Programación Orientada a Objetos.

Práctica 06.

Objetivos:

- Implementar el patrón de diseño Singleton.
- Practicar clases abstractas e interfaces.
- Implementar una estructura de paquetes en el sistema bancario.

Antes de iniciar los ejercicios, baje de la página del curso, sección Material, el archivo Practica06.zip al escritorio y descomprímalo para obtener los archivos necesarios para esta práctica.

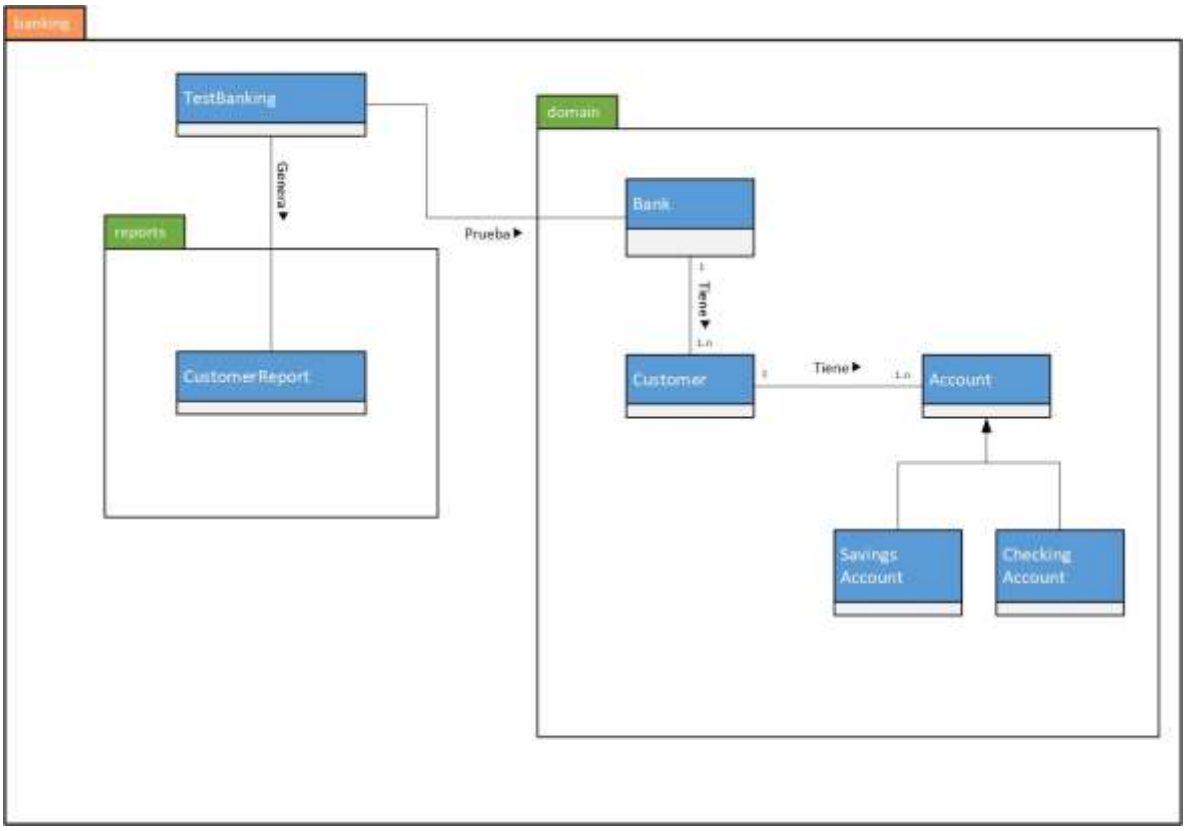
Ejercicio 1. Implementar el patrón de diseño Singleton en la clase Bank y establecer una estructura de paquetes para el Sistema Bancario.

Implementación del patrón de diseño Singleton en la clase Bank.

1) Abra el proyecto de netbeans BankLtd que se ha venido desarrollando hasta ahora.
2) Borre la clase TestBanking de la versión anterior del proyecto.
3) En la clase Bank, defina una variable **estática** de clase Bank, llamada bank con modificador de acceso private.
4) Haga **privado** el constructor de Bank.
5) Incluya un inicializador estático que cree una instancia de la variable estática bank.
6) Cree el método estático **public static Bank getBank(),** que entregue la instancia única bank.

Creación de nuevos paquetes.

Estudie el siguiente diagrama de paquetes de UML y a continuación implemente la estructura siguiendo las instrucciones.



1) Utilizando las facilidades de Netbeans, dentro del paquete banking, cree dos subpaquetes llamados domain y reports.

2) Mueva, utilizando la facilidad de refactor de Netbeans, las clases Bank, Customer, Account, SavingsAccount y CheckingAccount al paquete banking.domain. Note que Netbeans inserta los postulados package necesarios.

3) Copie la clase proporcionada en el material de la práctica llamada TestBanking.java al paquete banking y estudie su contenido.

4) Copie la clase proporcionada en el material de la práctica llamada CustomerReport.java al paquete banking.reports y estudie su contenido.

5) En el programa CustomerReport busque la línea con el comentario /** Paso1: obtenga el objeto bank **/ y modifique el código para incluir la llamada al método estático getBank() que entrega la instancia singleton del banco.

6) Ejecute el programa TestBanking y estudie el resultado.

Ejercicio 2. Trabajar con interfaces y clases abstractas.

1) Cree un nuevo proyecto de Netbeans llamado Practica06 de tipo Java Application. **Deseleccione la casilla Create Main Class**.

2) Dentro del proyecto Practica06, cree un paquete llamado ejercicio2.

3) En el paquete ejercicio2 cree la clase abstracta Animal, como sigue:

   a) atributo protected int llamado legs, que registre el número de patas de un animal.
   b) constructor público que inicialice el atributo legs.
   c) método abstracto eat de tipo void.
   d) método no abstracto walk tipo void que despliegue un mensaje indicando como camina el animal (incluya el número de patas).

4) Cree la clase Spider, como sigue:

   a) subclase de Animal.
   b) constructor sin argumentos que llame a la superclase pasando el valor 8 como número de patas.
   c) implementación del método eat al gusto, e.g. desplegando un mensaje a propósito.
   d) implementación del método walk al gusto.

5) Cree la interfaz Pet, que incluya los métodos:

   a) public String getName();
   b) public void setName(String name);
   c) public void Play();

6) Cree la clase Cat que extienda Animal e implemente Pet, como sigue:

   a) atributo de clase String para guardar el nombre del gato.
   b) constructor que tome un parámetro tipo String que especifique el nombre del gato y llame al constructor de la superclase, pasando el valor 4 como número de patas.
   c) otro constructor sin argumentos, que llame al constructor del inciso b) usando la palabra clave this y pasando un String nulo como argumento.
   d) implementación de los métodos de la interfaz Pet.
   e) implementación de los métodos eat() y play().

7) Cree la clase Fish que extienda Animal e implemente Pet, como sigue:

   a) constructor sin argumentos que llame al constructor de la superclase pasando 0 como número de patas.
   b) implemente los métodos de la interfaz Pet.
   c) substituya (override) los métodos de la clase Animal para especificar que el pez no camina y no tiene patas.

8) Copie el archivo TestAnimals.java proporcionado en el material de la práctica al paquete ejercicio2 y estudie su contenido.

9) Ejecute TestAnimals y observe el resultado.