Ejercicio 3

Considere el siguiente extracto de código de un framework para construir clientes de correo electrónico, en particular el módulo de definición de filtros de correo. Este módulo del framework permite definir un filtro para correos electrónicos que siempre verifica si debe agregarse o no el correo a un inbox dado, y en caso positivo lo agrega.

Por ejemplo, podría definirse un filtro que verifique mediante el método isEmailAllowed si un correo incluye en el asunto el string "Objetos 2" y, en tal caso, lo agrega mediante el método addEmail a la bandeja de entrada marcándolo como importante.

```
1    abstract class EmailFilter{
2        private EmailInbox emailInbox;
3
4        public EmailFilter(EmailInbox emailInbox){
5            this.emailInbox = emailInbox;
6        }
7
8        public filterEmail(Email anEmail){
9            if (this.isEmailAllowed(anEmail)){
10               this.addEmail(anEmail);
11           }
12       }
13
14       public abstract Boolean isEmailAllowed(Email anEmail);
15       public abstract void addEmail(Email anEmail);
16   }
```

Responda las siguientes preguntas, basándose en el subconjunto de clases y métodos que conoce del framework:

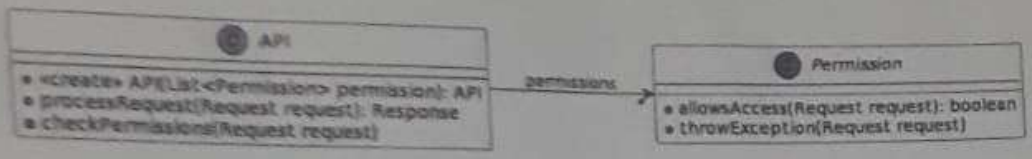
1. *¿El comportamiento variable del framework (hotspots), está implementado mediante herencia o composición? Justifique su respuesta.*
2. *Indique la/s línea/s donde encuentra inversión de control. Justifique su respuesta.*
3. *El método isEmailAllowed, ¿es un hot spot? ¿Por qué?*

If 8% > mayo y 10%.

else 11% > agosto y 15%.

Ejercicio 3: Frameworks

Considere el siguiente extracto de código y diagrama de clases UML de un framework para desarrollo de API Rests, en particular el módulo de autorización de acceso.



Este framework provee una clase, API, que define la lógica invariante necesaria para:
(i) procesar un request HTTP, siempre verificando que se cumplan con todos los permisos de acceso, y,
(ii) retornar el estado HTTP 200 OK para el caso en que se cumplan todos los permisos de acceso o retornar el estado HTTP 403 FORBIDDEN en caso de que alguno de los permisos no se cumpla.

Es responsabilidad de quienes utilizan el framework implementar la interfaz *Permission* para definir el comportamiento específico para chequear los permisos de accesos (por ejemplo, para cualquier usuario registrado, sólo para usuarios administradores, etc). Eso incluye también la lógica para disparar AccessDeniedException o alguna subclase de ésta.

```
public class API {
  private List<Permission> permissions;

  public API(List<Permission> permissions){
    super();
    this.permissions = permissions;
  }

  public Response processRequest(Request request){
    ...
    try{
      this.checkPermissions(request);
    }
    catch (AccessDeniedException e){
      return new Response(HTTPStatus.403);
    }
    return new Response(HTTPStatus.200);
  }

  private void checkPermissions(Request request) throws AccessDeniedException{
    for (Permission permission: this.getPermissions()){
      if (!permission.allowsAccess(request)){
        permission.throwException(request);
      }
    }
  }
}

public interface Permission{
  public boolean allowsAccess(Request request);
  public void throwException(Request request) throws AccessDeniedException;
}
```

Tareas:
Responda las siguientes preguntas, basándose en el subconjunto de clases y métodos del framework presentado anteriormente:

1. ¿El comportamiento variable del framework (hotspots), está implementado mediante herencia o composición? Justifique su respuesta.
2. ¿Observa hook methods? ¿Cuáles?
3. ¿Qué parte de lo que se dice anteriormente respecto al framework se corresponde con el frozenspot?
4. En lo que se describe anteriormente y lo que se indica debe hacer quien utiliza el framework, ¿observa inversión de control? ¿donde?

Ejercicio 3 - Frameworks

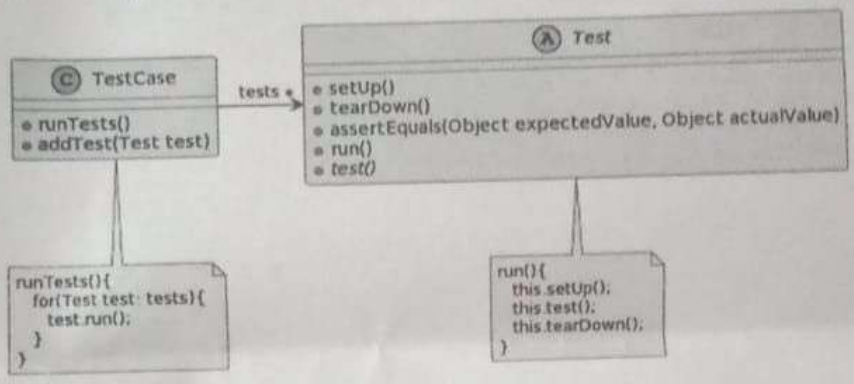
OO2 Parcial 2da fecha - 1/7/2023

Consideremos un framework muy simple para automatizar pruebas de software.

El framework permite crear casos de prueba. Cada caso de prueba ("test case" en inglés) incluye una o varias pruebas ("test" en inglés). Al correr un caso de prueba, se corren una a una, en secuencia, todas las pruebas incluidas en el caso. El orden en el que se ejecutan las pruebas es aleatorio. Si una prueba falla (es decir, no pasa) el caso se interrumpe. Un caso de pruebas pasa si pasan todas las pruebas que incluye. Las pruebas pueden requerir preparación (set up, en inglés) y desarmado (tear down en inglés). El framework se asegura de que la preparación (si existe) y el desarmado (si existe) se efectúan antes y después de cada test. El framework ofrece un conjunto fijo de asserts (chequeos) que pueden utilizarse en las pruebas.

El programador puede definir nuevas pruebas, puede combinar pruebas en casos de manera arbitraria, y puede correr los casos. Al definir una prueba, el programador determina cómo hacer la preparación (si es necesaria), los pasos de la prueba (usando los asserts que provee el framework), y el desarmado (si es necesario).

Considere el siguiente diagrama de clases que documenta el framework.



Para crear programas (casos de prueba) con el framework, el programador debe:

1. Programar una subclase de Test por cada prueba. En esa subclase debe programar el método abstracto test() utilizando el método assertEquals() heredado, el cual dispara una excepción en caso de recibir valores diferentes, haciendo que la prueba y todo el caso de prueba fallen. Si lo considera necesario, además, puede implementar los métodos setUp() y tearDown(), para hacer algo para preparar y/o desarmar el test.
2. Crear instancias de TestCase y agregarle tests (instancias de sus clases concretas de Test).
3. Ejecutar los casos enviándoles el mensaje runTests()

Tareas:
Responda las siguientes preguntas, basándose en el subconjunto de clases y métodos del framework presentado anteriormente:

1. Indique, para cada uno de los siguientes ítems, si es parte del frozen spot, si es parte de algún hotspot o si no corresponde a ninguno de los dos.
   a. El framework se asegura de que la preparación y el desarmado se efectúan antes y después de cada test.
   b. Al definir una prueba, el programador determina cómo hacer la preparación (si es necesaria).
   c. La clase Test es abstracta.
   d. Al correr un caso de prueba se corren todas las pruebas incluidas en el caso.
   e. Al definir una prueba el programador define los pasos de la prueba (usando los asserts que provee el framework).
2. ¿Observa métodos gancho? ¿Cuáles?
3. ¿Observa inversión de control? ¿Dónde?
4. ¿Caracterizaría el framework como caja blanca o caja negra? ¿Por qué?