Métodos de añadir:

Método:
addOperario(Operario op):

Agrega el operario pasado por parámetro al ArrayList de operarios.
Pre: op != null. ArrayList de operarios inicializados.

Post: El operario pasado por parámetro se agrega al ArrayList de operarios.

Parameters:

operario - : Operario que se busca agregar a la cervecería.

Throws:

UsuarioRepetidoException - : Se lanza si ya hay un operario registrado con el mismo username.

ESCENARIOS:

NRO escenario	Descripción
1	-Operario cargado con username SANTISOSA
2	-Cervecería vacía de operarios.

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
Operario Op		
EstadoOperarios	Operario no existe (1)	Operario existe(2)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	Operario con username “operario” *Escenario 1 y 2*	Actualiza la lista de operarios	1
Incorrecta	Operario con username “SANTISOSA” *Escenario 1*	UsuarioRepetidoException	2

Método:
AddMesa(Mesa mesa)

Agrega la mesa pasada por parametro al ArrayList de mesas.

Pre: mesa != null.

Post: La mesa pasada por parametro se agrega al ArrayList de mesas.

Parameters:

mesa - : mesa que se desea agregar a la cerveceria.

Throws:

MesaRepetidaException - : Se lanza si ya hay una mesa registrada con el mismo nroMesa.

ESCENARIOS:

NRO escenario	Descripción
1	-Mesa cargada con nro mesa = 0.
2	-Cerveceria vacia de mesas

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
Mesa mesa		
EstadoMesas	Mesa no existe (1)	mesa existe(2)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	mesa con nromesa = 3 *Escenario 1 y 2*	Actualiza la lista de mesas	1
Incorrecta	Mesa con nromesa = 0 *Escenario 1*	MesaRepetidaException	**2**

Método:

void addMozo(Mozo mozo)

Agrega el mozo pasado por parámetro al ArrayList de mozos.

Pre: mozo != null.

Post: El mozo pasado por parámetro se agrega al ArrayList de mozos.

Parameters:

mozo - : mozo que se desea agregar a la cervecería.

Throws:

MozoRepetidoException - : Se lanza si ya hay un mozo registrado con el mismo nombre

ESCENARIOS:

NRO escenario	Descripción
1	-Mozo cargado con nombre Sant Lapi
2	-Cerveceria vacia de mozos

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
Mozo mozo		
EstadoMozos	Mozo no existe (1)	mozo existe(2)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	mozo con nombre = Peter Jackson *Escenario 1 y 2*	Actualiza la lista de mozos	1
Incorrecta	Mozo con nombre = Santi lapi *Escenario 1*	MozoRepetidoException	**2**

Método:

addProducto(Producto producto)

Agrega el producto pasado por parametro al ArrayList de productos.

Pre: producto != null.

Post: El producto pasado por parametro se agrega al ArrayList de productos.

Parameters:

producto - : producto que se desea agregar a la cerveceria.

Throws:

ProductoRepetidoException - : Se lanza si ya hay un producto registrado con el mismo nombre.

ESCENARIOS:

NRO escenario	Descripción
1	-Producto cargado con nombre “Hamburguesa”
2	-Cerveceria vacia de productos.

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
Producto producto		
EstadoProductos	producto no existe (1)	producto existe(2)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	producto con nombre = CheeseCake *Escenario 1 y 2*	Actualiza la lista de mozos	1
Incorrecta	producto con nombre = Hamburguesa *Escenario 1*	ProductoRepetidoException	**2**

Método:

public void addPromoProd(PromoProducto promo)

Agrega una promo producto a la lista de promociones de producto de la cerveceria

Pre: promo != null

Post: La promo pasada por parametro se agrega al ArrayList de promos de producto.

Parameters:

promo - : promocion de Producto que se desea agregar al ArrayList de promociones de producto de la cerveceria.

Throws:

PromoRepetidaException - : Se lanza si ya existe una promocion para ese producto con los mismos valores.

ProductoInexistenteException - : Se lanza si la quiere agregar una promoProducto de un producto que no existe en la cerveceria.

ESCENARIOS:

NRO escenario	Descripción
1	-Contiene una promo PromoHamburguesa = new PromoProducto(diasPromo, hamburguesa, false, true, 3, 180); diasPromo = lunes y viernes. -Contiene el producto milanesa y hamburguesa.
2	-Cerveceria vacía de promociones.

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
PromoProducto promo		
EstadoProductos	producto no existe (1)	producto existe(2)
EstadoPromociones	promocion existe (3)	promocion no existe(4)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	Nueva promocion con producto existente milanesa..	Actualiza la lista de PromoProductos	2,4
Incorrecta	PromoHamburguesa = new PromoProducto(diasPromo, hamburguesa, false, true, 3, 180);	PromoRepetidaException	**2,3**
Incorrecta	Nueva promocion con producto no listado, cocacola	ProductoInexistenteException	**1,4**

Método:
public void addPromoTemp(PromoTemporal promo)
Agrega una promo temporal a la lista de promociones temporales de la cervecería

Pre: promo != null

Post: La promocion pasada por parámetro se agrega al ArrayList de promos temporales .

Parameters:

promo - : promocion temporal que se desea agregar al ArrayList de promociones temporales de la cervecería.

Throws:

PromoRepetidaException - : Se lanza si ya existe una promoTemporal con el mismo nombre.

ESCENARIOS:

NRO escenario	Descripción
1	-Contiene una promo promoViernes
2	-Cerveceria vacía de promociones.

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
PromoTemp promo		
EstadoPromociones	promocion no existe (1)	promocion existe(2)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	Nueva promocion llamada “promoLunes”	Actualiza la lista de PromoProductos	1
Incorrecta	Promoción con nombre “promoViernes”	PromoRepetidaException	**2**

Métodos de Eliminar:

Método:
deleteOperario(Operario op): *//Sin documentación.*
Pre:

Post:

Throws:

OperarioInexistenteException

ESCENARIOS:

NRO escenario	Descripción
1	-Operario cargado con username SANTISOSA

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
Operario Op		
EstadoOperarios	Operario existe (1)	Operario no existe(2)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	Operario con username “Pedro salazar”	OperarioInexistenteException	1
Incorrecta	Operario con username “SANTISOSA”	Actualiza la lista de operarios, eliminando al operario.	**2**

Método:
deleteMesa(Mesa mesa): *//Sin documentación.*
Pre:

Post:

Throws:

ComandaAbiertaException

MesaInexistenteException

ESCENARIOS:

NRO escenario	Descripción
1	-Mesa cargada con nro mesa = 0. -Mesa con nromesa=3 con comanda cargada

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
Mesa mesa		
EstadoMesas	Mesa existe (1) Mesa libre (2)	mesa no existe(3) mesa no libre(4)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	Mesa con nro mesa =0	Elimina la mesa de la lista de mesas.	2,3
Incorrecta	Mesa con nro mesa =9	MesaInexistente Exception	1
Incorrecta	Mesa con nro mesa = 3	ComandaAbierta Exception	3,4

Método:

void deleteMozo(Mozo mozo) *//Sin documentación.*

Pre:

Post:

Throws:

MozoInexistenteException - : Se lanza si ya hay un mozo registrado con el mismo nombre

ESCENARIOS:

NRO escenario	Descripción
1	-Mozo cargado con nombre Sant Lapi

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
Mozo mozo		
EstadoMozos	Mozo existe (1)	mozo no existe(2)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	Mozo con nombre = Santi lapi	Actualiza la lista de mozos eliminando al indicado.	**2**
Incorrecta	Mozo con nombre = Peter Jackson	MozoInexistente Exception	**1**

Método:

public void deleteProducto(Producto prod) //*Sin documentación.*

Pre:

Post:

Throws:

ProductoEnComandaException

ProductoInexistenteException

ESCENARIOS:

NRO escenario	Descripción
1	-Producto hamburguesa cargado en sistema en COMANDA. -Producto pancho no cargado en sistema. -Producto Cerveza cargado en sistema.

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
Producto producto		
EstadoProductos	producto existe (1) producto en comanda abierta(2)	producto no existe(3) producto no esta en comanda abierta(4)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	producto con nombre = Cerveza	Actualiza la lista de productos eliminando al indicado.	1,4
Incorrecta	producto con nombre = pancho	ProductoInexistenteException	3
	producto con nombre = hamburguesa	ProductoEnComandaException	1,2

Método:

public void deletePromoProducto(PromoProducto promo) *//Sin documentación.*

Pre:

Post:

Throws:

PromoInexistenteException

ESCENARIOS:

NRO escenario	Descripción
1	-Contiene una promo PromoHamburguesa -No contiene promo llamada “promo”

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
PromoProducto promo		
EstadoPromociones	promocion existe (1)	promocion no existe(2)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	PromoHamburguesa	Actualiza la lista de promo productos eliminando la indicada.	1
Incorrecta	promo	PromoInexistenteException	2

Método:

public void deletePromoTemp(PromoTemporal promo) *//Sin documentación.*

Pre:

Post:

Throws:

PromoInexistenteException

ESCENARIOS:

NRO escenario	Descripción
1	-Contiene una promo promoViernes -No contiene promo llamada “promoLunes”

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
PromoProducto promo		
EstadoPromociones	promocion existe (1)	promocion no existe(2)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	promoViernes	Actualiza la lista de promo temporales eliminando la indicada.	1
Incorrecta	promoLunes	PromoInexistenteException	2

Metodo Estado:

Método:

public void setEstado(Mozo mozo, Estado e)

Modifica el estado del mozo pasado por parametro (por el estado e).

Pre: mozo != null. e != null y debe ser ACTIVO, FRANCO o AUSENTE.

Post: El mozo pasado por parametro tendra como nuevo estado el pasado por parametro.

Parameters:

mozo - : mozo al cual se quiere cambiar su estado.

e - : nuevo estado que se le asignara al mozo.

Throws:

MozoInexsitenteException - : Se lanza si el mozo pasado por parámetro no existe en el ArrayList de mozos de la cervecería.

ESCENARIOS:

NRO escenario	Descripción
1	-Mozo en sistema con nombre “Santi Lapi” -Mozo no cargado en sistema con nombre "Cristiano ronaldo

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
Mozo mozo		
Estado e		
EstadoColeccionMozos	Mozo existe (1)	Mozo no existe(2)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	Mozo con nombre “Santi Lapi”, estado = Franco	Setea el estado del mozo en Franco	1
Incorrecta	Mozo con nombre “Cristiano ronaldo” Estado = Activo	MozoInexsitente Exception	2

Metodo Asignación:

Método:

public void asignarMesa(Mozo mozo, Mesa mesa)

Asigna al mozo pasado por parametro la mesa pasada por parametro

Pre: mozo != null. mesa != null

Post: La mesa se agrega al ArrayList de mesas del mozo. El estado de la mesa pasa a "OCUPADA"

Parameters:

mozo - : El mozo al cual se quiere agregar una mesa.

mesa - : Numero de mesa a asignarle.

Throws:

MozoNoDisponibleException - : Se lanza si el mozo pasado por parámetro no está disponible.

MozoInexsitenteException - : Se lanza si el mozo pasado por parámetro no existe en el ArrayList de mozos de la cervecería.

MesaNoDisponibleException - : Se lanza si la mesa pasada por parámetro no está disponible.

MesaInexsitenteException - : Se lanza si la mesa pasada por parámetro no existe en el ArrayList de mesas de la cervecería.

ESCENARIOS:

NRO escenario	Descripción
1	-Mozo en sistema con nombre “Santi Lapi” -mesa cargada en sistema con nro=0. -Mozo en sistema con nombre “Wencho Avalos” y Estado Franco. -Mozo no cargado en sistema con nombre “Cristiano ronaldo”
2	-Mesa no cargada en el sistema con nro=4. -Mozo cargado en sistema con nombre “Guillermo” estado activo. -Mesa cargada en sistema que ya fue asignada con nro = 9

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
Mozo mozo	mozo activo (1)	mozo no activo(2)
Mesa mesa	Mesa no asignada(3)	Mesa asignada (4)
EstadoColeccionMozos	Mozo existe (5)	Mozo no existe(6)
EstadoColeccionMesas	Mesa existe (7)	Mesa no existe (8)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	Mesa con nro=0 Mozo con nombre = “Santi Lapi” *Escenario 1*	Se asigna la mesa al mozo.	1,3,5,7
Incorrecta	Mesa con nro=0 Mozo con nombre = “Wencho Avalos” *Escenario 1*	MozoNoDisponibleException	2,3,5,7
Incorrecta	Mesa con nro=0 Mozo con nombre = “Cristiano Ronaldo” *Escenario 1*	MozoInexsitenteException	6
Incorrecta	mesa con nro = 9 Mozo con nombre = “Guillermo”	MesaNoDisponibleException	1,4,5,7
Incorrecta	mesa con nro = 4 Mozo con nombre = “Guillermo”	MesaInexsitenteException	8

Métodos de Comandas (Tomar y Cerrar)

Método:

public void tomarComanda(Mesa mesa, ArrayList<Pedido> pedidos)

Toma una nueva comanda en caso de que la mesa esté libre. Si la mesa pasada por parámetro está ocupada, se agregan los pedidos al ArrayList de pedidos de la comanda abierta de esa mesa.

//Los pedidos tienen productos y cantidades de ellos.

Pre: mozo != null. mesa != null

Post: La mesa se agrega al ArrayList de mesas del mozo. El estado de la mesa pasa a "OCUPADA"

Parameters:

mozo - : El mozo al cual se quiere agregar una mesa.

mesa - : Numero de mesa a asignarle.

Throws:

MesaInexistenteException

MesaNoDisponibleException

ProductosInvalidosException

Escenario

NRO escenario	Descripción
1	-Cerveceria cargada con producto “Hamburguesa” y “Cerveza” -Producto no cargado en sistema “Pancho” -Mesa asignada con nro 0. -Mesa con nro 3 no cargada en sistema. -Mesa no asignada con nro 1

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
Mesa mesa	Mesa asignada(1)	Mesa no asignada (2)
Lista Pedidos		
EstadoColeccionMesa	Mesa existe (3)	Mesa no existe(4)
EstadoColeccionProductos	Producto existe (5)	Producto no existe (6)

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	Mesa con nro=0 Pedido con Hamburguesa y Cerveza.	Se toma la comanda, se agregan los pedidos a esta.	1,3,5
Incorrecta	Mesa con nro = 3 Pedido con Hamburguesa	MesaInexistente Exception	4
Incorrecta	Mesa con nro = 0 Pedido con pancho	ProductosInvalid osException	6
Incorrecta	Mesa con nro = 1 Pedido con Hamburguesa	MesaNoDisponib leException	2,3,5

Método:
public void cerrarMesa(Mesa mesa, String formaPago) *//Sin documentación.*

Nota de testeo: *Debido a que no contamos con mucha información del método, además de testear si la mesa posee comanda o no, en Eclipse (con Junit) se testeo que las promociones estén bien aplicadas al cerrar la mesa (que el precio de venta sea acorde a las promociones del sistema).*

Pre:

Post:

Throws:

MesaSinComandaException

Escenario

NRO escenario	Descripción
1	-Mesa con nro mesa = 0 con comanda. -Mesa con nro mesa = 1 con comanda. -Promoción con forma de pago Tarjeta.
2	-Mesa con nro mesa = 3 sin comanda tomada.

TABLA DE PARTICIONES:

Condiciones de entrada	Clases válidas	Clases inválidas
Mesa mesa	Mesa con comanda(1) Mesa apta promo(3)	Mesa sin comanda(2) Mesa no apta promo(4)
String formaPago		
EstadoColeccionMesa		

Batería de Pruebas

Tipo clase (correcta/incorrecta)	valores de entrada	Salida esperada	Clases de prueba cubiertas
Correcta	-Mesa con nro=0 -Forma pago “TARJETA” *Escenario 1.*	Se cierra la mesa, al tener una forma de pago apta para una promoción existente se aplica.	1,3
Correcta	-Mesa con nro = 1 -Forma pago “TARJETA” *Escenario 1.*	Se cierra la mesa, sin promocion.	1,4
Incorrecta	-Mesa con nro =3 -Forma de pago “EFECTIVO” *Escenario 2.*	MesaSinComanda Exception	2