Universidad de Oviedo
Escuela Politécnica de Ingeniería de Gijón
Departamento de Informática
GIJÓN

Practica 4: Pruebas de Componentes

Nombre corto: Practica 4
Código del equipo: IS2024G33

Pruebas de Componentes

Doc. Id.: PRC
Versión: 1.0

Fecha: 07-01-2024
Entregado por: Gustavo Sobrado Aller

Escrito por: Gustavo Sobrado Aller

Carácter: Definitivo

Equipo de trabajo

DIR	José Ramón de Diego
JEDS	Gustavo Sobrado Aller

Tabla de Contenido

Lista de Tablas

Historia

Versión	Fecha	Cambios introducidos
1.0	05-12-2024	

1. Introducción.

Esta práctica consiste en diseñar pruebas de componentes de forma individual para casos prácticos, como la identificación del tipo de un triángulo y la adición de un producto a un carrito de compras en una tienda. Se deben aplicar varios criterios de cobertura, en especial con la cobertura de sentencias. En la segunda parte, se requiere implementar estas pruebas utilizando JUnit. La documentación del diseño de las pruebas y las clases JUnit están denominadas como TriangleTest.java y CarritoTest.java, en ellas se detalla el código de los métodos que mencionaré más adelante.

2. Proyecto TriangleTest

2.1. Casos de Prueba

I. Caso de Prueba 1: testEquilatero

Objetivo: Probar que el triángulo es equilátero.

Condiciones de la prueba: Los valores de cada lado del triángulo son iguales y cumplen las condiciones para que el triángulo sea válido.

Entradas: 3, 3, 3.

Salida Esperada: "Equilatero".

II. Caso de Prueba 2: testNotEquilatero

Objetivo: Probar que el triángulo no es equilátero.

Condiciones de la prueba: Dos lados son iguales, pero el tercero es diferente, lo que convierte al triángulo en isósceles.

Entradas: 2, 3, 3.

Salida Esperada: "Isosceles".

III. Caso de Prueba 3: testIsosceles

Objetivo: Probar que el triángulo es isósceles.

Condiciones de la prueba: Dos lados son iguales y el tercero es diferente, lo que convierte al triángulo en isósceles.

Entradas: 5, 5, 3.

Salida Esperada: "Isosceles".

IV. Caso de Prueba 4: testEscaleno

Objetivo: Probar que el triángulo es escaleno.

Condiciones de la prueba: Ningún lado del triángulo es igual al otro, lo que convierte al triángulo en escaleno.

Entradas: 4, 5, 6.

Salida Esperada: "Escaleno".

V. Caso de Prueba 5: testRectangulo

Objetivo: Probar que el triángulo es rectángulo.

Condiciones de la prueba: El triángulo cumple el teorema de Pitágoras.

Entradas: 3, 4, 5.

Salida Esperada: "Rectangulo".

VI. Caso de Prueba 6: testTrianguloLadosNegativos

Objetivo: Probar que un triángulo con un lado negativo es inválido.

Condiciones de la prueba: Uno de los lados es negativo, lo que hace que el triángulo sea inválido.

Entradas: -3, 4, 5.

Salida Esperada: "Invalido".

VII. Caso de Prueba 7: testTrianguloLadosCero

Objetivo: Probar que un triángulo con un lado igual a cero es inválido.

Condiciones de la prueba: Uno de los lados es cero, lo que hace que el triángulo sea inválido.

Entradas: 0, 4, 5.

Salida Esperada: "Invalido".

VIII. Caso de Prueba 8: testTrianguloLadoDemasiadoLargo

Objetivo: Probar que un triángulo con la suma de dos lados menor o igual que el tercer lado es inválido.

Condiciones de la prueba: La suma de dos lados es igual o menor que el tercer lado, lo que hace que el triángulo sea inválido.

Entradas: 1, 2, 3.

Salida Esperada: "Invalido".

IX. Caso de Prueba 9: testTrianguloLadosSumaIgual

Objetivo: Probar que un triángulo con la suma de dos lados igual al tercer lado es inválido.

Condiciones de la prueba: La suma de dos lados es igual al tercer lado, lo que hace que el triángulo sea inválido.

Entradas: 5, 5, 10.

Salida Esperada: "Invalido".

X. Caso de Prueba 10: testTrianguloIsoscelesRectangulo

Objetivo: Probar que el triángulo es rectángulo y no isósceles.

Condiciones de la prueba: Los lados cumplen el teorema de Pitágoras y dos lados son iguales.

Entradas: 5, 5, √50.

Salida Esperada: "Rectangulo" (El test podría fallar si el orden de las condiciones en getTriangleType() no es el correcto).

XI. Caso de Prueba 11: testTrianguloRectanguloLadosGrandes

Objetivo: Probar que un triángulo con lados grandes sigue siendo clasificado correctamente como rectángulo.

Condiciones de la prueba: Los lados son extremadamente grandes, pero cumplen el teorema de Pitágoras.

Entradas: 1e10, 1e10, 1.41421356e10.

Salida Esperada: "Rectangulo".

XII. Caso de Prueba 12: testTrianguloRectanguloConDecimales

Objetivo: Probar que un triángulo con lados decimales cumple el teorema de Pitágoras.

Condiciones de la prueba: Los lados cumplen el teorema de Pitágoras con valores decimales.

Entradas: 0.3, 0.4, 0.5.

Salida Esperada: "Rectangulo".

XIII. Caso de Prueba 13: testTrianguloIsoscelesPrecision

Objetivo: Probar que dos lados casi iguales son considerados como un triángulo isósceles.

Condiciones de la prueba: Dos lados son casi iguales pero no exactamente.

Entradas: 5.0, 5.0, 5.0000001.

Salida Esperada: "Isosceles" (Este test podría fallar si no se tiene en cuenta la precisión).

XIV. Caso de Prueba 14: testTrianguloOverflow

Objetivo: Probar que el triángulo con valores extremadamente grandes no provoca un desbordamiento (overflow).

Condiciones de la prueba: Los lados son extremadamente grandes, pero deben ser clasificados correctamente.

Entradas: Double.MAX_VALUE / 2, Double.MAX_VALUE / 2, Double.MAX_VALUE / 2 - 1.

Salida Esperada: "Equilatero" (Este test podría fallar debido a overflow).

XV. Caso de Prueba 15: testTrianguloEscalenoCondicionesExtremas

Objetivo: Probar que los lados que apenas cumplen las condiciones de validez siguen siendo clasificados correctamente.

Condiciones de la prueba: Lados que apenas cumplen las condiciones de validez.

Entradas: 1.0, 1.0, 1.9999999.

Salida Esperada: "Isosceles" (Este test podría fallar debido a la precisión).

2.2. Cobertura de Sentencias

La siguiente tabla muestra la cobertura de sentencias en la clase Triangle, indicando qué métodos y sentencias del código son cubiertas por cada prueba. Se asegura que todas las ramas y condiciones sean evaluadas durante las pruebas.

Método	Sentencia Cubierta	Prueba que la Cubre
isTriangle()	Comprobación de lados negativos o cero (retorna false)	testTrianguloLadosNegativos, testTrianguloLadosCero
isTriangle()	Comprobación de la desigualdad triangular (retorna false si la suma de dos lados no supera el tercer lado)	testTrianguloLadoDemasiadoLargo, testTrianguloLadosSumaIgual
isTriangle()	Retorno true cuando el triángulo es válido	Todos los casos válidos de triángulos
isRightTriangle()	Comprobación de si el triángulo cumple el teorema de Pitágoras (rectángulo)	testRectangulo, testTrianguloIsoscelesRectangulo
getTriangleType()	Verifica si el triángulo es inválido	testTrianguloLadosNegativos, testTrianguloLadosCero, testTrianguloLadoDemasiadoLargo
getTriangleType()	Comprobación de triángulo equilátero (todos los lados iguales)	testEquilatero
getTriangleType()	Comprobación de triángulo isósceles (dos lados iguales)	testNotEquilatero, testIsosceles
getTriangleType()	Comprobación de triángulo rectángulo	testRectangulo, testTrianguloIsoscelesRectangulo
getTriangleType()	Caso escaleno (triángulo sin lados iguales y no rectángulo)	testEscaleno

Tabla 1: Resumen Clase Triangle 1

A modo de simplificar la tabla de sentencias en la que se marcará con una X las celdas en las que se cumple dicha sentencia, pongo identificadores a cada caso. Es importante tener en cuenta que no se muestran todos los casos de uso, ya que con estos cinco cubrimos todas las sentencias posibles. La tabla resultante seria la siguiente:

Caso de Prueba	Inválido	Equilátero	Isósceles	Escaleno	Rectángulo
1		X			
3			X		
4				X	
5					X
6	X				

Tabla 2: Cobertura de Sentencias

2.3. Cobertura de Decisiones

La siguiente tabla muestra la cobertura de las decisiones lógicas dentro de la clase Triangle. Cada decisión es evaluada durante las pruebas, y se indica qué pruebas cubren cada una de las decisiones.

Método	Sentencia Cubierta	Prueba que la Cubre
isTriangle()	Comprobación de lados negativos o cero (retorna false)	testTrianguloLadosNegativos, testTrianguloLadosCero
isTriangle()	Comprobación de la desigualdad triangular (retorna false si la suma de dos lados no supera el tercer lado)	testTrianguloLadoDemasiadoLargo, testTrianguloLadosSumaIgual
isTriangle()	Retorno true cuando el triángulo es válido	Todos los casos válidos de triángulos
isRightTriangle()	Comprobación de si el triángulo cumple el teorema de Pitágoras (rectángulo)	testRectangulo, testTrianguloIsoscelesRectangulo
getTriangleType()	Verifica si el triángulo es inválido	testTrianguloLadosNegativos, testTrianguloLadosCero, testTrianguloLadoDemasiadoLargo
getTriangleType()	Comprobación de triángulo equilátero (todos los lados iguales)	testEquilatero
getTriangleType()	Comprobación de triángulo isósceles (dos lados iguales)	testNotEquilatero, testIsosceles
getTriangleType()	Comprobación de triángulo rectángulo	testRectangulo, testTrianguloIsoscelesRectangulo
getTriangleType()	Caso escaleno (triángulo sin lados iguales y no rectángulo)	testEscaleno

Tabla 3: Resumen Clase Triangle 2

Los casos de uso de cubren cada decisión vienen dados por la siguiente tabla (marcaremos con una "F" las decisiones que se evalúen a falso y con una "C" las que se evalúen a cierto), al igual que en las sentencias hay más casos que las evalúan, pero con los seis especificados en la siguiente tabla ya cubrimos todas las decisiones.

Caso de Prueba	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
1	F	F	C	F	C	F	C	F	F	F
3	F	F	C	F	C	F	F	C	F	F
4	F	F	C	F	C	F	F	F	F	C
5	F	F	C	C	F	F	F	F	C	F
6	C	F	F	F	C	C	F	F	F	F
8	F	C	F	F	F	F	F	F	F	F

Tabla 4: Cobertura de Decisiones

2.4. Cobertura de Condiciones

Uso la misma tabla a modo de resumen para cada tipo de cobertura aunque no cambie, debido a que solo tenemos una clase en donde se desarrollan todos los métodos.

Método	Condición Cubierta	Prueba que la Cubre
isTriangle()	Lados negativos o cero (retorna false)	testTrianguloLadosNegativos, testTrianguloLadosCero
isTriangle()	Desigualdad triangular (retorna false si la suma de dos lados no supera el tercer lado)	testTrianguloLadoDemasiadoLargo, testTrianguloLadosSumaIgual
isTriangle()	Retorno true cuando el triángulo es válido	Todos los casos válidos de triángulos: testEquilatero, testNotEquilatero, testIsosceles, testEscaleno, testRectangulo
isRightTriangle()	Cumple el teorema de Pitágoras (triángulo rectángulo)	testRectangulo, testTrianguloIsoscelesRectangulo
getTriangleType()	Verifica si el triángulo es inválido (no cumple las condiciones de lados válidos)	testTrianguloLadosNegativos, testTrianguloLadosCero, testTrianguloLadoDemasiadoLargo
getTriangleType()	Triángulo equilátero (todos los lados iguales)	testEquilatero
getTriangleType()	Triángulo isósceles (dos lados iguales)	testNotEquilatero, testIsosceles
getTriangleType()	Triángulo rectángulo (cumple el teorema de Pitágoras)	testRectangulo, testTrianguloIsoscelesRectangulo
getTriangleType()	Triángulo escaleno (sin lados iguales y no rectángulo)	testEscaleno

Tabla 5: Resumen Clase Triangle 3

Haremos el mismo procedimiento que en la cobertura de decisiones:

Caso de Prueba	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
1	F	F	C	F	C	F	C	F	F	F
3	F	F	C	F	C	F	F	C	F	F
4	F	F	C	F	C	F	F	F	F	C
5	F	F	C	C	F	F	F	F	C	F
6	C	F	F	F	C	C	F	F	F	F
8	F	C	F	F	F	F	F	F	F	F

Tabla 6: Cobertura de Condiciones

2.5. Cobertura de Múltiple Condición

Usamos la misma tabla:

Método	Condición Cubierta	Prueba que la Cubre
isTriangle()	Lados negativos o cero (retorna false)	testTrianguloLadosNegativos, testTrianguloLadosCero
isTriangle()	Desigualdad triangular (retorna false si la suma de dos lados no supera el tercer lado)	testTrianguloLadoDemasiadoLargo, testTrianguloLadosSumaIgual
isTriangle()	Retorno true cuando el triángulo es válido	Todos los casos válidos de triángulos: testEquilatero, testNotEquilatero, testIsosceles, testEscaleno, testRectangulo
isRightTriangle()	Cumple el teorema de Pitágoras (triángulo rectángulo)	testRectangulo, testTrianguloIsoscelesRectangulo
getTriangleType()	Verifica si el triángulo es inválido (no cumple las condiciones de lados válidos)	testTrianguloLadosNegativos, testTrianguloLadosCero, testTrianguloLadoDemasiadoLargo
getTriangleType()	Triángulo equilátero (todos los lados iguales)	testEquilatero
getTriangleType()	Triángulo isósceles (dos lados iguales)	testNotEquilatero, testIsosceles
getTriangleType()	Triángulo rectángulo (cumple el teorema de Pitágoras)	testRectangulo, testTrianguloIsoscelesRectangulo
getTriangleType()	Triángulo escaleno (sin lados iguales y no rectángulo)	testEscaleno

Tabla 7: Resumen Clase Triangle 4

En este caso como vamos a hacer combinaciones de varios métodos les pongo un identificador a modo de simplificar la tabla. La lista de identificadores junto con sus nombres es esta:

- **MC1**: Lados Negativos o Cero + Desigualdad Triangular
- **MC2**: Lados Negativos o Cero + Triángulo Válido
- **MC3**: Lados Negativos o Cero + Teorema de Pitágoras (Rectángulo)
- **MC4**: Lados Negativos o Cero + Triángulo Inválido
- **MC5**: Lados Negativos o Cero + Equilátero
- **MC6**: Lados Negativos o Cero + Isósceles
- **MC7**: Lados Negativos o Cero + Rectángulo
- **MC8**: Lados Negativos o Cero + Escaleno
- **MC9**: Desigualdad Triangular + Triángulo Válido
- **MC10**: Desigualdad Triangular + Teorema de Pitágoras (Rectángulo)
- **MC11**: Desigualdad Triangular + Triángulo Inválido
- **MC12**: Desigualdad Triangular + Escaleno

Caso de Prueba	MC1	MC2	MC3	MC4	MC5	MC6	MC7	MC8	MC9	MC10	MC11	MC12
1	F	C	F	F	C	F	F	F	F	F	F	F
2	F	C	F	F	C	C	F	F	F	F	F	F
4	F	C	F	F	C	F	F	C	C	F	F	C
5	F	C	C	C	F	F	F	C	F	C	C	F
6	C	F	F	F	C	F	F	F	F	F	F	F
8	F	F	F	F	C	F	F	F	F	F	F	F
10	F	C	C	C	F	F	C	C	F	C	C	F
13	F	C	C	F	C	F	C	F	F	F	F	C
15	F	C	F	F	C	F	C	C	C	F	F	C

Tabla 8: Cobertura de Múltiple Condición

3. Proyecto CarritoTest

3.1 Casos de Prueba

I. Caso de Prueba 1: testCarritoOferta

Objetivo: Probar que el producto en oferta se añade correctamente al carrito y se actualiza el total.
Condiciones de la prueba: El producto está en el catálogo, está en oferta con un valor de 95, y el carrito ya tiene productos por valor de 100.
Entradas:

- Producto: "0001" con precio de 100.00, 1 unidad.
- Carrito con total acumulado de 100.00.

Salida Esperada: El total del carrito debería ser 195.00.

II. Caso de Prueba 2: testProductoNoEnCatalogo

Objetivo: Probar que un producto que no está en el catálogo no puede ser añadido al carrito.
Condiciones de la prueba: El producto no está en el catálogo.
Entradas:

- Producto: "0002" con precio de 20.00, 2 unidades.
- Carrito con total acumulado de 50.00.

Salida Esperada: El código de error debe ser 101 y el total del carrito sigue siendo 50.00.

III. Caso de Prueba 3: testStockInsuficiente

Objetivo: Probar que si el stock es insuficiente, el producto no puede ser añadido al carrito.
Condiciones de la prueba: El producto está en el catálogo pero no hay suficiente stock disponible.
Entradas:

- Producto: "0003" con precio de 30.00, se intentan añadir 6 unidades.
- Carrito con total acumulado de 0.00.

Salida Esperada: El código de error debe ser 102 y el total del carrito sigue siendo 0.00.

IV. Caso de Prueba 4: testProductoSinOferta

Objetivo: Probar que un producto que no está en oferta se añade al carrito con su precio original.
Condiciones de la prueba: El producto está en el catálogo pero no está en oferta.
Entradas:

- Producto: "0004" con precio de 50.00, 2 unidades.
- Carrito con total acumulado de 0.00.

Salida Esperada: El total del carrito debe ser 100.00.

V. Caso de Prueba 5: testProductoEnOferta

Objetivo: Probar que un producto en oferta se añade correctamente al carrito y se aplica el precio de oferta.
Condiciones de la prueba: El producto está en el catálogo, está en oferta con un valor de 80.00, y el carrito ya tiene productos por valor de 100.
Entradas:

- Producto: "0003" con precio original de 100.00, 2 unidades.
- Carrito con total acumulado de 100.00.

Salida Esperada: El total del carrito debería ser 260.00.

VI. Caso de Prueba 6: testCarritoConOfertaYStockExacto

Objetivo: Probar que un producto en oferta y con stock exacto se añade correctamente al carrito.
Condiciones de la prueba: El producto está en el catálogo, está en oferta con un valor de 80.00, y el stock es exacto a la cantidad requerida.
Entradas:

- Producto: "0005" con precio de 100.00, 3 unidades (stock exacto).
- Carrito con total acumulado de 0.00.

Salida Esperada: El total del carrito debe ser 240.00.

VII. Caso de Prueba 7: testActualizacionStock

Objetivo: Probar que el stock del producto se actualiza correctamente cuando se añade al carrito.
Condiciones de la prueba: El producto está en el catálogo, no está en oferta, el stock inicial es 5, y se solicitan 3 unidades.
Entradas:

- Producto: "0005" con precio de 20.00, 3 unidades solicitadas.
- Carrito con total acumulado de 0.00.

Salida Esperada: El stock del producto debe disminuir a 2.

VIII. Caso de Prueba 8: testPrecioOferta

Objetivo: Probar que cuando un producto está en oferta, se aplica el precio de oferta.
Condiciones de la prueba: El producto está en oferta con un precio de 15.00.
Entradas:

- Producto: "0005" con precio original de 20.00.

Salida Esperada: El precio de oferta es 15.00.

IX. Caso de Prueba 9: testPrecioOriginalCuandoNoHayOferta

Objetivo: Probar que cuando un producto no está en oferta, se devuelve el precio original.
Condiciones de la prueba: El producto no está en oferta.
Entradas:

- Producto: "0005" con precio original de 20.00.

Salida Esperada: El precio devuelto debe ser 20.00.

X. Caso de Prueba 10: testStockInsuficienteNoDecrementar

Objetivo: Probar que si el stock es insuficiente, no se decrementa el stock y no se añade el producto al carrito.
Condiciones de la prueba: El producto está en el catálogo, pero el stock disponible es insuficiente.
Entradas:

- Producto: "0005" con precio de 20.00, 3 unidades disponibles.
- Carrito con total acumulado de 0.00.

Salida Esperada: El código de error debe ser 102 y el stock no debe haber cambiado.

XI. Caso de Prueba 11: testCarritoVacioOBorde

Objetivo: Probar que un carrito vacío o con un total cercano a cero se comporta correctamente.
Condiciones de la prueba: Intentar añadir 0 productos al carrito.
Entradas:

- Producto: "0015" con precio de 1.00, 0 unidades.
- Carrito con total acumulado de 0.00.

Salida Esperada: El total del carrito sigue siendo 0.00.

XII. Caso de Prueba 12: testOperacionesConsecutivas

Objetivo: Probar que se pueden añadir múltiples productos al carrito de manera consecutiva y el total se calcula correctamente.

Condiciones de la prueba: Añadir múltiples productos consecutivamente.

Entradas:

- Producto 1: "0016" con precio de 50.00, 2 unidades.
- Producto 2: "0017" con precio de 30.00, 3 unidades.
- Carrito con total acumulado de 0.00.

Salida Esperada: El total del carrito debe ser 190.00.

XIII. Caso de Prueba 13: testConcurrencia

Objetivo: Probar que el carrito maneja correctamente el acceso concurrente y actualiza el total y el stock.

Condiciones de la prueba: Simular acceso concurrente al carrito.

Entradas:

- Producto: "0018" con precio de 20.00, 5 unidades y 3 unidades solicitadas en el otro hilo.
- Carrito con total acumulado de 0.00.

Salida Esperada: El total del carrito debe ser 160.00 y el stock debe ser 2.

3.2 Cobertura de Sentencias

La tabla identifica claramente los métodos, las condiciones que cubren y los casos de prueba que las ejercitan, lo cual facilita la trazabilidad de las pruebas en el proyecto.

Clase	Método	Condición Cubierta	Prueba que Cubre
Carrito	nuevoItem()	Actualización del total del carrito al agregar un nuevo item	testCarritoOferta, testProductoSinOferta, testProductoEnOferta, testCarritoConOfertaYStockExacto, testActualizacionStock
ItemCarrito	ItemCarrito()	Asignación de código, unidades y precio al item	Todos los casos que crean un ItemCarrito.
Producto	Producto()	Asignación de código, stock y precio al producto	Todos los casos que crean un Producto.
Catalogo	existeProducto()	Verifica si el producto existe en el catálogo	testProductoNoEnCatalogo
Catalogo	decrementarStock()	Disminuye el stock del producto en el catálogo (aunque no implementado)	testActualizacionStock, testStockInsuficienteNoDecrementar
ControlCarritos	introducirProducto()	Verifica si el producto está en el catálogo	testProductoNoEnCatalogo, testProductoEnOferta, testCarritoOferta, testProductoSinOferta, testStockInsuficiente
ControlCarritos	introducirProducto()	Verifica si hay suficiente stock del producto	testStockInsuficiente, testStockInsuficienteNoDecrementar
ControlCarritos	introducirProducto()	Aplica el precio de oferta si el producto está en oferta	testProductoEnOferta, testPrecioOferta
ControlCarritos	introducirProducto()	Calcula el precio normal del producto si no está en oferta	testProductoSinOferta
ControlCarritos	introducirProducto()	Verifica que el stock se actualice correctamente al añadir productos	testActualizacionStock
ControlOfertas	enOferta()	Verifica si el producto está en oferta	testProductoEnOferta, testPrecioOferta
ControlOfertas	precioOferta()	Retorna el precio de oferta del producto	testPrecioOferta

Tabla 9: Resumen Proyecto Carritos

Primero vamos a identificar las sentencias clave del código y como las cubren los casos de prueba. Aquí presento las sentencias clave:

1) S1: total = total + it.getUnidades() * it.getPrecio(); (Carrito: nuevoItem)
2) S2: codigo = cod; (ItemCarrito: Constructor)
3) S3: unidades = uni; (ItemCarrito: Constructor)
4) S4: precio = pre; (ItemCarrito: Constructor)
5) S5: codigo = codPro; (Producto: Constructor)
6) S6: stock = stockPro; (Producto: Constructor)
7) S7: precio = precioPro; (Producto: Constructor)
8) S8: return existePro; (Catalogo: existeProducto)
9) S9: // No acción (vacío) (Catalogo: decrementarStock)
10) S10: double precio; (ControlCarritos: introducirProducto)
11) S11: if (!catalogo.existeProducto(pro)) (ControlCarritos: existeProducto)
12) S12: if (unidades > pro.getStock()) (ControlCarritos: verificar stock)
13) S13: if (controlOfertas.enOferta(pro)) (ControlCarritos: comprobar oferta)
14) S14: precio = controlOfertas.precioOferta(pro); (ControlCarritos: aplicar precio de oferta)
15) S15: precio = pro.getPrecio(); (ControlCarritos: aplicar precio normal)
16) S16: ItemCarrito it = new ItemCarrito(pro.getCodigo(), unidades, precio); (ControlCarritos: crear ItemCarrito)
17) S17: car.nuevoItem(it); (ControlCarritos: agregar item al carrito)
18) S18: catalogo.decrementarStock(pro, unidades); (ControlCarritos: decrementar stock)
19) S19: return 0; (ControlCarritos: retorno)

Ahora vamos a presentar los casos de prueba:

1) Caso de prueba 1: testCarritoOferta
2) Caso de prueba 2: testProductoNoEnCatalogo
3) Caso de prueba 3: testStockInsuficiente
4) Caso de prueba 4: testProductoSinOferta
5) Caso de prueba 5: testProductoEnOferta
6) Caso de prueba 6: testCarritoConOfertaYStockExacto
7) Caso de prueba 7: testActualizacionStock
8) Caso de prueba 8: testPrecioOferta
9) Caso de prueba 9: testPrecioOriginalCuandoNoHayOferta
10) Caso de prueba 10: testStockInsuficienteNoDecrementar
11) Caso de prueba 11: testCarritoVacioOBorde
12) Caso de prueba 12: testOperacionesConsecutivas
13) Caso de prueba 13: testConcurrencia

Generamos una tabla con cada caso de prueba y las sentencias. Marcamos con una X las celdas en las que se cumple dicha sentencia:

CP	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19
1	x				x		x			x	x	x	x	x	x	x	x	x	
2	x	x			x	x				x	x			x	x	x		x	
3	x	x			x	x				x	x			x	x	x		x	
4	x				x			x	x		x	x	x	x	x	x	x	x	
5	x		x	x			x		x	x	x	x	x	x	x			x	
6				x		x		x					x		x	x	x	x	x
7				x		x		x					x		x	x	x	x	x
8	x				x							x		x		x	x		x
9	x				x							x		x		x	x		x
10	x		x	x			x	x		x	x	x	x		x	x	x	x	
11	x		x	x			x	x		x	x	x	x		x	x		x	
12	x		x	x			x	x		x	x	x	x		x	x	x	x	
13	x		x		x		x				x	x	x	x	x	x	x	x	
14				x							x	x	x	x	x	x	x	x	
15	x		x		x		x	x	x		x	x	x	x	x	x	x	x	

Tabla 10: Cobertura de Sentencias

3.3 Cobertura de Decisiones

Para realizar la cobertura de decisiones, necesitamos analizar cada decisión (que generalmente está vinculada a una condición que toma el valor verdadero o falso, afectando el flujo del programa) y comprobar si cada caso de prueba evalúa esa decisión de manera correcta. Es decir, debemos identificar en qué casos se evalúan las decisiones a cierto (C) o a falso (F), dependiendo de los valores de las entradas.

En el caso de ControlCarritos, algunas decisiones clave incluyen:

1) Decisión en introducirProducto (si el producto existe en el catálogo):

   if (!catalogo.existeProducto(pro))

2) Decisión para verificar si hay suficiente stock:

   if (unidades > pro.getStock())

3) Decisión para comprobar si el producto está en oferta:

   if (controlOfertas.enOferta(pro))

4) Decisión para decidir si se debe usar el precio de oferta o el precio original:

   if (controlOfertas.enOferta(pro))

5) Decisión de retornar error si el producto no está en el catálogo:

   return 101

6) Decisión de retornar error si el stock es insuficiente:

   return 102

Vamos a evaluar la cobertura de decisiones, asignando "C" (cierto) cuando la decisión se evalúa a cierto y "F" (falso) cuando se evalúa a falso.

Decisiones:

1) D1: Producto en catálogo (decisión de si existe el producto en el catálogo).
2) D2: Stock disponible suficiente (decisión de si la cantidad solicitada excede el stock).
3) D3: Producto en oferta (decisión de si el producto está en oferta).
4) D4: Precio de oferta o precio original (decisión de si se debe usar el precio de oferta o no).
5) D5: Error por producto no encontrado (decisión que retorna 101).
6) D6: Error por stock insuficiente (decisión que retorna 102).

A continuación se muestra la tabla:

Caso de Prueba	D1	D2	D3	D4	D5	D6
1	C	F	C	C	F	F
2	F	C	F	F	C	C
3	C	C	F	F	F	C
4	C	C	F	F	F	F
5	C	C	C	C	F	C
6	C	F	F	F	F	C
7	C	C	F	F	F	F
8	F	C	F	F	C	C
9	C	C	C	C	F	C
10	C	C	C	C	F	C
11	C	C	C	C	F	C
12	C	C	C	C	F	C
13	C	C	C	C	F	C
14	C	C	F	F	F	C
15	C	C	F	F	F	C

Tabla 11: Cobertura de Decisiones

3.4 Cobertura de Condiciones

Para realizar la cobertura de condiciones, debemos identificar las condiciones dentro del código y evaluar si cada caso de prueba cubre todas las posibles salidas (es decir, verdadero o falso) para cada una de estas condiciones.

En el caso del proyecto Carrito y las clases asociadas, las principales condiciones son:

1) Condición en introducirProducto (Producto en el catálogo):

   if (!catalogo.existeProducto(pro))

2) Condición en introducirProducto (Stock suficiente):

   if (unidades > pro.getStock())

3) Condición en introducirProducto (Producto en oferta):

   if (controlOfertas.enOferta(pro))

4) Condición en introducirProducto (Actualizar total en carrito):

   if (unidades > 0)

Las condiciones son las que se encuentran dentro de los condicionales (if) del código. Para cada una de estas condiciones, necesitamos evaluar si se cubren en los casos de prueba disponibles.

A continuación se presenta la tabla que indica C (si cubre la condición) o F (si no cubre la condición):

Caso de Prueba	C1	C2	C3	C4
1	C	F	C	C
2	F	C	F	C
3	C	C	F	C
4	C	C	F	C
5	C	C	C	C
6	C	F	F	C
7	C	C	F	C
8	F	C	F	C
9	C	C	C	C
10	C	C	C	C
11	C	C	C	C
12	C	C	C	C
13	C	C	C	C
14	C	C	F	C
15	C	C	F	C

Tabla 12: Cobertura de Condiciones

3.5 Cobertura de Múltiple Condición

La cobertura de condiciones múltiples se refiere a evaluar combinaciones de condiciones en las que varias condiciones deben ser evaluadas simultáneamente en una estructura condicional. En este caso, dado que estamos trabajando con una serie de condiciones que se encuentran dentro de decisiones múltiples (if), la cobertura de condiciones múltiples asegurará que todas las combinaciones de condiciones posibles sean evaluadas en los casos de prueba.

En el código, algunas de las decisiones tienen combinaciones de condiciones, por ejemplo:

1) Condición de Producto en el catálogo + Condición de Stock Suficiente:

   !catalogo.existeProducto(pro) + unidades > pro.getStock()

2) Condición de Producto en el catálogo + Producto en Oferta + Actualización del total en el carrito:

   !catalogo.existeProducto(pro) + controlOfertas.enOferta(pro) + unidades > 0

3) Condición de Producto en el catálogo + Producto en Oferta + Condición de Precio de Oferta:

   !catalogo.existeProducto(pro) + controlOfertas.enOferta(pro) + precio = controlOfertas.precioOferta(pro)

Vamos a crear la tabla comparando cada caso de prueba con las combinaciones de condiciones, y marcar con "C" cuando esa combinación de condiciones se evalúa correctamente y con "F" cuando no.

Condiciones Múltiples:

1) MC1: Producto en catálogo + Desigualdad Triangular
2) MC2: Producto en catálogo + Triángulo Válido
3) MC3: Producto en catálogo + Teorema de Pitágoras (Rectángulo)
4) MC4: Producto en catálogo + Triángulo Inválido
5) MC5: Producto en catálogo + Equilátero
6) MC6: Producto en catálogo + Isósceles
7) MC7: Producto en catálogo + Rectángulo
8) MC8: Producto en catálogo + Escaleno
9) MC9: Desigualdad Triangular + Triángulo Válido
10) MC10: Desigualdad Triangular + Teorema de Pitágoras (Rectángulo)
11) MC11: Desigualdad Triangular + Triángulo Inválido
12) MC12: Desigualdad Triangular + Escaleno

A continuación se presenta la tabla que indica C (si cubre la condición) o F (si no cubre la condición):

Caso de Prueba	MC1	MC2	MC3	MC4	MC5	MC6	MC7	MC8	MC9	MC10	MC11	MC12
1	F	C	F	F	C	F	F	F	F	F	F	F
2	F	C	F	F	C	C	F	F	F	F	F	F
3	F	C	F	F	C	C	F	F	F	F	F	F
4	F	C	F	F	C	F	F	C	C	F	C	F
5	F	C	F	F	C	C	F	F	C	C	F	F
6	C	F	F	F	C	F	F	F	F	F	F	F
7	C	F	F	F	C	F	F	F	F	F	F	F
8	F	F	F	F	C	F	F	F	F	F	F	F
9	F	F	F	F	C	F	F	F	F	F	F	F
10	F	C	F	F	C	C	F	F	C	C	F	F
11	F	C	F	F	C	C	F	F	C	C	F	F
12	F	C	F	F	C	C	F	F	C	C	F	F
13	F	C	F	F	C	C	F	F	C	C	F	F
14	C	F	F	F	C	F	F	F	F	F	F	F
15	F	C	F	F	C	F	F	C	F	F	F	F

Tabla 13: Cobertura de Múltiple Condición