ORDER SYSTEM

REQUERIMIENTOS

RF 1. Registrar:

- Restaurantes, se debe poder ingresar la información respectiva de un restaurante y poder registrarlo. Debe mostrar un mensaje de que el restaurante fue agregado exitosamente al registrarlo o un mensaje de que no pudo ser agregado si el restaurante ya existía.
- Clientes, se debe poder ingresar la información respectiva de un cliente y poder registrarlo. Debe mostrar un mensaje de que el cliente fue agregado exitosamente al registrarlo o un mensaje de que no pudo ser registrado si el cliente ya existía.
- Productos, se debe poder ingresar la información respectiva de un producto y poder registrarlo. Debe mostrar un mensaje de que el producto fue registrado exitosamente al agregar o un mensaje de que no pudo ser registrado si el producto ya existía.
- Ordenes, se debe poder ingresar la información respectiva de una orden y poderla registrar. Debe mostrar un mensaje de que la orden fue agregado exitosamente al registrarla o un mensaje de que no pudo ser registrada si la orden ya existía.

RF 2. Actualizar:

- Datos del restaurante dado su identificación que es el nit, donde debe tener la posibilidad de actualizar todos sus atributos, donde se mostrara un mensaje de que se pudo actualizar efectivamente los datos, o un mensaje de que salió algo mal.
- Datos de un producto dado su identificación que es el código, donde debe tener la posibilidad de actualizar todos sus atributos, donde se mostrara un mensaje de que se pudo actualizar efectivamente los datos, o un mensaje de que salió algo mal.
- Datos de un cliente dado su número identificación, donde debe tener la posibilidad de actualizar todos sus atributos, donde se mostrará un mensaje de que se pudo actualizar efectivamente los datos, o un mensaje de que salió algo mal.
- Datos de una orden dada su identificación dado su código, donde debe tener la posibilidad de actualizar todos sus atributos, donde se mostrará un mensaje de que se pudo actualizar efectivamente los datos, o un mensaje de que salió algo mal.

RF 3. Permitir cambiar el estado de un pedido entre solicitado, en proceso, enviado y entregado. Se puede cambiar el estado del pedido hacia adelante, pero no hacia atrás y actualizar su estado después de haberlo cambiado, retornando un mensaje si pudo ser actualizado o no.

RF 4. Guardar toda su información a través de la socialización. Toda información del sistema de ordenes debe guardarse a través de serialización, tanto restaurantes, como clientes, productos y órdenes.

RF 5. Exportar un archivo csv de pedidos, con el separador que elija el usuario y tiene que estar ordenado por los siguientes criterios:

Nit del restaurante ascendente, documento del cliente descendente, fecha del pedido ascendente y código del producto ascendente

RF 6. Tener una opción que permita listar en pantalla todos los restaurantes en orden alfabético ascendente, otra opción que permita listar en pantalla todos los clientes de un respectivo restaurante en orden de su número telefónico descendente, desplegando toda la información de cada uno.

RF 7. Tener una opción que permita buscar eficientemente un cliente dado un nombre e indicar el tiempo que tardó la búsqueda. Para ello necesita el nit del restaurante al que pertenece para después con una búsqueda binaria retornar un mensaje si lo encuentra y el tiempo de ejecución o si no lo encuentra.

RF 8. Verificar que todos los productos de un pedido pertenezcan al restaurante que el cliente eligió para hacer el pedido, retornando un mensaje si el producto pertenece al restaurante o en efecto si no.

RF 9. Tener al menos 2 pruebas unitarias automáticas por cada clase del modelo, donde se evalué si los métodos funcionan correctamente.

RF 10. Implementar al menos dos algoritmos de ordenamiento.

RF 11. Importar:

- Datos de un archivo csv con información de restaurantes, dado el nombre del archivo retornando un mensaje según si existe o no.
- Datos de un archivo csv con información de clientes, dado el nombre del archivo retornando un mensaje según si existe o no.
- Datos de un archivo csv con información de productos, dado el nombre del archivo retornando un mensaje según si existe o no.
- Datos de un archivo csv con información de órdenes, dado el nombre del archivo retornando un mensaje según si existe o no.

Configuración de los Escenarios

Nombre	Clase	Escenario
setupStage1	AppTest	Object of class App
setupStage1	AppTest	Object of class App Object of class Restaurant with name="Cheers", nit="901,458,652-7" and manager="Justin Ross";
setupStage1	RestaurantTest	Object of class Restaurant with name="Cheers", nit="901,458,652-7" and manager="Justin Ross"
setupStage2	RestaurantTest	Object of class Restaurant with name="Cheers", nit="901,458,652-7 "and manager="Justin Ross" Object of class Client with nameC="Michael", lname "Jasckson", id_number="1.1193.033.576", id_type="Cedula", phone="3043808681" and adress="Carrera 92#54-42"
setupStage1	ClientTest	Object of class Client with name="Michael", lname =" Stauber ", id_number="1.1193.033.576", id_type="Cedula", phone="3043808681" and adress="Carrera 92#54-42".
setupStage2	ClientTest	Object of class Client with name="Jack ", lname="Sparrow", id_number="1.123.653.321", id_type="Cedula", phone="3531308681" and adress="Carrera 21#55-42"
setupStage1	ProductTest	Object of class Product with code="A49F", name="Coca-cola", description="Soda", cost=1500 and nit="901,458,652-7";
setupStage2	ProductTest	Object of class Product with code="AG43F", name="Pizza", description="Napolitan pizza", cost=32000 and nit="321,434,652-2"
setupStage1	OrderTest	Object of class Product with codeP="A49F", name="Coca-cola", description="Soda", cost=1500 and nitP="902,458,652-7"
setupStage2	OrderTest	Object of class Product with codeP="4349F", name="Chocorramo", description="Cake", cost=1000 and nitP="202,358,652-7" Object of class Order with code="A23F", date=new Date(), code_client="1.323.323.429" and nit="901,458,652-7"

Diseño de Casos de Prueba

Test Objective: Verify that the App and searchRestaurant methods of the App class works correctly, checking the information that passes through the parameter meets the requirements to be able to create and search correctly.

Clase	Método	Escenario	Valores de Entrada	Resultado
App	App	setupStage1	Anyone	The restaurant has been added successfully. Each of the attributes of the new restaurant is correctly assigned.
App	searchClient	setupStage2	name="Cheers" nit="901,458,652-7" manager="Justin Ross"	The client was founded . The restaurant identification number matches one of the restaurant list.

Test Objective: Verify that the Restaurant and searchClient methods of the Restaurant class works correctly, checking the information that passes through the parameter meets the requirements to be able to create and search correctly.

Clase	Método	Escenario	Valores de Entrada	Resultado
Restaurant	Restaurant	setupStage1	name="Cheers" nit="901,458,652-7" manager="Justin Ross"	The restaurant has been added successfully Each of the attributes of the new restaurant is correctly assigned the information passed by parameter.
Restaurant	searchClient	setupStage2	name="Cheers" nit="901,458,652-7" manager="Justin Ross" nameC="Michael" lname=" Jasckson" id_number="1.1193.033.576" id_type="Cedula"; phone="3043808681";	The client was founded . The client identification number matches one of the client list.

			adress="Carrera 92#54-42";	

Test Objective: Verify that the Client and setId_type methods of the Client class works correctly, checking the information that passes through the parameter meets the requirements to be able to create and change correctly.

Clase	Método	Escenario	Valores de Entrada	Resultado
Client	Client	setupStage1	name="Michael " lname=" Jasckson" id_number="1.1193.033.576" id_type="Cedula" phone="3043808681" adress="Carrera 92#54-42"	The client has been added successfully Each of the attributes of the new client is correctly assigned the information passed by parameter.
Client	setId_type	setupStage2	name="Jack Stauber" lname=" Stauber" id_number="1.123.653.321" id_type="Cedula" phone="3531308681" adress="Carrera 21#55-42"	The type identification of the client has been changed successfully. The attribute was update.

Test Objective: Verify that the Product and setCost methods of the Product class works correctly, checking the information that passes through the parameter meets the requirements to be able to create and the cost of the product is corretly

Clase	Método	Escenario	Valores de Entrada	Resultado
Product	Product	setupStage1	code="A49F" name="Coca-cola" description="Soda" cost=1500 nit="901,458,652-7";	The product has been added successfully Each of the attributes of the new product is correctly assigned the information passed by parameter.
Product	setCost	setupStage2	code="AG43F" name="Pizza" description="Napolitan pizza; cost=32000 nit="321,434,652-2"	The cost of the product can not be negative The cost of the product was not changed, it remains the same

Test Objective: Verify that the Order and setCoder_Client methods of the Order class works correctly, checking the information that passes through the parameter meets the requirements to be able to create and change correctly.

Clase	Método	Escenario	Valores de Entrada	Resultado
Order	Order	setupStage1	codeP="A49F" name="Coca-cola" description="Soda" cost=1500 nitP="902,458,652-7"	The order has been added successfully Each of the attributes of the new order is correctly assigned the information passed by parameter.
Order	SetCode_client	setupStage2	codeP="4349F" name="Chocorramo" description="Cake" cost=1000 nitP="202,358,652-7" code="A23F" date=new Date() code_client="1.323.323.429" nit="901,458,652-7"	The code of the client was successfully changed The attribute was update.