Nombre Alumno/s: Vargas Portillo Jonatan Ezequiel DNI: 41442040

Nombre Profesor: Mettini, Aldo, Airaldi Lezcano Andrea
Grupo Laboratorio: TP: 3

Fecha de entrega: 10/09/2023

En este trabajo se fue viendo más a detalle el uso del doble encapsulamiento que en el trabajo anterior solo algunos ejercicios tenían estas características de lo que es POO.
Esto lo podemos ver en la mayoría de ejercicios pero si tuviera que elegir uno para mostrar sería el siguiente. La clase **Rectángulo** posee estos métodos en el cual podemos observar el doble encapsulamiento ya que evitamos acceder a los atributos directamente y hacemos uso de los accessors .

```
    // Método para calcular la superficie del rectángulo
    public double superficie() {
        return this.getAlto() * this.getAncho();
    }

    // Método para calcular el perímetro del rectángulo
    public double perimetro() {
        return 2 * (this.getAlto() + this.getAncho());
    }
```

Otra características de la POO es que podemos observar cómo usamos los métodos a través de lo que son los mensajes. Un mensaje es una petición enviada a un objeto para que este se comporte de determinada manera como lo podemos ver en algunos ejecutables como el siguiente.
En la clase **RegistroCivil** se ve cómo se utilizan los mensajes para la comunicación del objeto con algún método en este caso la petición al método **casarseCon(mujer)** que utiliza el objeto **hombre**.

```
    private static void casarseCon(Hombre hombre, Mujer mujer){
        if(hombre.getEsposa() == null && mujer.getEsposo() == null){
            hombre.casarseCon(mujer);
            hombre.casadaCon();
        }else{
            System.out.println(x:"Ya se encuentran csados");
        }
    }
}
```

Algo que se utilizó en algunos ejecutables es el uso del menú y con la implementación del menú tuve que realizar unos métodos estáticos para que el menú quedará más legible, para que no quedara muy sobrecargado de código.

Esta es la implementación de uno de los menu de la clase **Puntos** en el cual tuve que implementar algunos métodos estáticos para que el menú no quedara sobrecargado.

```java
public static void main(String[] args){
    Scanner teclado = new Scanner(System.in);
    Punto[] puntos = new Punto[2];
    int contadorPuntos = 0;

    int opcion;

    do {
        System.out.println(x:"\t\t\t\t\t----------MENU----------:");
        System.out.println(x:"\t\t\t\t\t1. Crear Puntos");
        System.out.println(x:"\t\t\t\t\t2. Calcular la distancia entre dos puntos");
        System.out.println(x:"\t\t\t\t\t0. Salir");
        System.out.print(s:"\t\t\t\t\tSeleccione una opción: ");
        opcion = teclado.nextInt();

        switch (opcion) {
            case 1:
                crearPunto(contadorPuntos, puntos,teclado);
                contadorPuntos++;
                break;
            case 2:
                calcularDistancia(contadorPuntos, puntos);
                break;

            case 0:
                System.out.println(x:"Saliendo del programa.");
                break;

            default:
                System.out.println(x:"Opción no válida. Intente de nuevo.");
        }
    } while (opcion != 0);

    teclado.close();
}
```

Los métodos estáticos son los que no dependen de un objeto particular de la clase, se les puede invocar sin necesidad de crear objetos. Estos fueron algunos de los métodos estáticos utilizados para la elaboración del menú de la clase ejecutable **Puntos**

```
    public static void crearPunto(int contador, Punto[] puntos, Scanner teclado){
        if (contador < 2) {
            puntos[contador] = crearPunto(teclado);
            System.out.println(x:"Punto creado exitosamente.");
        } else {
            System.out.println(x:"Ya has creado el número máximo de puntos (2).");
        }
    }

    public static void calcularDistancia(int contador, Punto[] puntos){
        if (contador >= 2) {
            double distancia = puntos[0].distanciaA(puntos[1]);
            System.out.println("La distancia entre los puntos es: " + distancia +"\n\n");
        } else {
            System.out.println(x:"Debes crear al menos dos puntos primero.");
        }
    }

    public static Punto crearPunto(Scanner scanner) {
        System.out.print(s:"Ingrese la coordenada X: ");
        double x = scanner.nextDouble();
        System.out.print(s:"Ingrese la coordenada Y: ");
        double y = scanner.nextDouble();
        return new Punto(x, y);
    }

}
```

También observamos la relación entre clases, esto nos indica cómo se comunican los objetos de esas clases entre sí, podemos observar que en las clases necesitamos declarar un atributo de la clase con la cual queremos relacionar y también declarar otros constructores que acepten el envío de ese tipo de argumento como lo podemos observar en el siguiente ejercicio.
En la clase **Persona** podemos observar como uno de los atributos es de clase **Calendar**

```
import java.util.*;

public class Persona {
    private int nroDni;
    private String nombre;
    private String apellido;
    private Calendar fechaNacimiento; // Fecha de nacimiento de la persona
```

se tuvo que agregar un nuevo método constructor para que uno de sus argumentos sea de la clase **Calendar**.

```
// Constructor para inicializar los atributos de la persona con fecha de nacimiento
public Persona(int p_dni, String p_nombre, String p_apellido, Calendar p_fecha) {
    this.setDNI(p_dni);
    this.setNombre(p_nombre);
    this.setApellido(p_apellido);
    this.setFechaNacimiento(p_fecha);
}
```

También se tuvieron que declarar sus respectivos getter y setter

```java
// Método para obtener la fecha de nacimiento de la persona (getter)
public Calendar getFechaNacimiento() {
    return fechaNacimiento;
}

// Método para asignar la fecha de nacimiento de la persona (setter)
public void setFechaNacimiento(Calendar fechaNacimiento) {
    this.fechaNacimiento = fechaNacimiento;
}
```

Como se realizó el cambio del atributo de la clase **Calendar** el cual antes correspondía a un atributo del tipo **int** se tuvo que modificar los setter y getter correspondiente al atributo anterior para que siga funcionando.

```java
private void setAnioNacimiento(int p_anio){
    this.fechaNacimiento = new GregorianCalendar();
    this.fechaNacimiento.set(Calendar.YEAR, p_anio);
}

// Método para obtener el año de nacimiento de la persona (getter)
public int getAnioNacimiento(){
    return this.fechaNacimiento.get(Calendar.YEAR);
}
```