Trabajo Voluntario

Entornos de desarrollo

Diagrama biblioteca y Modificación de código "Antonio el crack"

Trabajo realizado por David Morgade Gil – 1º DAM – MEDAC – PACIFICO

INTRODUCCIÓN

Diagrama de clases – Diseño de Biblioteca

En este primer apartado explicaré como he realizado el diagrama de clases para la biblioteca, para ello he utilizado la herramienta Visual Paradigm (la versión gratuita online), el diagrama realizado sería el siguiente:

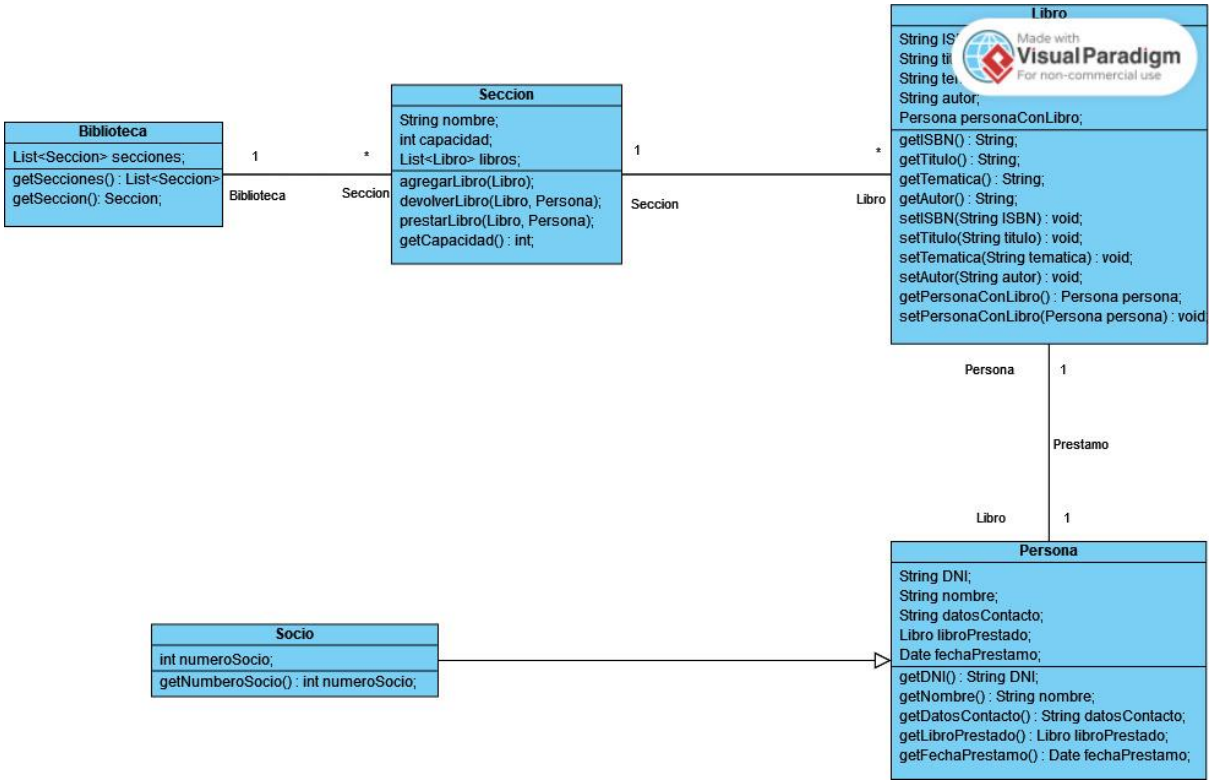


Diagrama realizado en visual paradigm

Hemos creado 5 clases diferentes, la clase Biblioteca que contendrá una lista con las diferentes secciones, después tendremos cada sección que podrá contener varios libros, la clase libro que serán los que se contienen en estas secciones con los diferentes y atributos definidos en la actividad, por último tendremos la clase persona y socio, la clase socio heredará de la clase persona, esta clase persona tendrá los diferentes atributos y métodos, a la clase socio se le añadirá el atributo numeroSocio y el resto de atributos de la clase persona.

CONTENIDO

Código de "Antonio el Crack"

Lo primero que he realizado en el código es utilizar un formateador de código como "Prettier" para VSCode, gracias a este formateador he conseguido eliminar todos los espacios e indentaciones innecesarias dentro del código, ahorrándome el trabajo de tener que hacerlo a mano (podríamos considerarlo como una herramienta CASE aunque no genere código).

Una vez he conseguido formatear el código, he probado a ejecutar el código y observo que se imprime la primera línea "numers" y puedo ir introduciendo los números, pero el problema se queda completamente pillado:

Por lo tantos vemos que el código no está funcionando correctamente.

Empiezo refactorizando y primero que todo, he decidido crear una nueva clase llamada "Números", en ella hemos creado los siguientes atributos de clase:

Tendremos el array números, donde almacenaremos los números por teclado, y dos atributos más para contar los pares e impares.

En el constructor iniciaremos directamente el array de 10 numeros y ambos contadores desde 0:

```java
// Constructor
public Numeros() {
    this.numeros = new int[10]; // Inicializamos el array con 10 posici
    this.pares = 0; // Inicializamos el contador de pares a 0
    this.impares = 0; // Inicializamos el contador de impares a 0
}
```

Después tendremos varios métodos, uno para pedir los números por teclado, este constará de un trycatch para manejar la excepción en caso de que lo que no se introduzca un numero valido:

```java
// Metodos de la clase
public void introducirNumeros(Scanner sc) { // Metodo para introducir los numeros
    System.out.println(x:"Introduce 10 numeros sin decimales:"); // Pedimos los numeros
    int i = 0; // Inicializamos el contador
    while (i < numeros.length) { // Mientras el contador sea menor que la longitud del array
        try { // Intentamos introducir un numero
            numeros[i] = sc.nextInt();
            i++;
        } catch (Exception e) { // Si no se introduce un numero entero mostramos un error y volvemos a pedir el numero
            System.out.println(x:"Error: Ingresa un número entero válido.");
            sc.nextLine();
        }
    }
}
```

Después tendremos el método para mostrar los números ordenados, en este caso clonaremos el array original ya que creo conveniente que no se mute el array original ya que el ejercicio de Antonio solo nos pide que lo mostremos ordenado, no que mutemos el array original:

```java
public void mostrarNumerosOrdenados() { // Metodo para mostrar los numeros ordenados
    int[] arrayClonado = numeros.clone(); // Clonamos el array para no modificar el original
    Arrays.sort(arrayClonado); // Ordenamos el array
    System.out.println(x:"Los numeros ordenados de menor a mayor son:");
    for (int numero : arrayClonado) { // Recorremos el array ordenado y mostramos los numeros
        System.out.print(numero + " ");
    }
    System.out.println(x:"");
}
```

Finalmente, en esta clase tendremos los dos últimos métodos, uno para contar los pares e impares y otro para mostrar por pantalla los pares e impares, usando el operador de modulo con un bucle for:

```
public void contarParesImpares() { // Metodo para contar los pares e impares
    for (int numero : numeros) { // Recorremos el array
        if (numero % 2 == 0) {
            pares++; // Si el numero es par sumamos 1 al contador de pares
        } else {
            impares++; // Si el numero es impar sumamos 1 al contador de impares
        }
    }
}

public void mostrarConteoParesImpares() { // Metodo para mostrar el conteo de pares e impares
    System.out.println("Hay " + pares + " numeros pares"); // Mostramos el conteo de pares
    System.out.println("Hay " + impares + " numeros impares"); // Mostramos el conteo de impares
}
```

Con esto la clase principal de "Trabajo", nos quedará muchísimo más limpia, ya que solo tendremos que usar los diferentes métodos de la clase Números, quedando así:

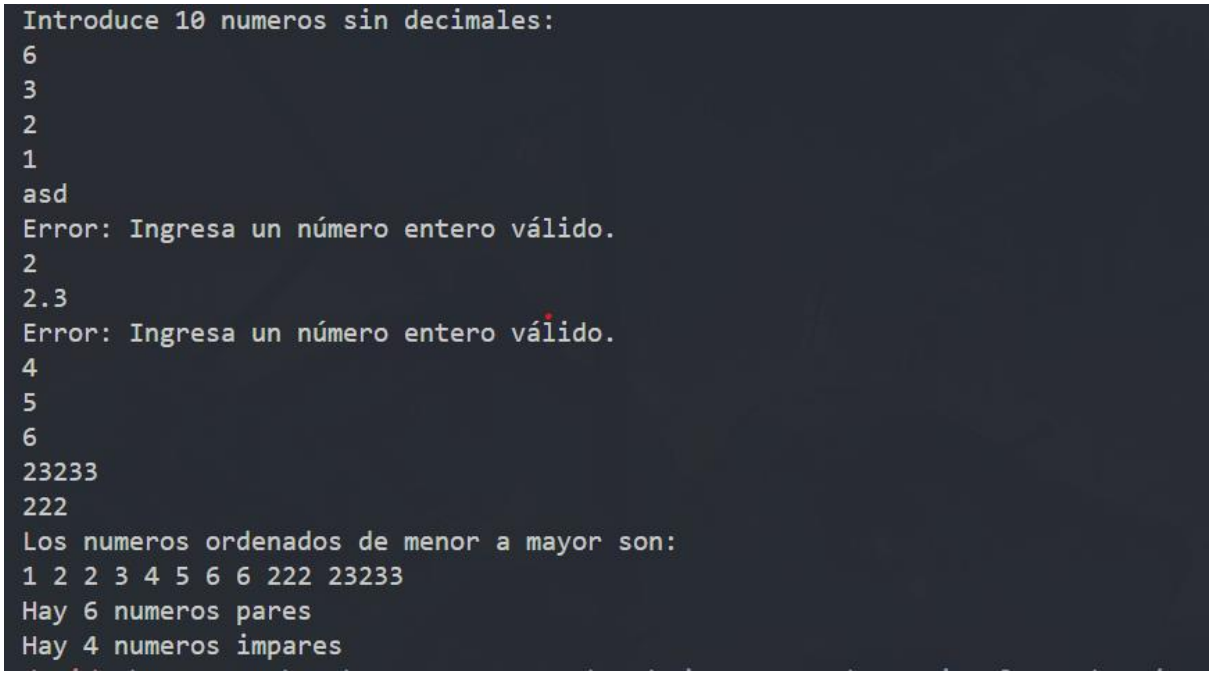
```
public class Trabajo {

    /**
     * @param args the command line arguments
     */
    Run | Debug
    public static void main(String[] args) {
        // Instanciamos la clase Numeros
        Numeros numeros = new Numeros();
        // Declaramos el Scanner
        Scanner sc = new Scanner(System.in);
        // Introducimos los numeros
        numeros.introducirNumeros(sc);
        // Mostramos los numeros ordenados
        numeros.mostrarNumerosOrdenados();
        // Contamos los pares e impares
        numeros.contarParesImpares();
        // Mostramos el conteo de pares e impares
        numeros.mostrarConteoParesImpares();
        // Cerramos el Scanner
        sc.close();
    }
```

Gracias a refactorizar en diferentes clases, usar nombres que sean convenientes y comentarios que indican que hacen cada cosa, conseguimos un código funcional y mucho más limpio.

CONCLUSIÓN

Finalmente realizaremos las diferentes pruebas para ver si nuestro código funciona, vamos a ejecutarlo para ver la diferencia con el código original:



Vemos que la aplicación funciona correctamente, además vemos que nuestra excepción esta manejando cuando se introduce un valor no válido por teclado.

BIBLIOGRAFÍA

Recursos utilizados para la realización del trabajo

- Stackoverflow: Para resolución de dudas respecto al código y buenas practicas.

- VisualParadigm: Herramienta utilizada para realizar el diagrama de clases.

- Oracle: Revisión de la documentación de Java.

- Discoduroderoer: Profesor de programación que ayuda de manera altruista por internet y que me a ayudado mucho a aprender sobre Java y en general de la programación orientada a objetos.

- CleanCode: Libro escrito por Robert C. Martin que para mi es la biblia de las buenas prácticas sobre todo para Java.