UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ciencias

Equipo NullPointerException:
Adrián Aguilera Moreno - 421005200
Diego Angel Rosas Franco - 318165330
Marco Antonio Rivera Silva - 318183583

Modelado y programación

Proyecto 01

Instrucciones de instalación, compilación y ejecución.

Se dará por hecho que el usuario sabe moverse en terminal.

Requerimientos previos:

- Se debe contar con una versión de Java 8 o superior en su computadora. De preferencia la versión más reciente.

Ejecución del proyecto:

- Si está leyendo esto significa que desempaquetó con éxito el proyecto.
- Abra su terminal y diríjase a la ruta donde desempaquetó el proyecto.
- Una vez estando en la ruta

  `Proyecto01_NullPointerException`,

  diríjase a

  `Proyecto01_NullPointerException/src/fciencias/modelado/`
- Ejecute: "`javac CheemsMart.java`", esto generará los .class del proyecto.
- Ejecute: "`java CheemsMart`", esto ejecutará el proyecto mostrándole el menú solicitado para la practica.

Ingreso al sistema:

A continuación se muestra una tabla de usuarios en la base de datos del sistema.

Username	Nombre Completo	Contraseña
Ross	Rosa Victoria Villa Padilla	Zeldaqwerty
arturGod	Arturo Lemus Pablo	Password
fulano	Fulano Federico Fernandez Tragedio	diosEstaAqui123
Direccion	Número de cuenta bancaria	País
Calle Olivos 145, Azulejos, Madrid	4510467245	España
5ta Avenida No 45, Chicago	1234543210	Estados Unidos
Calle Chongus 132, Colonia De las Tragedias	5578551018	Mexico

Como se aprecia, las contraseñas son muy "seguras", excelentes para esta prueba.
Para poder ingresar al sistema necesitará el Username y la Contraseña.

Justificación de uso de patrones:

Strategy

Utilizamos Strategy en en la parte de cambiar el idioma debido a que lo resuelve de una forma conveniente ya que teníamos pensado utilizar State, sin embargo notamos que a pesar que en ambos tendríamos que modificar el idioma en tiempo de ejecución había una gran desventaja al usar State, por ejemplo limitar los estados con múltiples condiciones para evitar que se tuviera más de un estado a la vez, en cambio Strategy podíamos controlar que en efecto solo tuviera un idioma de una manera mucho más sencilla y efectiva, además de que sería sumamente sencillo agregar nuevos idiomas en Strategy que en State.

Decorator

Implementamos el patrón de diseño Decorator en la parte de descuentos, pues los productos pueden aplicar a uno o más descuentos, incluso del mismo tipo (al menos no hay una restricción para esto) de manera conveniente en nuestra implementación, donde el `Producto` original sin modificaciones en precios es el centro del decorador.

Proxy

Decidimos implementar Proxy en la parte del Servidor pues en el PDF del proyecto se dice: *Por cuestiones de seguridad, es imperiosamente necesario que la tienda virtual no cargue directamente el catálogo real, pues si llega a presentar vulnerabilidades de seguridad, cualquiera podría cambiarlo.* Pensamos que para solucionar esto sería bueno tener un objeto Proxy Servidor que nos sirviera de intermediarios con el Servidor Real, por ahora esto no se aprovechó al máximo para los Requerimientos del proyecto, pero lo pensamos a la larga, pues seguramente se harían varias peticiones al Servidor, y justo para preservar la seguridad de la tienda y sus datos el objeto intermediario es una mejor opción.

Otros patrones que se pudieron usar:

Iterator

Pensamos que en vez de usar Proxy en el Servidor podríamos usar Iterator, y así regresar un iterador de las estructuras que necesitaramos, sin embargo, a la larga esto podría hacer rigido el sistema, en el sentido de que cuando se quiera agregar seguridad al programa, se debería dar una solución especifica o agregar un nuevo patrón. En cambio con Proxy ya ayudamos a que la seguridad sea mantenible.

Singleton

Nos percatamos que podíamos haber usado Singleton en la clase Servidor, pues de esa forma tendríamos un servidor centralizado en el que tendríamos a todos nuestros usarios y el catálogo de la tienda. Sin embargo, vimos que esto podría causar un problema de rigidez, pues a futuro es posible que la tienda planeé tener servidores a lo largo del mundo para mejorar la velocidad de respuesta acorde a la región en la que se encuentre el usuario (teoricamente hablando), y en cada servidor, regional por ejemplo, tener un catálogo distinto a futuro, ya que actualmente solo se tiene un catálogo en el mismo idioma para todas las regiones. Debido a esto es que Servidor cuenta con un identificador único, para asi poder diferenciarlo de los demás servidores futuros.

Template

Pensamos en usar template en los idiomas, de hecho al final se quedó esta idea que tuvimos, la cual fue tener un menuPrincipal() en MenuCompra donde se llamarían ciertos métodos con mensajes al

usuario en su idioma, y justo los MenuCompra¡Idioma¿ estarían definiendo estos métodos o "pasos". Pero el problema aqui era que menuPrincipal no era secuencial y debido a eso no se podría aplicar el patrón a menos que lo hicieramos de forma secuenciañ.

Diagrama UML: