`. Asigna valores a ambas variables y luego muestra la suma, resta, multiplicación y división de `num1` y `num2` en la consola.

2. **Operadores Relacionales y Booleanos:**

Declara una variable entera llamada `edad` y asigna un valor. Luego, utiliza una estructura `if` para determinar si la persona es mayor de edad. Muestra el resultado en la consola.

3. **Estructuras de Control Variadas:**

Crea un programa que solicite al usuario ingresar un número entero. Si el número es par, muestra "El número es par" en la consola; de lo contrario, muestra "El número es impar".

4. **Bucles:**

Utiliza un bucle `for` para mostrar los números del 1 al 10 en la consola.

5. **Bucles y Operadores:**

Utiliza un bucle `while` para imprimir todos los números pares del 0 al 20 en la consola.

6. **Operadores Matemáticos y Asignación:**

Declara una variable entera `contador` y asígnale el valor 0. Utiliza un bucle `do-while` para incrementar `contador` en 5 en cada iteración hasta que `contador` sea mayor que 50. Muestra el valor de `contador` en cada iteración.

7. **Estructuras de Control Anidadas:**

Escribe un programa que solicite al usuario ingresar dos números enteros y luego determine si el primer número es divisible por el segundo número. Muestra el resultado en la consola.

8. **Estructuras de Control Switch:**

Crea un programa que solicite al usuario ingresar un número del 1 al 7 representando un día de la semana. Utiliza una estructura `switch` para mostrar el nombre del día en la consola.

9. **Tipos Primitivos y Operadores:**

Declara una variable de tipo double llamada precio y asígnale un valor decimal. Luego, declara una variable de tipo int llamada cantidad y asígnale un valor entero. Calcula el total multiplicando el precio por la cantidad e imprime el resultado en la consola.

10. **Operadores Relacionales y Booleanos:**

Declara una variable de tipo boolean llamada llueve e inicialízala con un valor. Utiliza una estructura if-else para imprimir "Llevo paraguas" si llueve es true, y "No llevo paraguas" si llueve es false.

11. Estructuras de Control y Tipos Primitivos:

Escribe un programa que solicite al usuario ingresar su edad y una temperatura en grados Celsius. Si la temperatura es superior a 25 grados y la edad es menor de 40 años, muestra "¡Hace calor!" en la consola; de lo contrario, muestra "El clima es agradable".

12. Bucles y Tipos Primitivos:

Utiliza un bucle for para calcular la suma de los primeros 10 números enteros. Imprime el resultado en la consola.

13. Operadores Matemáticos y Asignación:

Declara una variable num de tipo int e inicialízala con un valor. Luego, incrementa num en 1 utilizando el operador de incremento (++) y muestra el nuevo valor en la consola.

14. Estructuras de Control Anidadas y Booleanos:

Escribe un programa que solicite al usuario ingresar su edad y si tiene un carnet de estudiante (utiliza un valor boolean para representar esta condición). Si la edad es menor de 18 o tiene un carnet de estudiante, muestra "Tiene descuento"; de lo contrario, muestra "No tiene descuento".

15. Estructuras de Control Switch y Tipos Primitivos:

Crea un programa que solicite al usuario ingresar el código de un producto (utiliza un valor char para representar el código). Utiliza una estructura switch para determinar el nombre del producto correspondiente y muestra el resultado en la consola.

16. Suma de Números Long:

Declara dos variables de tipo long y asigna valores a ambas. Luego, calcula la suma de estos dos números y muestra el resultado en la consola.

17. Multiplicación de Números Long:

Declara tres variables de tipo long y asigna valores a ellas. Calcula el producto de estos tres números y muestra el resultado en la consola.

18. Operaciones Aritméticas con Long:

Declara una variable de tipo long y asígnale un valor. Realiza varias operaciones aritméticas como suma, resta, multiplicación y división utilizando este número y muestra los resultados en la consola.

19. Conversión de Tipos:

Declara una variable de tipo int y asigna un valor grande que no quepa en un int. Luego, convierte este valor a long y muestra ambos valores (el original y el convertido) en la consola.

20. Uso de Constantes Long:

Declara una constante de tipo long con un valor muy grande (por ejemplo, representando el número de segundos en un año). Utiliza esta constante en un cálculo, como por ejemplo, para convertir segundos a minutos y muestra el resultado en la consola.

CADENAS DE TEXTO

- Clase String
- Clase Character
- Clase StringBuilder
- Clase StringBuffer

1. **Longitud de una Cadena:**

Escribe un programa que solicite al usuario ingresar una cadena de texto y luego muestre la longitud de la cadena en la consola.

2. **Concatenación de Cadenas:**

Declara dos variables de tipo String y asígnales valores. Luego, concatena estas dos cadenas y muestra el resultado en la consola.

3. **Mayúsculas y Minúsculas:**

Escribe un programa que solicite al usuario ingresar una palabra en minúsculas y luego convierta esta palabra a mayúsculas antes de mostrarla en la consola.

4. **Subcadena:**

Declara una cadena de texto y luego utiliza el método substring() para extraer una subcadena de la cadena original. Muestra la subcadena resultante en la consola.

5. **Reemplazo de Caracteres:**

Escribe un programa que solicite al usuario ingresar una frase y luego reemplace todas las ocurrencias de una letra específica por otra letra. Muestra la frase resultante en la consola.

6. **División de Cadena:**

Declara una cadena de texto que contenga una frase larga. Utiliza el método split() para dividir la cadena en palabras individuales y muestra cada palabra en la consola en una línea separada.

7. **Búsqueda de Subcadena:**

Escribe un programa que solicite al usuario ingresar una frase y una palabra. Luego, busca si la palabra ingresada se encuentra dentro de la frase y muestra el resultado en la consola.

8. **Eliminación de Espacios en Blanco:**

Declara una cadena de texto que contenga espacios en blanco al principio y al final. Utiliza el método trim() para eliminar estos espacios en blanco y muestra la cadena resultante en la consola.

9. **Conteo de Palabras:**

Escribe un programa que solicite al usuario ingresar un párrafo. Luego, cuenta el número de palabras en el párrafo (ignorando los espacios en blanco adicionales) y muestra el resultado en la consola.

10. **Inversión de Palabras:**

Escribe un programa que solicite al usuario ingresar una frase. Luego, invierte el orden de las palabras en la frase y muestra la frase invertida en la consola. Por ejemplo, si el usuario ingresa "Hola mundo", el programa debería mostrar "mundo Hola".

11.Comprobación de Carácter Alfabético:

Escribe un programa que solicite al usuario ingresar un carácter. Utiliza el método isLetter() de la clase Character para verificar si el carácter ingresado es alfabético o no. Muestra un mensaje indicando el resultado en la consola.

12.Comprobación de Carácter Numérico:

Declara una variable de tipo char e inicialízala con un dígito. Utiliza el método isDigit() de la clase Character para determinar si el carácter es un dígito numérico. Muestra el resultado en la consola.

13.Comprobación de Carácter en Minúsculas:

Escribe un programa que solicite al usuario ingresar un carácter. Utiliza el método isLowerCase() de la clase Character para verificar si el carácter ingresado es una letra minúscula. Muestra un mensaje indicando el resultado en la consola.

14.Conversión a Mayúsculas:

Declara una variable de tipo char e inicialízala con una letra minúscula. Utiliza el método toUpperCase() de la clase Character para convertir el carácter a mayúsculas y muestra el resultado en la consola.

15.Conversión a Minúsculas:

Escribe un programa que solicite al usuario ingresar una letra mayúscula. Utiliza el método toLowerCase() de la clase Character para convertir la letra a minúsculas y muestra el resultado en la consola.

16.Comprobación de Carácter de Espacio en Blanco:

Declara una variable de tipo char e inicialízala con un carácter de espacio en blanco. Utiliza el método isWhitespace() de la clase Character para verificar si el carácter es un espacio en blanco. Muestra el resultado en la consola.

17.Comprobación de Carácter de Puntuación:

Escribe un programa que solicite al usuario ingresar un carácter. Utiliza el método isLetterOrDigit() de la clase Character para verificar si el carácter es una letra o un dígito. Muestra un mensaje indicando el resultado en la consola.

18.Concatenación Eficiente:

Escribe un programa que cree un objeto StringBuilder y lo utilice para concatenar una serie de cadenas de texto. Muestra el resultado final en la consola.

19.Inversión de Cadena:

Declara una cadena de texto y luego utiliza un objeto StringBuilder para invertir la cadena. Muestra la cadena invertida en la consola.

20.Eliminación de Caracteres:

Declara una cadena de texto y utiliza un objeto StringBuilder para eliminar todos los caracteres que coincidan con un carácter específico. Muestra la cadena resultante en la consola.

21.Reemplazo de Subcadena:

Escribe un programa que solicite al usuario ingresar una frase y dos palabras. Utiliza un objeto StringBuilder para reemplazar todas las ocurrencias de la primera palabra con la segunda palabra en la frase. Muestra la frase resultante en la consola.

22.Inserción de Caracteres:

Declara una cadena de texto y utiliza un objeto StringBuilder para insertar un carácter específico en cada posición de la cadena. Muestra la cadena resultante en la consola.

23.Borrado de Contenido:

Escribe un programa que cree un objeto StringBuilder con un contenido inicial y luego borre todo su contenido. Muestra el estado del StringBuilder después del borrado en la consola.

24.Concatenación Eficiente:

Escribe un programa que cree un objeto StringBuffer y lo utilice para concatenar una serie de cadenas de texto. Muestra el resultado final en la consola.

25.Inversión de Cadena:

Declara una cadena de texto y luego utiliza un objeto StringBuffer para invertir la cadena. Muestra la cadena invertida en la consola.

26.Eliminación de Caracteres:

Declara una cadena de texto y utiliza un objeto StringBuffer para eliminar todos los caracteres que coincidan con un carácter específico. Muestra la cadena resultante en la consola.

27.Reemplazo de Subcadena:

Escribe un programa que solicite al usuario ingresar una frase y dos palabras. Utiliza un objeto StringBuffer para reemplazar todas las ocurrencias de la primera palabra con la segunda palabra en la frase. Muestra la frase resultante en la consola.

28.Inserción de Caracteres:

Declara una cadena de texto y utiliza un objeto StringBuffer para insertar un carácter específico en cada posición de la cadena. Muestra la cadena resultante en la consola.

29.Borrado de Contenido:

Escribe un programa que cree un objeto StringBuffer con un contenido inicial y luego borre todo su contenido. Muestra el estado del StringBuffer después del borrado en la consola.

FECHAS

- Fechas Clase Date (a la antigua, deprecated)
- Fechas paquete java.time

1. **Obtener la Fecha Actual:**

Escribe un programa que utilice la clase Date para obtener la fecha y hora actuales del sistema y luego muestre esta información en la consola.

2. **Comparación de Fechas:**

Declara dos objetos Date que representen diferentes fechas. Luego, compara estas fechas utilizando el método compareTo() de la clase Date y muestra el resultado en la consola.

3. **Sumar Días a una Fecha:**

Declara un objeto Date que represente una fecha específica. Utiliza el método setTime() para modificar la fecha sumando un número específico de días. Muestra la nueva fecha en la consola.

4. **Formato de Fecha Personalizado:**

Declara un objeto Date que represente una fecha y hora específicas. Utiliza la clase SimpleDateFormat para formatear esta fecha en un formato personalizado (por ejemplo, "dd/MM/yyyy HH:mm:ss") y muestra la fecha formateada en la consola.

5. **Obtener la Fecha Actual:**

Escribe un programa que utilice la clase LocalDate para obtener la fecha actual del sistema y luego muestre esta información en la consola.

6. **Obtener la Hora Actual:**

Declara un objeto LocalTime y utilízalo para obtener la hora actual del sistema. Muestra esta información en la consola.

7. **Obtener la Fecha y Hora Actuales:**

Utiliza la clase LocalDateTime para obtener la fecha y hora actuales del sistema. Muestra esta información en la consola.

8. **Comparación de Fechas:**

Declara dos objetos LocalDate que representen diferentes fechas. Utiliza el método compareTo() para comparar estas fechas y muestra el resultado en la consola.

9. **Sumar y Restar Días a una Fecha:**

Declara un objeto LocalDate que represente una fecha específica. Utiliza el método plusDays() y minusDays() para sumar y restar un número específico de días a esta fecha. Muestra las fechas resultantes en la consola.

10. **Diferencia de Días entre dos Fechas:**

Escribe un programa que solicite al usuario ingresar dos fechas utilizando la clase LocalDate. Luego, calcula la diferencia en días entre estas fechas y muestra el resultado en la consola.

11. **Formato de Fecha Personalizado:**

Declara un objeto LocalDateTime que represente una fecha y hora específicas. Utiliza la clase DateTimeFormatter para formatear esta fecha en un formato personalizado (por ejemplo, "dd/MM/yyyy HH:mm:ss") y muestra la fecha formateada en la consola.

12.Día de la Semana:

Declara un objeto LocalDate que represente una fecha específica. Utiliza el método getDayOfWeek() para obtener el día de la semana correspondiente a esta fecha y muestra el resultado en la consola.

13.Verificar Año Bisiesto:

Escribe un programa que solicite al usuario ingresar un año. Utiliza la clase Year para verificar si el año ingresado es bisiesto o no. Muestra un mensaje en la consola indicando el resultado.

14.Zonas Horarias:

Declara un objeto ZonedDateTime que represente la fecha y hora actual en una zona horaria específica (por ejemplo, "America/New_York"). Muestra esta información en la consola.

NÚMEROS

- Autoboxing y unboxing
- Grandes números y alta precisión
- Clase NumberFormat

Autoboxing y Unboxing:

1. Declara una lista de enteros primitivos (**int**). Luego, utiliza autoboxing para convertirlos en objetos **Integer** y súmalos utilizando un bucle for-each.

2. Declara un array de objetos **Integer**. Utiliza unboxing para convertirlos en valores enteros primitivos y encuentra el valor máximo y mínimo en el array.

3. Escribe un método que tome un parámetro de tipo **Integer** y devuelva su valor incrementado en uno. Utiliza autoboxing y unboxing en el método.

Grandes Números y Alta Precisión:

4. Utiliza la clase **BigInteger** para calcular el factorial de un número entero grande ingresado por el usuario.

5. Utiliza la clase **BigDecimal** para calcular la suma de una serie de números decimales de alta precisión ingresados por el usuario.

6. Escribe un programa que calcule el valor de π utilizando la fórmula de Leibniz para aproximaciones de alta precisión. Utiliza la clase **BigDecimal** para mantener la precisión necesaria.

Clase NumberFormat:

7. Escribe un programa que formatee un número entero con separadores de miles y lo muestre en la consola utilizando la clase **NumberFormat**.

8. Escribe un programa que formatee un número decimal con dos decimales y lo muestre en la consola utilizando la clase **NumberFormat**.

9. Escribe un programa que solicite al usuario ingresar un número y un patrón de formato decimal personalizado (por ejemplo, "#,##0.00"). Utiliza la clase **NumberFormat** para formatear el número según el patrón y muestra el resultado en la consola.

10. Escribe un programa que formatee un número entero como moneda y lo muestre en la consola utilizando la clase **NumberFormat**.

Otros Ejercicios Variados:

11. Escribe un programa que solicite al usuario ingresar un número y determine si es par o impar. Utiliza autoboxing y unboxing para manejar los números enteros.

11. Escribe un programa que calcule la suma de todos los números enteros entre 1 y un número entero ingresado por el usuario. Utiliza autoboxing y unboxing para manejar los números enteros.

12. Escribe un programa que solicite al usuario ingresar una cantidad de dinero en dólares y conviértala a euros utilizando la tasa de cambio actual. Utiliza la clase **BigDecimal** para mantener la precisión necesaria y la clase **NumberFormat** para formatear el resultado.

13. Escribe un programa que calcule el interés compuesto para una inversión dada utilizando la fórmula A = P(1 + r/n)^(nt), donde A es el saldo total, P es el principal, r es la tasa de interés anual, n es el número de veces que se compone el interés al año, y t es el número de años. Utiliza la clase **BigDecimal** para mantener la precisión necesaria.

14. Escribe un programa que solicite al usuario ingresar un número y calcule su raíz cuadrada utilizando la clase **BigDecimal** y el algoritmo de Newton-Raphson para calcular raíces cuadradas.

ARRAYS

1. **Ordenamiento de Números:**

Escribe un programa que cree un array de números enteros desordenados y los ordene en orden ascendente utilizando el método sort() de la clase Arrays. Muestra el array ordenado en la consola.

2. **Búsqueda de Elemento:**

Declara un array de cadenas de texto y solicita al usuario ingresar una cadena para buscar en el array. Utiliza el método binarySearch() de la clase Arrays para buscar la cadena en el array y muestra el resultado en la consola.

3. **Copia de Arrays:**

Declara un array de números enteros y luego copia sus elementos a otro array utilizando el método copyOf() de la clase Arrays. Muestra ambos arrays en la consola para verificar que se haya realizado la copia correctamente.

4. **Relleno de Elementos:**

Escribe un programa que cree un array de números enteros con un tamaño especificado y luego rellene todos sus elementos con un valor específico utilizando el método fill() de la clase Arrays. Muestra el array resultante en la consola.

5. **Comparación de Arrays:**

Declara dos arrays de números enteros y utiliza el método equals() de la clase Arrays para comparar si son iguales. Muestra un mensaje en la consola indicando el resultado de la comparación.

6. **Conversión a Cadena:**

Declara un array de cadenas de texto y utiliza el método toString() de la clase Arrays para convertir el array en una cadena de texto. Muestra la cadena resultante en la consola.

7. **Ordenamiento Personalizado:**

Escribe un programa que cree un array de cadenas de texto y los ordene en orden descendente según la longitud de las cadenas utilizando un comparador personalizado y el método sort() de la clase Arrays. Muestra el array ordenado en la consola.

8. **Cálculo de Suma:**

Declara un array de números enteros y calcula la suma de todos sus elementos utilizando un bucle y el método stream().sum() de la clase Arrays. Muestra la suma en la consola.

9. Eliminación de Elementos Duplicados:

Escribe un programa que cree un array de números enteros con elementos duplicados y elimine todos los elementos duplicados utilizando el método stream().distinct() de la clase Arrays. Muestra el array resultante en la consola.

10. Búsqueda de Elementos Pares:

Declara un array de números enteros y utiliza el método stream().filter() de la clase Arrays para buscar todos los números pares en el array. Muestra los números pares encontrados en la consola.

11. Cálculo de Promedio:

Declara un array de números enteros y calcula el promedio de todos sus elementos utilizando el método stream().average() de la clase Arrays. Muestra el promedio en la consola.

12. Filtrado de Elementos Mayores a un Valor:

Escribe un programa que cree un array de números enteros y utilice el método stream().filter() de la clase Arrays para filtrar todos los números mayores a un valor específico. Muestra los números filtrados en la consola.

13. Conversión a Array Primitivo:

Declara un ArrayList de números enteros y conviértelo en un array primitivo de enteros utilizando el método toArray() de la clase Arrays. Muestra el array resultante en la consola.

14. Búsqueda de Elemento con Índice:

Escribe un programa que cree un array de cadenas de texto y busque un elemento específico utilizando el método indexOf() de la clase Arrays. Si el elemento existe, muestra su índice en la consola; de lo contrario, muestra un mensaje indicando que no se encontró.

15. División de Array:

Declara un array de números enteros y utiliza el método copyOfRange() de la clase Arrays para crear un nuevo array que contenga solo una parte del array original. Muestra el nuevo array en la consola.

LISTAS Y COLECCIONES

En Java, hay varias clases y interfaces en el paquete `java.util` que se utilizan para trabajar con listas y colecciones. Estas son algunas de las clases e interfaces más comunes que se utilizan para trabajar con listas y colecciones en Java.

ArrayList: Implementa la interfaz `List` y proporciona una implementación de lista dinámica respaldada por un array. Permite agregar, eliminar y acceder a elementos de manera eficiente.

LinkedList: Implementa la interfaz `List` y proporciona una implementación de lista doblemente enlazada. Permite agregar y eliminar elementos de manera eficiente tanto al principio como al final de la lista, pero el acceso aleatorio es más lento que en `ArrayList`.

HashSet: Implementa la interfaz `Set` y proporciona una implementación de conjunto basada en una tabla hash. Almacena elementos únicos y no garantiza el orden de los elementos.

TreeSet: Implementa la interfaz `SortedSet` y proporciona una implementación de conjunto basada en un árbol binario de búsqueda. Almacena elementos únicos y los mantiene ordenados en orden ascendente o descendente.

HashMap: Implementa la interfaz `Map` y proporciona una implementación de mapa basada en una tabla hash. Almacena pares clave-valor y no garantiza el orden de los elementos.

TreeMap: Implementa la interfaz `SortedMap` y proporciona una implementación de mapa basada en un árbol binario de búsqueda. Almacena pares clave-valor y los mantiene ordenados en orden ascendente o descendente según la clave.

Queue: Es una interfaz que define un comportamiento de cola (FIFO - First In, First Out). Algunas de las implementaciones comunes son `LinkedList` y `PriorityQueue`

Deque: Es una interfaz que define un comportamiento de cola doble (puede agregar y eliminar elementos tanto al principio como al final de la cola). Algunas de las implementaciones comunes son `ArrayDeque` y `LinkedList`.

Se incluyen algunos ejercicios sobre estas clases e interfaces.

ArrayList:

1. Agrega una serie de números enteros a un ArrayList y luego imprime cada elemento en la consola.

2. Elimina un elemento específico de un ArrayList y luego imprime el ArrayList actualizado.

3. Encuentra el índice de un elemento específico en un ArrayList y muestra su posición en la consola.

4. Verifica si un ArrayList está vacío o no y muestra un mensaje apropiado en la consola.

5. Copia todos los elementos de un ArrayList a otro ArrayList y luego imprime el nuevo ArrayList.

6. Ordena un ArrayList de cadenas en orden ascendente y muestra el ArrayList ordenado.

7. Elimina todos los elementos duplicados de un ArrayList y muestra el ArrayList resultante.

8. Calcula la suma de todos los elementos en un ArrayList de números enteros y muestra el resultado.

9. Convierte un ArrayList en un array primitivo de enteros y muestra el array resultante en la consola.

10. Crea un ArrayList de objetos personalizados y realiza una búsqueda de un objeto específico en el ArrayList.

LinkedList:

11. Agrega elementos al principio y al final de una LinkedList y muestra la LinkedList resultante.

12. Elimina el primer y último elemento de una LinkedList y muestra la LinkedList resultante.

13. Verifica si una LinkedList contiene un elemento específico y muestra el resultado en la consola.

14. Reemplaza un elemento específico en una LinkedList con otro elemento y muestra la LinkedList resultante.

15. Itera sobre una LinkedList utilizando un bucle for-each y muestra cada elemento en la consola.

HashSet:

16. Agrega una serie de cadenas a un HashSet y muestra el HashSet resultante.

17. Elimina un elemento específico de un HashSet y muestra el HashSet actualizado.

18. Verifica si un HashSet está vacío o no y muestra un mensaje apropiado en la consola.

19. Comprueba si un HashSet contiene todos los elementos de otro HashSet y muestra el resultado.

20. Itera sobre un HashSet utilizando un Iterator y muestra cada elemento en la consola.

TreeSet:

21. Agrega una serie de números enteros a un TreeSet y muestra el TreeSet resultante.

22. Elimina el elemento mínimo y máximo de un TreeSet y muestra el TreeSet actualizado.

23. Verifica si un TreeSet contiene un elemento específico y muestra el resultado en la consola.

24. Muestra todos los elementos de un TreeSet en orden ascendente utilizando un bucle for-each.

25. Muestra todos los elementos de un TreeSet en orden descendente utilizando un Iterator.

HashMap:

26. Crea un HashMap que almacene nombres como claves y edades como valores y muestra el HashMap resultante.

27. Actualiza el valor asociado a una clave específica en un HashMap y muestra el HashMap actualizado.

28. Verifica si un HashMap contiene una clave específica y muestra el resultado en la consola.

29. Muestra todas las claves de un HashMap utilizando un bucle for-each.

30. Muestra todos los valores de un HashMap utilizando un Iterator.

TreeMap:

31. Crea un TreeMap que almacene nombres como claves y edades como valores y muestra el TreeMap resultante.

32. Elimina una clave específica de un TreeMap y muestra el TreeMap actualizado.

33. Verifica si un TreeMap contiene una clave específica y muestra el resultado en la consola.

34. Muestra todas las claves de un TreeMap en orden ascendente utilizando un Iterator.

35. Muestra todos los valores de un TreeMap en orden descendente utilizando un bucle for-each.

Queue:

36. Agrega elementos a una Queue y muestra la Queue resultante.

37. Elimina el primer elemento de una Queue y muestra la Queue actualizada.

38. Verifica si una Queue contiene un elemento específico y muestra el resultado en la consola.

39. Muestra todos los elementos de una Queue utilizando un bucle while.

Deque:

40. Agrega elementos al principio y al final de un Deque y muestra el Deque resultante.

41. Elimina el primer y último elemento de un Deque y muestra el Deque actualizado.

42. Verifica si un Deque contiene un elemento específico y muestra el resultado en la consola.

43. Muestra todos los elementos de un Deque en orden ascendente utilizando un Iterator.

44. Muestra todos los elementos de un Deque en orden descendente utilizando un bucle for-each.

EXCEPCIONES (complejidad baja)

1.División Segura:

Escribe un programa que solicite al usuario dos números enteros y luego divida el primero por el segundo. Maneja la posible excepción de división por cero.

2. Conversión de Cadena a Número:

Solicita al usuario que ingrese un número como cadena y conviértelo a tipo entero. Maneja la excepción que puede ocurrir si el usuario ingresa una cadena no numérica.

3. Lectura de Archivo:

Escribe un programa que intente abrir un archivo específico y mostrar su contenido en la consola. Maneja la excepción que puede ocurrir si el archivo no se encuentra.

4. Lectura de Entrada de Usuario:

Solicita al usuario que ingrese su nombre y edad. Utiliza el método `nextInt()` de `Scanner` para leer la edad como un entero. Maneja la excepción que puede ocurrir si el usuario ingresa algo que no sea un entero como edad.

5. Búsqueda de Elemento en un Array:

Declara un array de números enteros y solicita al usuario que ingrese un índice. Intenta acceder al elemento en ese índice y maneja la excepción que puede ocurrir si el índice está fuera del rango del array.

6. Conversión de Tipo de Dato:

Declara una variable de tipo `double` y solicita al usuario que ingrese una cadena. Intenta convertir la cadena a un número decimal y maneja la excepción que puede ocurrir si la cadena no es un número válido.

7. Acceso a Elementos de una Lista:

Crea una lista de cadenas de texto y solicita al usuario que ingrese un índice. Intenta acceder al elemento en ese índice y maneja la excepción que puede ocurrir si el índice está fuera del rango de la lista.

8. Cálculo de Promedio de una Lista:

Solicita al usuario que ingrese una serie de números separados por comas. Divide la cadena en números individuales, calcula su promedio y muestra el resultado. Maneja la excepción que puede ocurrir si el usuario ingresa una cadena no numérica.

10. Manejo de Argumentos de Línea de Comandos:

Escribe un programa que intente convertir el primer argumento de línea de comandos a un número entero y luego lo imprima en la consola. Maneja la excepción que puede ocurrir si el usuario no proporciona ningún argumento o si el argumento no es un número válido.

MÉTODOS

1. **Máximo y Mínimo:** Escribe un método llamado **maximo** que tome dos números como parámetros y devuelva el mayor de los dos. Luego, escribe otro método llamado **minimo** que haga lo mismo pero devuelva el menor de los dos.

2. **Promedio de Números:** Escribe un método llamado **calcularPromedio** que tome un array de números como parámetro y devuelva su promedio.

3. **Factorial de un Número:** Escribe un método llamado **factorial** que tome un número entero como parámetro y devuelva su factorial (el producto de todos los enteros positivos menores o iguales a él).

4. **Suma de Dígitos:** Escribe un método llamado **sumaDigitos** que tome un número entero como parámetro y devuelva la suma de sus dígitos.

5. **Contar Vocales:** Escribe un método llamado **contarVocales** que tome una cadena de texto como parámetro y devuelva el número de vocales que contiene.

6. **Invertir Cadena:** Escribe un método llamado **invertirCadena** que tome una cadena de texto como parámetro y devuelva una nueva cadena con sus caracteres en orden inverso.

7. **Comprobar Palíndromo:** Escribe un método llamado **esPalindromo** que tome una cadena de texto como parámetro y devuelva **true** si es un palíndromo (se lee igual de izquierda a derecha que de derecha a izquierda) y **false** en caso contrario.

8. **Calcular Potencia:** Escribe un método llamado **potencia** que tome dos números enteros como parámetros y devuelva el primero elevado a la potencia del segundo.

9. **Calcular Área del Círculo:** Escribe un método llamado **areaCirculo** que tome el radio de un círculo como parámetro y devuelva su área.

10. **Generar Números Primos:** Escribe un método llamado **generarPrimos** que tome un número entero como parámetro y devuelva un array con todos los números primos menores que él.

11. **Ordenar un Array:** Escribe un método llamado **ordenarArray** que tome un array de números como parámetro y lo ordene en orden ascendente utilizando el algoritmo de selección.

12. **Calcular Fibonacci:** Escribe un método llamado **fibonacci** que tome un número entero como parámetro y devuelva el término correspondiente en la secuencia de Fibonacci.

13. **Divisores de un Número:** Escribe un método llamado **divisores** que tome un número entero como parámetro y devuelva un array con todos sus divisores.

14. **Combinaciones de una Cadena:** Escribe un método llamado **combinaciones** que tome una cadena de texto como parámetro y devuelva un array con todas las posibles combinaciones de caracteres de la cadena.

15. **Buscar Subcadena:** Escribe un método llamado **buscarSubcadena** que tome dos cadenas de texto como parámetros y devuelva **true** si la segunda cadena es una subcadena de la primera, y **false** en caso contrario.

16. **Calcular Factorial Recursivo:** Escribe un método llamado **factorialRecursivo** que tome un número entero como parámetro y devuelva su factorial utilizando recursión.

17. **Calcular Potencia Recursiva:** Escribe un método llamado **potenciaRecursiva** que tome dos números enteros como parámetros y devuelva el primero elevado a la potencia del segundo utilizando recursión.

18. **Invertir Número:** Escribe un método llamado **invertirNumero** que tome un número entero como parámetro y devuelva otro número con sus dígitos en orden inverso.

19. **Suma de Matrices:** Escribe un método llamado **sumaMatrices** que tome dos matrices bidimensionales como parámetros y devuelva otra matriz que sea la suma de las dos matrices originales.

20. **Generar Permutaciones:** Escribe un método llamado **permutaciones** que tome una cadena de texto como parámetro y devuelva un array con todas las permutaciones posibles de los caracteres de la cadena.

21. **Buscar Máximo en Matriz:** Escribe un método llamado **buscarMaximo** que tome una matriz bidimensional de enteros como parámetro y devuelva el valor máximo encontrado en la matriz.

22. **Número Amigo:** Escribe un método llamado **sonAmigos** que tome dos números enteros como parámetros y devuelva **true** si son números amigos (dos números son amigos si la suma de los divisores propios de uno es igual al otro y viceversa).

23. **Validar Email:** Escribe un método llamado **esEmailValido** que tome una cadena de texto como parámetro y devuelva **true** si es una dirección de correo electrónico válida según una estructura básica (contiene **@** y un dominio).

24. **Conteo de Palabras:** Escribe un método llamado **contarPalabras** que tome una cadena de texto como parámetro y devuelva el número de palabras en la cadena. Una palabra se define como una secuencia de caracteres separados por espacios.

25. **Números Primos en un Rango:** Escribe un método llamado **primosEnRango** que tome dos enteros **inicio** y **fin** como parámetros y devuelva un array con todos los números primos en ese rango.

26. **Determinar Año Bisiesto:** Escribe un método llamado **esBisiesto** que tome un entero representando un año como parámetro y devuelva **true** si es un año bisiesto y **false** en caso contrario.

27. **Suma de Dígitos Recursiva:** Escribe un método llamado **sumaDigitosRecursiva** que tome un número entero como parámetro y devuelva la suma de sus dígitos utilizando recursión.

28. **Calcular Serie de Fibonacci Iterativa:** Escribe un método llamado **fibonacciIterativo** que tome un número entero como parámetro y devuelva el término correspondiente en la secuencia de Fibonacci utilizando un enfoque iterativo.

29. **Verificar Palíndromo con Ignorar Mayúsculas/Minúsculas:** Escribe un método llamado **esPalindromoIgnorarCasing** que tome una cadena de texto como parámetro y devuelva **true** si es un palíndromo ignorando mayúsculas y minúsculas, y **false** en caso contrario.

30. **Generar Matriz Transpuesta:** Escribe un método llamado **transponerMatriz** que tome una matriz bidimensional de enteros como parámetro y devuelva una nueva matriz que sea la transpuesta de la original.

31. **Contar Caracteres Específicos:** Escribe un método llamado **contarCaracter** que tome una cadena de texto y un carácter como parámetros, y devuelva el número de veces que el carácter aparece en la cadena.

32. **Calcular Suma de Columnas de una Matriz:** Escribe un método llamado **sumaColumnas** que tome una matriz bidimensional de enteros como parámetro y devuelva un array con la suma de cada columna de la matriz.

33. **Rotar Array:** Escribe un método llamado **rotarArray** que tome un array de enteros y un entero **n** como parámetros, y devuelva un nuevo array con los elementos rotados **n** posiciones a la derecha.

34. **Filtrar Números Pares:** Escribe un método llamado **filtrarPares** que tome un array de enteros como parámetro y devuelva un nuevo array con solo los números pares del array original.

35. **Buscar Elemento en una Matriz:** Escribe un método llamado **buscarElementoMatriz** que tome una matriz bidimensional de enteros y un entero como parámetros, y devuelva **true** si el entero se encuentra en la matriz, y **false** en caso contrario.