Práctica 6

Debuggear

Entorno de desarrollo

PROGRAMACIÓN

1. Ejercicio 1

En España cada persona está identificada con un Documento Nacional de Identidad (DNI) en el que figura un número y una letra, por ejemplo 56999545W. Realiza un programa donde le pidas al usuario SOLO el número del dni y el programa te devuelva la letra. Para calcular la letra solo tienes que dividir el número del DNI entre 23, el resto de esta división se corresponde con la posición de la letra en el abecedario. Utiliza un array para guardar CADA letra del abecedario. Tienes que tener en cuenta que el número de DNI está compuesto por 8 dígitos.

2. Ejercicio 2

Escribe un programa que lea 10 números por teclado y que luego los muestre en orden inverso, es decir, el primero que se introduce es el último en mostrarse y viceversa.

3. Ejercicio 3

Vamos a crear un juego sencillo de "Piedra, Papel o Tijeras". En este juego, el usuario jugará contra la computadora. En este juego, el usuario elige entre Piedra (1), Papel (2) o Tijeras (3). La computadora elige al azar una de las tres opciones. Luego, se determina el ganador según las reglas clásicas del juego. El juego continúa hasta que el usuario decide salir ingresando 0.

ENTORNO DESARROLLO

4. Ejercicio 1

Si tenemos el siguiente código:

```
public class Verde {
    public static void main(String[] args) {
        int numero = 4;
        int[] array = new int[23];
        numero=8-20;
```

```
        numero=14+numero-2;
        int numero2 = -18;
        for (int i = 0; i < 27; i++) {
            int numero3=27+14;
            numero2=numero2;
        }
        for (int i = 0; i < 4; i++) {
            numero2=1*7+numero2;
        }
        for (int i = 0; i < 38; i++) {
            int numero3=246;
            numero=-3+numero+4;
        }
        for (int i = 0; i < array.length; i++) {
            array[i]=numero-26;
        }
        System.out.println("Fin");
    }
}
```

Indicar los siguientes datos:
1. El valor de “numero” al comenzar la cuarta iteración del tercer bucle.
2. La suma de “numero” y “array[3]” al final del programa.

5. Ejercicio 2

Dada el siguiente Código:

```
public class Amarillo {
    public static void main(String[] args) {
        int numero = 4;
        int[] array = new int[23];
        numero=8-20;
        numero=14+numero-2;
        int numero2 = -18;
        for (int i = 0; i < 27; i++) {
            numero2=numero2+4*2;
        }
        for (int i = 0; i < 8; i++) {
            int numero3=296;
            numero=-13+numero+7;
        }
        for (int i = 0; i < array.length; i++) {
            array[i]=numero-2;
        }
        System.out.println("Fin");
    }
```

}
Indicar los siguientes datos:
Indicar los siguientes datos:
1. El valor de “numero” al comenzar la cuarta iteración del tercer bucle.
2. La suma de “numero” y “array[3]” al final del programa.

6. Ejercicio 3

Agrega puntos de interrupción en diferentes partes del código, especialmente antes y después de la ejecución de métodos. Observa cómo cambian los valores de las variables (num, cuadrado, suma, etc.) mientras ejecutas el programa en modo de depuración.

a. Queremos saber el valor de suma cuando se inserta 7, e i = 5.
b. Y la variable cubo cuando i = 6.

```
import java.util.Scanner;



public class DebuggingExercise {

   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);



      System.out.println("Ingrese un número: ");

      int num = scanner.nextInt();



      int cuadrado = calcularCuadrado(num);

      System.out.println("El cuadrado del número es: " + cuadrado);



      int sumaCubos = calcularSumaCubos(num);
```

```
    System.out.println("La suma de los cubos hasta el número es: " + sumaCubos);
}

public static int calcularCuadrado(int x) {
    int cuadrado = x * x;
    return cuadrado;
}

public static int calcularSumaCubos(int n) {
    int suma = 0;
    for (int i = 1; i <= n; i++) {
        int cubo = calcularCubo(i);
        suma += cubo;
    }
    return suma;
}

public static int calcularCubo(int y) {
    int cubo = y * y * y;
    return cubo;
}
```

```
}
```

7. Ejercicio 4

Este programa solicita al usuario dos números, calcula la suma y el producto utilizando métodos separados, y luego imprime los resultados. Establece puntos de interrupción en lugares estratégicos, como antes y después de la entrada de datos, antes y después de llamar a los métodos, etc. Observa cómo cambian las variables (num1, num2, suma, multiplicacion, etc.) mientras ejecutas el programa en modo de depuración.

a. Necesito la trazabilidad de producto en cada una de las iteraciones de i, osea el valor producto para cada interacción de i.

```
import java.util.Scanner;


public class DebuggingExercise2 {

   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);


      System.out.println("Ingrese un número: ");

      int num1 = scanner.nextInt();


      System.out.println("Ingrese otro número: ");

      int num2 = scanner.nextInt();


      int suma = sumarNumeros(num1, num2);

      System.out.println("La suma de los dos números es: " + suma);
```

```
        int multiplicacion = multiplicarNumeros(num1, num2);
        System.out.println("El producto de los dos números es: " + multiplicacion);
    }

    public static int sumarNumeros(int a, int b) {
        int suma = a + b;
        return suma;
    }

    public static int multiplicarNumeros(int x, int y) {
        int producto = 0;
        for (int i = 0; i < y; i++) {
            producto = sumarNumeros(producto, x);
        }
        return producto;
    }
}
```

8. Ejercicio 5

Corrige el código para evitar la excepción.

```
public class NullPointerExceptionExample {
```

```
        public static void main(String[] args) {
            String palabra = null;
            int longitud = palabra.length();
            System.out.println("Longitud de la palabra: " + longitud);
        }
    }
```

9. Ejercicio 6

Dando este código con un error y tu tarea será utilizar el debugger para identificar y corregir el problema.

```
import java.util.Scanner;

public class SumaNumeros {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Programa de suma de dos números.");

        System.out.print("Ingresa el primer número: ");
        int num1 = scanner.nextInt();

        System.out.print("Ingresa el segundo número: ");
```

```
        int num2 = scanner.nextInt();

        int resultado = sumarNumeros(num1, num2);

        System.out.println("La suma de " + num1 + " y " + num2 + " es: " + resultado);

        scanner.close();
    }

    public static int sumarNumeros(int a, int b) {
        // Hay un error en la siguiente línea. ¡Usa el debugger para encontrarlo!
        int suma = a - b; // Cambiar a + b

        return suma;
    }
}
```

Instrucciones:

a. Descarga y abre este código en IntelliJ IDEA.
b. Ejecuta el programa y observa el resultado incorrecto.
c. Utiliza el debugger de IntelliJ IDEA para investigar y corregir el error en la función sumarNumeros.
d. Ejecuta el programa nuevamente y verifica que la suma sea correcta.

10. Ejercicio 7

Creamos un programa que realiza la búsqueda binaria en un array ordenado. Sin embargo, hay un error en el código que deberás identificar y corregir utilizando el debugger.

```
public class BusquedaBinaria {

   public static void main(String[] args) {

      int[] array = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};



      int elementoBuscar = 6;

      int indiceEncontrado = busquedaBinaria(array, elementoBuscar);



      if (indiceEncontrado != -1) {

        System.out.println("El elemento " + elementoBuscar + " se encuentra en el índice " + indiceEncontrado + ".");

      } else {

          System.out.println("El elemento " + elementoBuscar + " no se encuentra en el array.");

      }

   }



   public static int busquedaBinaria(int[] array, int elementoBuscar) {

      int inicio = 0;

      int fin = array.length - 1;
```

```
        while (inicio <= fin) {
            int medio = inicio + (fin - inicio) / 2;

            if (array[medio] == elementoBuscar) {
                return medio;
            } else if (array[medio] < elementoBuscar) {
                inicio = medio + 1;
            } else {
                fin = medio - 1;
            }
        }

        return -1; // Elemento no encontrado
    }
}
```

Instrucciones:

1. Descarga y abre este código en IntelliJ IDEA.
2. Ejecuta el programa y observa el resultado incorrecto.
3. Utiliza el debugger de IntelliJ IDEA para investigar y corregir el error en la función busquedaBinaria. Asegúrate de entender cómo funciona el algoritmo de búsqueda binaria y sigue el flujo del programa.

4. Ejecuta el programa nuevamente y verifica que la búsqueda binaria devuelva el resultado correcto.