Universidad de Granada

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

UNIVERSIDAD DE GRANADA

Desarrollo de Software

Practica 3
Pruebas Software

Profesor: Alberto Jesús Durán López
Alumnos:
Victor Casalini Gonzalez
Joaquin Salas Castillo
Antonio Fernandez Santiago
Kaito Lorenzo Okochi

5 de mayo de 2024

Índice

Los resultados se pueden encontrar en github a traves del siguiente enlace `https://github.com/vicmaviclu/DS/tree/master/P3/ejercicio_grupal`

1. Ejercicio Grupal
Realización de Pruebas Software sobre la aplicación desarrollada en la Practica 2

Planificación de Pruebas

Prueba de Valores por Defecto

Tiene como **objetivo** crear un objeto pizza con los valores por defecto para comprobar que son correctos, lo que asegura que se inicializa un objeto pizza de manera correcta y sin fallos.

```
test('Valores por defecto de pizza', () {
  checkPizza(pizza!, 'Pizza ', 0.0, [], '');
});
```

Prueba de Pizza Foto

El **objetivo** es verificar que se puede añadir una foto a la pizza y que la foto se guarda correctamente, esto es fundamental para que el usuario pueda visualizar correctamente el tipo de pizza.

```
test('Pizza con foto', () {
  final foto = 'assets/margarita.jpg';
  final pizzaConFoto = PizzaConFoto(pizza: pizza!, foto: foto);
  expect(pizzaConFoto.foto, foto);
});
```

Prueba del Builder

El **objetivo** es comprobar que el patrón *Builder* está implementado correctamente y puede construir las diferentes pizzas con los valores correctos, ya que esté patrón es una parte fundamental de la creación de los objetos Pizza en la aplicación.

```
Run | Debug
test('Builder de pizza', () {
  pizza = Director(MargaritaBuilder()).build(mediumSize);
  checkPizza(pizza!, margaritaName, margaritaPrice, margaritaIngredients,
      mediumSize);
});
```

Prueba de la Factoría

El **objetivo** es verificar que el patrón *Factory Method* toma los valores correctos y se los pasa al builder, ya que este patrón se encarga de crear las diferentes pizzas en nuestra aplicación.

```
Run | Debug
test('Factoria de pizza', () {
  pizza = PizzaFactory.createPizza(margaritaName, mediumSize);
  checkPizza(pizza!, margaritaName, margaritaPrice, margaritaIngredients,
      mediumSize);
});
```

Prueba del Decorator

El **objetivo** es comprobar que este patrón añada los ingredientes extra a la pizza actualizando su coste, ya que es imprescindible que funcione correctamente si el usuario decide añadir ingredientes extra a la hora de realizar el pedido.

```
Run | Debug
test('Decorador de pizza', () {
  pizza = PizzaFactory.createPizza(margaritaName, mediumSize);

  PizzaAdicionalQueso(pizza!).updateCoste();
  PizzaAdicionalQueso(pizza!).updateDescription();

  checkPizza(pizza!, margaritaName, margaritaPrice + 0.5,
      margaritaIngredients + ['queso extra'], mediumSize);
});
```

Prueba Pizza Extra

El **objetivo** es comprobar si la adición de un ingrediente extra a una pizza se realiza correctamente.

```
  test('Pizza extra', () {
    pizza = PizzaFactory.createPizza(nombre, tamano);

    PizzaExtras.anadirExtras(pizza!, ['Extra queso']);

    checkPizza(
        pizza!, nombre, precio + 0.5, ingredientes + ['queso extra'], tamano);
  });
});
```

Groups utilizados en prueba del pedido y coste total

En estos groups se almacenan los datos que se utilizarán es los dos tests siguientes que representan las pizzas que se pedirán así como los datos necesarios para realizar los pedidos.

```
group("Comprobacion APP", () {
  const tamano = 'Mediana';
  Pizza? pizza, pizza2;

  setUp(() {
    pizza = Director(MargaritaBuilder()).build(tamano);
    pizza2 = Director(MargaritaBuilder()).build(tamano);
    PizzaExtras.anadirExtras(pizza!, ['Extra queso']);
  });

  Run | Debug
  group('Realizar Pedido', () {
    Pedido? pedido;

    setUp(() {
      pedido = Pedido(
        pizzas: [pizza!, pizza2!],
        direccion: '123 Calle Falsa',
        tarjeta: '1234567812345678',
        numeroTelefono: '1234567890',
      );
      pedido!.hacerPedido();
    });
```

Prueba del Pedido

En esta prueba el **objetivo** es comprobar que se puede construir un pedido que contiene más de una pizza y al terminar se quede como "pedido realizado". Esta funcionalidad es crucial ya que se asegura de que el pedido se realiza correctamente.

```
Run | Debug
test('Pedido realizado', () {
  expect(pedido!.pedidoRealizado, isTrue);
});
```

Prueba del Coste Total

El **objetivo** es comprobar que el coste de las pizzas del pedido se calcula correctamente, ya que el coste es fundamental para la construcción del pedido.

```
test('Comprobar Coste Total', () {
  var costeTotalEsperado =
      pizza!.getCoste(pizza!.tamano) + pizza2!.getCoste(pizza2!.tamano);
  expect(pedido!.getCosteTotal(), costeTotalEsperado);
});
```

Prueba de eliminar las pizzas

El **objetivo** es comprobar que las pizzas de un pedido se eliminen correctamente.

```
  test('Pedido elimina las pizzas', () {
    pedido!.clear();
    expect(pedido!.pizzas, isEmpty);
  });
});
```

Group utilizado en pruebas relacionadas con la carta

En este group se almacenan los datos que se utilizarán es los tests siguientes que comprobarán la creación del objeto carta y su modificación al añadirle pizzas o quitarselas.

```
group('Carta', () {
  Carta? carta;
  const fotoPizza = 'assets/margarita.jpg';

  setUp(() {
    carta = Carta();
    carta!.reset();
    carta!.addPizza(pizza!, fotoPizza);
    carta!.addPizza(pizza2!, fotoPizza);
  });
```

Prueba de añadir pizzas a la carta

El **objetivo** es comprobar que las pizzas se añadan correctamente a la carta.

```
test('añade a la carta', () {
  expect(carta!.pizzas[0], isNotNull);
  expect(carta!.pizzas[1], isNotNull);
  expect(carta!.pizzas[0].pizza, equals(pizza!));
  expect(carta!.pizzas[1].pizza, equals(pizza2!));
  expect(carta!.pizzas[1].foto, equals(fotoPizza));
  expect(carta!.pizzas.length, 2);
});
```

Prueba de eliminar pizzas de la carta

El **objetivo** es comprobar que las pizzas se eliminen correctamente de la carta por su nombre.

```
test('removePizza elimina una pizza de la carta', () {
  carta!.removePizza(pizza!.getNombre);
  carta!.removePizza(pizza2!.getNombre);
  expect(carta!.pizzas, isNot(contains(pizza!)));
  expect(carta!.pizzas, isNot(contains(pizza2!)));
  expect(carta!.pizzas.length, 0);
});
```

Prueba de resetear la carta

El **objetivo** es comprobar que todas las pizzas se eliminen correctamente de la carta.

```
  test('reset borra todas las pizzas de la carta', () {
    carta!.reset();
    expect(carta!.pizzas, isEmpty);
  });
```

Tablas

[Pruebas software] ANÁLISIS		
Elemento a probar	**Condiciones**	**Datos requeridos**
Valores por defecto de pizza	Crear una nueva instancia de pizza	Ninguno
Pizza con foto	Crear una nueva instancia de PizzaConFoto	Una instancia de Pizza y la ruta de la imagen.
Builder de Pizza	Crear una pizza Margarita utilizando MargaritaBuilder	El tamaño de la pizza.
Factoría de Pizza	Crear una pizza utilizando PizzaFactory	El nombre de la pizza y el tamaño.
Decorador de Pizza	Añadir queso extra a una pizza utilizando PizzaAdicionalQueso	Una instancia de Pizza.
Pizza extra	Añadir queso extra a una pizza utilizando PizzaExtras	Una instancia de Pizza y una lista.
Pedido realizado	Realizar un pedido	Dos instancias de pizza, la dirección, el número de tarjeta y el número de teléfono.
Comprobar coste total	Comprobar el coste total de un pedido	Un pedido realizado.
Pedido elimina las pizzas	Eliminar pizzas de un pedido	Un pedido realizado.
Añade a la carta	Añadir pizzas a la carta	La instancia de carta, dos instancias de Pizza y la ruta de la imagen.
RemovePizza	Eliminar pizzas de la carta por el nombre	Una instancia de carta con pizzas añadidas y el nombre de la pizza a eliminar.
Reset carta	Elimina todas las pizzas de la carta	La instancia de carta.

[Pruebas de Diseño] DISEÑO			
Casos de Prueba	**Entornos de Prueba**	**Datos de Prueba**	**Relación con las condiciones de prueba (trazabilidad)**
Comprobación de pizzas	Flutter Test	Pizza, nombre, foto, tamaño, extras	Se verifica que la construcción de las pizzas es correcta y que los diferentes patrones que intervienen en su creación interactúan correctamente
Comprobación APP	Flutter Test	Pedido con pizzas, dirección, tarjeta, número de teléfono	Se verifica que los pedidos se realizan de manera correcta, calculando su coste y pudiendo eliminar las pizzas del pedido
Comprobación Carta	Flutter Test	Carta, pizza, nombre de pizza, foto	Se verifica que el objeto carta se construye correctamente, pudiendo añadir pizzas y quitarlas

Pruebas de Componentes (Widgets)

Estas pruebas se realizan para verificar el comportamiento y apariencia de los widgets individuales, además de su interacción con la interfaz de usuario y la lógica del modelo desarrollado.

Por defecto, las pruebas se realizan de manera síncrona, pero los widgets pueden interactuar con datos o servicios externos de forma asíncrona, lo que significa que pueden realizar operaciones como la obtención de datos de una API o la respuesta a eventos del usuario sin bloquear la interfaz de usuario. Por lo que, las pruebas de widgets deben ser capaces de simular y verificar este tipo de comportamiento asíncrono de manera efectiva.