

Trabajo Práctico Especial 2022

Grupo 3a

Testing: Lucas de Lellis, Renzo Agustin Romeo.

Código: Camila Belen Cacace, Teo Ramos Kees.

Enlace al video: <https://www.youtube.com/watch?v=mZ9fMtrGZwI>

Repositorio de Github: <https://github.com/RenzoRomeo/taller-de-programacion-1>

Nuestro código está desarrollado en la rama “grupo_a” y nuestro testeo está realizado en la rama “grupo_b” en el paquete src/main/java/test

Introducción

El proceso de testing se utiliza en la etapa de verificación para comprobar que el código funciona tal como se espera. Entre sus etapas se destacan:

- Caja Negra: utilizando únicamente el contrato del método, se suministra de entradas a un método y se verifica que las salidas sean las esperadas.
- Caja Blanca: dado el código, se deben generar casos de prueba para que se ejecuten todos los caminos independientes del módulo.
- Test de persistencia: consiste en verificar la integridad de los datos al persistir.
- Test de GUI: es una prueba automatizada que busca verificar el comportamiento de una interfaz. No es posible testear todos los caminos posibles sino que se utiliza un subgrupo de los mismos.
- Test de integración: se utiliza para verificar la interacción de los distintos módulos que ya fueron testeados. Se basan en los casos de uso del sistema.

Detalles de implementación

El programa desarrollado tiene como clase principal al Sistema. Esta es una clase que representa al comercio y que contiene todas las colecciones necesarias para el funcionamiento del mismo.

El sistema tiene un usuario Administrador que puede realizar operaciones de alta y baja para otros operarios, mozos, productos, mesas y promociones. También existe el tipo de usuario Operario, que puede asignar mozos a mesas, crear y cerrar comandas y agregar pedidos a las mismas. Un detalle de implementación es que la contraseña del operario se valida utilizando un regex (expresión regular) para verificar que contenga mayúsculas y números.

Una vez que el usuario inicia sesión, este establece el modo de operación del sistema. De esta forma, los métodos solo pueden ser utilizados por los usuarios autorizados.

El sistema es utilizado mediante una interfaz gráfica, utilizando el patrón de diseño de Modelo Vista Controlador.

Para realizar la documentación del programa y de los contratos de los diferentes módulos, se utilizó JavaDoc. La documentación generada luego fue intercambiada con el otro subgrupo para poder realizar las pruebas de caja negra.

A continuación se mencionan las diferentes pruebas realizadas sobre el programa desarrollado por el otro subgrupo para verificar que el mismo se ejecute correctamente.

Caja Negra

El testing de Caja Negra se utiliza cuando se conoce únicamente el contrato del módulo a testear. Se utiliza para verificar que las salidas obtenidas sean las esperadas.

Sistema

Esta clase representa el comercio a administrar. Contiene las colecciones de todas las otras entidades y las gestiona.

Los escenarios planteados son:

Número	Descripcion	
1	Estado operarios	Operario("Juan", "Perez", "Juan", "Juan123") y estado activo
		Operario("Jose", "Perez", "Jose", "Jose123") y estado inactivo
	Estado mozos	Mozo("Alberto", "Perez", new Date(1995, 5, 10), 0) y estado activo
	Estado producto	Producto(1, "Coca Cola", 80, 100, 50)
	Estado comandas	Comanda para la Mesa 1 con 1 unidad de Coca Cola
	Estado mesas	Mesa(1, 4) y Mesa(3, 3)
	Estado promociones	Promocion por producto para Coca Cola, aplica dos por uno.
		Promocion por producto para Coca Cola, aplica descuento por cantidad.

Número	Descripcion	
2	Estado operarios	Operario("Juan", "Perez", "Juan", "Juan123") y estado activo
		Operario("Jose", "Perez", "Jose", "Jose123") y estado inactivo
	Estado mozos	Mozo("Alberto", "Perez", new Date(1995, 5, 10), 0) y estado activo
	Estado producto	Producto(1, "Coca Cola", 80, 100, 50)
	Estado comandas	Comanda para la Mesa 1 con 1 unidad de Coca Cola
	Estado mesas	Mesa(1, 4) y Mesa(3, 3)
	Estado Promociones	vacio

IniciarSesionOperario

Tabla de particiones

Condición	Clases válidas	Clases inválidas
nombreUsuario		
estado operarios	1. Existe operario con nombreUsuario	2. No existe operario con nombreUsuario
estado operario	3. Operario activo	4. Operario inactivo
contrasenia	5. contrasenia correcta para el usuario	6. contrasenia incorrecta para el operario

Batería de pruebas

Tipos de clase	Escenario	Valores de entrada	Clases de prueba cubiertas	Salida esperada	Salida obtenida
Clase válida	1	nombreUsuario="Juan"; contrasenia="Juan123"	1, 3, 5	El operario inicia sesión	PASS
Clase inválida	1	nombreUsuario="Jorge"; contrasenia="Juan123"	2	UsuarioNoExisteException	PASS
	1	nombreUsuario="Jose"; contrasenia="Jose123"	1, 4, 5	UsuarioInactivoException	PASS
	1	nombreUsuario="Juan"; contrasenia="Pepe"	1, 3, 6	ContraseniaIncorrectaException	PASS

AgregarOperario

Tabla de particiones

Condición	Clases válidas	Clases inválidas
operario		
estado operarios	1. No existe un operario con el nombre de usuario especificado	2. Existe un operario con el nombre de usuario especificado

Batería de pruebas

Nota: El constructor de operario es Operario(nombre, apellido, nombreUsuario, contrasenia)

Tipos de clase	Escenario	Valores de entrada	Clases de prueba cubiertas	Salida esperada	Salida obtenida
Clase	1	Operario("Jorge",	1	Se agrega el	PASS

válida		"Perez", "Jorge", "Jorge123")		operario	
Clase inválida	1	Operario("Juan Martin", "Fernandez", "Juan", "Password123")	2	NombreDeUsuario NoDisponibleException	FAIL

En este caso, el test falló debido a que el sistema detectó que la contraseña no cumple los requisitos de seguridad, cuando en realidad sí los cumple.

EliminarOperario

Tabla de particiones

Condición	Clases válidas	Clases inválidas
operario		
estado operarios	1. Existe un operario con el nombre de usuario especificado	2. No existe un operario con el nombre de usuario especificado

Batería de pruebas

Nota: Operario Juan y Operario Esteban son referencias a los Operarios con dichos nombres.

Tipos de clase	Escenario	Valores de entrada	Clases de prueba cubiertas	Salida esperada	Salida obte nida
Clase válida	1	Operario Juan	1	Se elimina el operario	FAIL
Clase inválida	1	Operario Esteban	2	UsuarioNoExisteException	PAS S

En este caso, el test falló debido a que no se eliminó correctamente el operario del sistema.

AgregarMozo

Tabla de particiones

Condición	Clases válidas	Clases inválidas
mozo		
estado mozos	1. No existe un mozo con el nombre especificado	2. Existe un mozo con el nombre especificado

Batería de pruebas

Tipos de clase	Escenario	Valores de entrada	Clases de prueba cubiertas	Salida esperada	Salida obtenida
Clase válida	1	Mozo("Rodrigo", "Perez", Date(11/10/1990), 2)	1	Se agrega el mozo	PASS
Clase inválida	1	Mozo("Alberto", "Perez", Date(17/11/1991), 1)	2	MozoYaExistenteException	PASS

EliminarMozo

Tabla de particiones

Condición	Clases válidas	Clases inválidas
mozo		
estado mozos	1. Existe un mozo con el nombre especificado	2. No existe un mozo con el nombre especificado

Batería de pruebas

Nota: Mozo Alberto y Mozo Damian son referencias a los Mozo con dichos nombres.

Tipos de clase	Escenario	Valores de entrada	Clases de prueba cubiertas	Salida esperada	Salida obtenida
Clase válida	1	Mozo Alberto	1	Se elimina el mozo	PASS
Clase inválida	1	Mozo Damian	2	MozoNoExistenteException	PASS

EstablecerEstadoMozo

Tabla de particiones

Condición	Clases válidas	Clases inválidas
mozo		
estado mozos	1. Existe el mozo especificado	2. No existe el mozo especificado
estado	3. estado pertenece a {ACTIVO, DE_FRANCO, AUSENTE}	4. estado no pertenece a {ACTIVO, DE_FRANCO, AUSENTE}

Batería de pruebas

Tipos de clase	Escenario	Valores de entrada	Clases de prueba cubiertas	Salida esperada	Salida obtenida
Clase válida	1	Mozo Alberto, estado=ACTIVO	1, 3	Se elimina el mozo	PAS S
Clase inválida	1	Mozo Damian, estado=ACTIVO	2, 3	MozoNoExisteException	PAS S
Clase inválida	1	Mozo Alberto, estado=DESCANSANDO	1, 4	EstadoMozoInvalidoException	IMPOSIBLE

AgregarMesa

Tabla de particiones

Condición	Clases válidas	Clases inválidas
mesa		
estado mesas	1. No existe la mesa especificada	2. Existe la mesa especificada

Batería de pruebas

Nota: Mesa 1 y Mesa 2 son referencias a las mesas con dichas id.

Tipos de clase	Escenario	Valores de entrada	Clases de prueba cubiertas	Salida esperada	Salida obtenida
Clase válida	1	Mesa 1	1	Se agrega la mesa	PAS S
Clase inválida	1	Mesa 2	2	MesaYaExistenteException	PAS S

EliminarMesa

Nota: El sistema permite eliminar una mesa que tiene asociada una comanda. Esto no es posible testearlo ya que no lanza ninguna excepción ni invalida ninguna condición del contrato. Sin embargo sí invalida la SRS. Este error se debería detectar en la etapa de validación.

Tabla de particiones

Condición	Clases válidas	Clases inválidas
mesa		
estado Mesas	1. Existe la mesa	2. No existe la mesa

Batería de pruebas

Nota: Mesa 1 y Mesa 2 son referencias a las Mesas con dichos nombres.

Tipos de clase	Escenario	Valores de entrada	Clases de prueba cubiertas	Salida esperada	Salida obtenida
Clase válida	1	Mesa 1	1	Se elimina la mesa	PAS S
Clase inválida	1	Mesa 2	2	MezaNoExistenteException	PAS S

CrearComanda

Nota: El sistema permite pedir una cantidad negativa de producto. Esto no es posible testearlo ya que no lanza ninguna excepción ni invalida ninguna condición del contrato. Sin embargo sí invalida la SRS. Este error se debería detectar en la etapa de validación.

Tabla de particiones

Condición	Clases válidas	Clases inválidas
mesa	1. Mesa desocupada	2. Mesa ocupada
p	3. El producto tiene stock suficiente	4. El producto no tiene stock suficiente
cantidad		
Estado Mesas	5. Existe la mesa	6. No existe la mesa
Estado Productos	7. Existe el producto	8. No existe el producto
Estado Promociones Producto	9. Existen 2 productos promocionados el día actual	10. No existen 2 productos promocionados el día actual

Batería de pruebas

Tipos de clase	Escenario	Valores de entrada	Clases de prueba cubiertas	Salida esperada	Salida obtenida
----------------	-----------	--------------------	----------------------------	-----------------	-----------------

					nida
Clase válida	1	mesa3, cocaCola, 3	1, 3, 5, 7, 9	Se crea una comanda.	PAS S
	1	mesa3, cocaCola, 50	1, 4, 5, 7, 9	ProductoNoDisponibleException	PAS S
	1	mesa3, pepsi, 3	1, 5, 8, 9	ProductoNoExistenteException	PAS S
Clase inválida	1	mesa1, cocaCola, 3	2, 3, 5, 7, 9	MesaNoDisponibleException	PAS S
	1	mesa2, cocaCola, 3	3, 6, 7, 9	MesaNoExisteException	PAS S
	2	mesa3, cocaCola, 3	1, 3, 5, 7, 10	CantidadEnPromocionMenorException	PAS S

AgregarPedido

Tabla de particiones

Condición	Clases válidas	Clases inválidas
estado mesas	1. Existe la mesa especificada	2. No existe la mesa especificada
estado productos	3. Existe el producto especificado	4. No existe el producto especificado
estado comandas	5. Existe una comanda para la mesa especificada	6. No existe una comanda para la mesa especificada
producto	7. El producto tiene stock	8. El producto no tiene stock

Batería de pruebas

Tipos de clase	Escenario	Valores de entrada	Clases de prueba cubiertas	Salida esperada	Salida obtenida
Clase válida	1	producto=Producto Coca Cola, cantidad=2, mesa=Mesa 1	1, 3, 5, 7	Se agrega el pedido a la comanda	FAIL
	1	producto=Producto Cerveza, cantidad=2, mesa=Mesa 1	1, 4, 5	ProductoNoExisteException	PAS S
	1	producto=Producto Coca Cola, cantidad=2, mesa=Mesa 2	2, 3, 7	MesaNoExisteException	PAS S
	1	producto=Producto Coca Cola, cantidad=300, mesa=Mesa 1	1, 3, 5, 8	ProductoNoDisponibleException	PAS S
Clase inválida	1	producto=Producto Coca Cola, cantidad=2, mesa=Mesa 3	1, 3, 6	MesaNoTieneComandaException	PAS S

En este caso el test falló debido a que el pedido no fue agregado a la comanda.

Test de Cobertura

Element	Class, %	Method, % ▾
▼ test	93% (14/15)	88% (64/72)
CrearComandaTest	100% (1/1)	100% (7/7)
IniciarSesionTest	100% (1/1)	100% (6/6)
AgregarMesaTest	100% (1/1)	100% (4/4)
AgregarMozoTest	100% (1/1)	100% (4/4)
EliminarMesaTest	100% (1/1)	100% (4/4)
EliminarMozoTest	100% (1/1)	100% (4/4)
AgregarOperarioTest	100% (1/1)	100% (4/4)
EliminarOperarioTest	100% (1/1)	100% (4/4)
EstablecerEstadoMozoTest	100% (1/1)	100% (4/4)
CrearComandaEscenario2Test	100% (1/1)	100% (3/3)
AgregarPedidoTest	100% (1/1)	88% (8/9)
> escenarios	100% (2/2)	68% (11/16)
ConstructorOperarioTest	100% (1/1)	50% (1/2)
EscenariosSistema	0% (0/1)	0% (0/1)

Al realizar el test de cobertura para los casos de prueba planteados anteriormente, se observó que casi todas las pruebas mostraban una cobertura completa de las posibles ramas del método testeado. Los test que no presentan una cobertura del 100% son un constructor, que es trivial, y los asertos.

Casos de Uso

Procesar Comida

Descripción

Un cliente llega al comercio, ocupa una mesa, realiza pedidos a una comanda y luego cierra la comanda.

Actores

Operario del sistema a través de la GUI.

Pre-condiciones

- El operario debe estar logueado en el sistema.

Flujo normal

1. El operario crea una comanda para la mesa indicada.
2. El operario añade el pedido del cliente a la comanda de la mesa.

Mientras el cliente no quiera cerrar la comanda repite el paso 2.

3. El operario cierra la comanda de la mesa.

Excepciones

- E1. Mesa no existe. El sistema notifica el error y vuelve a la selección de mesa.
- E2. Mesa no disponible. El sistema notifica el error y vuelve a la selección de mesa.
- E3. Producto no existe. El sistema notifica el error y vuelve a la selección de producto.
- E4. Producto no disponible. El sistema notifica el error y vuelve a la selección de producto.

Post-condiciones

- Se genera la factura que representa la comanda del cliente.

Comentario

Entendemos que el trabajo práctico está incompleto dado que no fue posible terminar el código ni poder realizar todos los test pedidos. Pedimos disculpas por el desempeño que mejoraremos para la segunda parte.

Atte: grupo 3.