Mejorando el Encapsulamiento y Pruebas Unitarias en el Sistema de Gestión de Donantes de Sangre de la Cruz Roja



Valentina Montezuma González (2242058)

- (-)

- (-)

- (-)

Universidad del Valle

Mejorando el Encapsulamiento y Pruebas Unitarias en el Sistema de Gestión de Donantes de Sangre de la Cruz Roja

Valentina Montezuma González (2242058)

- (-)

- (-)

- (-)

Profesor

Víctor Andrés Bucheli

Taller Programación Orientada a Objetos

Universidad del Valle

Facultad de Ingeniería

Programación Orientada a Objetos

Santiago de Cali

2025

Taller Encapsulamiento y Pruebas Unitarias

Análisis del diseño de clases, código modificado y problemas resueltos.

Como problema inicial que identificamos fue que los atributos dirección, donorId, name, address, distric, bloodType y number de la clase Donor eran públicos aun tratándose de datos de información delicada, lo que no cumplía con el principio de encapsulamiento, por lo que realizamos los siguientes cambios:

1. Pusimos los atributos privados.
2. Se agregaron setters y getters para acceder a los atributos y modificarlos de una manera controlada y sin acceder directamente a ellos.
3. Implementamos validaciones en los setters para asegurar que los datos sean correctos (El número de teléfono mediante la API y el tipo de sangre usando los tipos existentes). Esto ya que anteriormente había errores al tomar el celular y en el caso de la sangre se podía poner cualquier letra haciendo que el programa no fuera preciso.
4. Movimos el método trim de la clase Donor ya que podía usarse más allá de la misma, por lo que se fue a una clase utilitaria llamada Utils.

Por parte de la clase BlooDatabase, los cambios realizados fueron en función de que anteriormente accedía a los datos privados de Donor, así que se asigno que en su lugar accediera a los getters para evitar errores y respetar el encapsulamiento. Tambien se agregaron los métodos de:

1. getBloodTypePercentages(): Que permite calcular el porcentaje de cada tipo de sangre en función de las personas que la tienen.
2. getMostCommonBloodType(): Que regresa el tipo de sangre más común.
3. Porcentaje de la población de cada tipo de sangre
4. Que tipo de sangre tiene la mayoría de la gente

Y finalmente la clase Utils que se creo para almacenar el método trim, pues este se encontraba anteriormente en Donor lo cual limitaba su uso.

Descripción de la reutilización de código.

El método trim, como se menciono anteriormente, se movió a una clase totalmente nueva, de manera que se facilita su reutilización en cualquier parte del proyecto y deja de estar limitado para una sola clase.

Se implemento una nueva librería, la cual fue <map> cuyo propósito fue proporcionar un contenedor asociativo que almacena pares clave-valor, esto para contar y agrupar datos. El uso especifico en el código fue en getBloodTypePercentages y getMostCommonBloodType, donde se uso std::map para contar cuantos donantes hay en cada tipo de sangre.

Descripción de la API integrada.

Implementación de interfaces publicas

Los getters y setters se crearon con el propósito de actuar como interfaz publica, pues son los que se usan a lo largo de BloodDatabase para acceder a los a tributos privados de Donor. Es decir, permiten su reutilización y a la vez mantener el encapsulamiento de manera correcta.

Código de las pruebas unitarias y resultados de las mismas