|  |  |  |
| --- | --- | --- |
| Universidad de Oviedo  Escuela Politécnica de Ingeniería de Gijón  Departamento de Informática  GIJÓN |  | Prácticas de  Ingeniería  del Software |

|  |  |  |  |
| --- | --- | --- | --- |
| **Práctica Pruebas** | | | |
|  |  |  |  |
|  | **Nombre corto:** |  | PrPru |
|  | **Código del equipo:** |  | IS2022G41 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Pruebas caja blanca** | | | |
|  |  |  |  |
|  | **Doc. Id.:** |  | Pru |
|  | **Versión:** |  | 1.0 |
|  |  |  |  |
|  | **Fecha:** |  | 04/01/2023 |
|  | **Entregado por:** |  | Juan Francisco Mier Montoto |
|  |  |  |  |
|  | **Escrito por:** |  | Juan Francisco Mier Montoto |
|  |  |  |  |
|  | **Carácter:** |  | Definitivo |

Tabla de Contenidos

1. Introducción 3

2. Casos de prueba para "Triángulo” 3

2.1. Cobertura de sentencias 3

2.2. Cobertura de decisión 4

2.3. Cobertura de condiciones 5

3. Casos prueba para “Carrito” 6

3.1. Cobertura de sentencias 6

3.2. Cobertura de decisiones 6

Historia

|  |  |  |
| --- | --- | --- |
| Versión | Fecha | Cambios introducidos |
| 1.0 | 04/01/2023 | Versión inicial. |

# 1. Introducción

# 2. Casos de prueba para "Triángulo”

## 2.1. Cobertura de sentencias

public String getTriangleType () {

if (! isTriangle())

return "Inválido"; # sentencia A

else if (side1 == side2 && side2 == side3)

return "Equilátero"; # sentencia B

else if (side1 == side2 || side2 == side3 || side1 == side3)

return "Isósceles"; # sentencia C

else if(isRightTriangle())

return "Rectángulo"; # sentencia D

else

return "Escaleno"; # sentencia E

}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Objetivo | Situación inicial | Valores de entrada | Salida esperada | Sentencia cubierta |
| T1 | Probar un equilátero válido | N/A | Lado 1 = 3, Lado 2 = 3, Lado 3 = 3 | Equilátero | B |
| T2 | Probar con un triángulo inválido | N/A | Lado 1 = 1, Lado 2 = 1, Lado 3 = 8 | Inválido | A |
| T3 | Probar con un isósceles válido | N/A | Lado 1 = 3, Lado 2 = 3, Lado 3 = 4 | Isósceles | C |
| T4 | Probar con un triángulo rectángulo válido | N/A | Lado 1 = 5, Lado 2 = 13, Lado 3 = 12 | Rectángulo | D |
| T5 | Probar con un escaleno válido | N/A | Lado 1 = 2, Lado 2 = 3, Lado 3 = 4 | Escaleno | E |

## 2.2. Cobertura de decisión

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| L1 | L2 | L3 | !isTriangle() | side1 == side2 && side2 == side3 | side1 == side2 || side2 == side3 || side1 == side3 | isRightTriangle() |
| 1 | 1 | 4 | C | - | - | - |
| 3 | 3 | 3 | F | C | - | - |
| 3 | 3 | 4 | F | F | C | - |
| 5 | 12 | 13 | F | F | F | C |
| 2 | 3 | 4 | F | F | F | F |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Objetivo | Situación Inicial | Valores de entrada | Salida esperada | Decisión cumplida |
| T1 | Verificar la primera decisión | N/A | Lado 1 = 1, Lado 2 = 1, Lado 3 = 4 | Inválido | !isTriangle() |
| T2 | Verificar la segunda decisión | N/A | Lado 1 = 1, Lado 2 = 1, Lado 3 = 8 | Equilátero | side1 == side2 && side2 == side3 |
| T3 | Verificar la tercera decisión | N/A | Lado 1 = 3, Lado 2 = 3, Lado 3 = 4 | Isósceles | side1 == side2 || side2 == side3 || side1 == side3 |
| T4 | Verificar la cuarta decisión | N/A | Lado 1 = 5, Lado 2 = 13, Lado 3 = 12 | Rectángulo | isRightTriangle() |
| T5 | Verificar la quinta decisión | N/A | Lado 1 = 2, Lado 2 = 3, Lado 3 = 4 | Escaleno | Ninguna |

## 2.3. Cobertura de condiciones

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| L1 | L2 | L3 | !isTriangle() | side1 == side2 | side2 == side3 | side1 == side3 | isRightTriangle() |
| 1 | 1 | 4 | C | - | - | - | - |
| 3 | 3 | 3 | F | C | C | - | - |
| 3 | 3 | 4 | F | C | F | - | - |
| 4 | 3 | 3 | F | F | C | - | - |
| 3 | 4 | 3 | F | F | F | C | - |
| 5 | 12 | 13 | F | F | F | F | C |
| 2 | 3 | 4 | F | F | F | F | F |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Objetivo | Situación Inicial | Valores de entrada | Salida esperada | Condición cumplida |
| T1 | Verificar la primera condición | N/A | Lado 1 = 1, Lado 2 = 1, Lado 3 = 4 | Inválido | !isTriangle() |
| T2 | Verificar la segunda condición | N/A | Lado 1 = 3, Lado 2 = 3, Lado 3 = 3 | Equilátero | side1 == side2 && side2 == side3 |
| T3 | Verificar la tercera condición | N/A | Lado 1 = 3, Lado 2 = 3, Lado 3 = 4 | Isósceles | side1 == side2 |
| T4 | Verificar la cuarta condición | N/A | Lado 1 = 4, Lado 2 = 3, Lado 3 = 3 | Isósceles | side2 == side3 |
| T5 | Verificar la quinta condición | N/A | Lado 1 = 3, Lado 2 = 4, Lado 3 = 3 | Isósceles | side1 == side3 |
| T6 | Verificar la sexta condición | N/A | Lado 1 = 5, Lado 2 = 13, Lado 3 = 12 | Rectángulo | isRightTriangle() |
| T7 | Verificar la séptima condición | N/A | Lado 1 = 2, Lado 2 = 3, Lado 3 = 4 | Escaleno | Ninguna. |

No es posible obtener cobertura de múltiples condiciones porque una vez conseguido cumplir side1 == side2 y side2 == side3, nunca se va a evaluar side1 == side3, por lo que no es necesario analizar dicha cobertura.

# 3. Casos prueba para “Carrito”

## 3.1. Cobertura de sentencias

if (!catalogo.existeProducto(pro)) return 101; # sentencia A

if (unidades > pro.getStock()) return 102; # sentencia B

if (controlOfertas.enOferta(pro)) precio = controlOfertas.precioOferta(pro); # sentencia C

else precio = pro.getPrecio(); # sentencia D

ItemCarrito it = new ItemCarrito(pro.getCodigo(), unidades, precio);

car.nuevoItem(it);

catalogo.decrementarStock(pro, unidades);

Tampoco hay múltiple condición en este caso.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Objetivo | Situación inicial | Valores de entrada | Salida esperada | Sentencia cubierta |
| T1 | Probar un producto que no existe | N/A | Id de producto inválida, como “-1”. | “Error: el producto no existe en el catálogo.” | Sentencia A |
| T2 | Probar un producto que no está en stock | N/A | Valor de stock inferior a la cantidad, como por ejemplo “0”. | “Error: no existe disponibilidad.” | Sentencia B |
| T3 | Probar un producto que está en oferta | N/A | Id de producto válido, con stock suficiente, presente en oferta en el catálogo. | “-“ (Se agrega el producto normalmente) | Sentencia C |
| T4 | Probar un producto | N/A | Id de producto válido, con stock suficiente, sin oferta en el catálogo. | “-“ (Se agrega el producto normalmente) | Sentencia D |

## 3.2. Cobertura de decisiones

En este caso, la cobertura de decisiones es igual a la cobertura de condiciones, puesto que las condiciones están compuestas por una sola condición cada una. De igual forma, no existe cobertura de múltiples condiciones ya que no es posible evaluar diferentes condiciones simultáneamente.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | !catalogo.existeProducto(pro) | unidades > pro.getStock() | controlOfertas.enOferta(pro) |
| T1 | C | F | F |
| T2 | F | C | F |
| T3 | F | F | C |
| T4 | F | F | F |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Objetivo | Situación inicial | Valores de entrada | Salida esperada | Decisión cumplida |
| T1 | Verificar la primera decisión | N/A | Id de producto inválida, como “-1”. | Inválido (error 101) | !catalogo.existeProducto(Pro) |
| T2 | Verificar la segunda decisión | N/A | Valor de stock inferior a la cantidad, como por ejemplo “0”. | Inválido (error 102) | unidades > pro.getStock() |
| T3 | Verificar la tercera decisión | N/A | Id de producto válido, con stock suficiente, presente en oferta en el catálogo. | Ninguna | controlOfertas.enOferta(pro) |
| T4 | Verificar la cuarta decisión | N/A | Id de producto válido, con stock suficiente, sin oferta en el catálogo. | Ninguna | Ninguna |