***Pruebas de Software***

Alejandro Mendoza Vargas

A01235311

**Investigación de pruebas de software**

1. **Pruebas Unitarias:** estas pruebas consisten en fragmentar el código y comprobar que funcione cada fragmento correctamente
2. **Pruebas de Integración:** estas pruebas consisten en integrar diferentes componentes o fragmentos y comprobar que funcionen correctamente
3. **Pruebas de Sistema:** estas pruebas se utilizan para comprobar que el sistema en general esté funcionando correctamente
4. **Pruebas de Regresión:** estas pruebas se realizan cada vez que cambia el código, se utilizan para corroborar que los cambios no han afectado negativamente el funcionamiento del software
5. **Pruebas Automatizadas:** estas pruebas consisten en la aplicación de herramientas de software para automatizar el proceso de revisión del software
6. **Pruebas bajo condiciones frontera:** este tipo de pruebas consiste en utilizar inputs que se encuentran en los extremos de los requerimientos para probar el comportamiento del software

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tipo de prueba** | **Nombre** | **Licencia** | **Lenguajes** | **Particularidades** |
| Unitaria | NUnit | Open Source | .NET | Viene de un software que se llama Junit, utilizado para Java |
| Unitaria | Embunit | Comercial | C y C++ | Te crea el código para las pruebas automáticamente |
| Unitaria | SimpleTest | Open Source | PHP | Permite crear los casos de prueba en scripts ejecutables |
| Integración | Hound | Open Source | JavaScript | Puede correr varias sesiones de browser en sucesión rápida |
| Integración | Buildbot | Open Source | Python | Se pueden configurar los “builders” para hacer diferentes trabajos |
| Integración | Testium | Open Source | CoffeeScript | Te permite hacer pruebas usando sintaxis similar a BDD |
| Integración | VectorCAST | Comercial | C o C++ | -Valida los sistemas integrados críticos para la seguridad y el negocio  -Se usa ampliamente en industrias financieras, dispositivos médicos, controles industriales, ferrocarriles. |
| Integración | Citrus | Open Source | Java | -Establece secuencia de mensajes y crea mensajes de error  -Enviar y recibir mensajes  -Espere el mensaje y active otro mensaje |
| Integración | LDRA | Open Source | C  C++  Java  Ada95 | -Las pruebas se pueden generar y ejecutar fácilmente  -Variedad de soporte para que las pruebas tengan un entorno común para una amplia gama de proyectos |
| Sistema | Selenium | Open Source | Java, Ruby, C#, Python y más | Permite la ejecución de texto en paralelo |
| Sistema | Appium | Open Source | Java, JavaScript, PHP, Ruby, Python, C#, etc. | Es cross-platform |
| Sistema | Load Runner | Licencia, pero cuenta con una versión gratuita | HTML5, MQTT, DB, .NET y Java | Cuenta con una curva de aprendizaje pronunciada |

**Repositorio:** <https://github.com/alemendoza-v/PiedraPapelTijera>

**Identificador: TEST-01**

Nombre: Presionar botón papel

Escenario:

DADO: Un botón con imagen que corresponde a papel

CUANDO: Presiono ese botón

ENTONCES: Se debe mostrar que escogí papel

Instrucciones:

1. Abrir la app
2. Presionar el botón papel
3. Validar que la opción es papel

Entradas:

-Click al botón papel

Salidas Esperadas:

-Jugador escogió: “Papel”

**Identificador: TEST-02**

Nombre: Presionar botón piedra

Escenario:

DADO: Un botón con imagen que corresponde a piedra

CUANDO: Presiono ese botón

ENTONCES: Se debe mostrar que escogí piedra

Instrucciones:

1. Abrir la app
2. Presionar el botón piedra
3. Validar que la opción es piedra

Entradas:

-Click al botón piedra

Salidas Esperadas:

-Jugador escogió: “Piedra”

**Identificador: TEST-03**

Nombre: Presionar botón tijeras

Escenario:

DADO: Un botón con imagen que corresponde a tijeras

CUANDO: Presiono ese botón

ENTONCES: Se debe mostrar que escogí tijeras

Instrucciones:

1. Abrir la app
2. Presionar el botón tijeras
3. Validar que la opción es tijeras

Entradas:

-Click al botón tijeras

Salidas Esperadas:

-Jugador escogió: “Tijeras”

**Identificador: TEST-04**

Nombre: Test Init Partida

Escenario:

DADO: Un puntaje del jugador y un puntaje de la computadora

CUANDO: Creo un objeto de la clase Partida

ENTONCES: Se debe crear un objeto con los valores correspondientes

Instrucciones:

1. Correr la prueba

Entradas:

-puntajeJugador = 0

-puntajeComputadora = 1

Salidas Esperadas:

-objeto de tipo Partida con partida.puntajeJugador = 0 y partida.puntajeComputadora = 1

**Identificador: TEST-05**

Nombre: Test Init Turno

Escenario:

DADO: Una opción del jugador y una opción de la computadora

CUANDO: Creo un objeto de la clase Turno

ENTONCES: Se debe crear un objeto con los valores correspondientes

Instrucciones:

1. Correr la prueba

Entradas:

-opcionJugador = “Papel”

-opcionComputadora = “Tijeras”

Salidas Esperadas:

-objeto de tipo Turno con turno.opcionJugador = “Papel” y turno.opcionComputadora = “Tijeras”

**Identificador: TEST-06**

Nombre: Test Opcion Random

Escenario:

DADO: Un objeto de la clase OpcionRandom

CUANDO: Se llama a la función obtenerOpcion()

ENTONCES: Se debe crear una opción aleatoria

Instrucciones:

1. Correr la prueba

Entradas:

Salidas Esperadas:

-un String que corresponda a alguna de las opciones:

1. Papel

2. Tijeras

3. Piedra

**Identificador: TEST-07**

Nombre: Test Jugar Turno Piedra vs Papel

Escenario:

DADO: Un objeto de tipo Partida con puntajes en 0 y un objeto de tipo Turno que contenga: una opción de jugador que corresponde a piedra y una opción de computadora que corresponde a papel

CUANDO: Se llama a la función Jugar()

ENTONCES: Se debe checar quien gana y cambiar el puntaje de la partida

Instrucciones:

1. Crear objeto de tipo Partida con puntajes en 0
2. Crear objeto de tipo Turno con opcionJugador = “Piedra” y opcionComputadora = “Papel”
3. Crear objeto de tipo JugarTurno con los parámetros necesarios (partida y turno)
4. Llamar la función Jugar()

Entradas:

-Un objeto de tipo partida con puntajes en 0

-Un objeto de tipo turno con opciones: opcionJugador = “Piedra” y opcionComputadora = “Papel”

Salidas Esperadas:

-String representando el ganador: “Computadora”

-Puntajes cambiados correctamente en el objeto de tipo Partida (puntajeComputadora + 1 = 1)

**Identificador: TEST-08**

Nombre: Test Jugar Turno Tijeras vs Papel

Escenario:

DADO: Un objeto de tipo Partida con puntajes en 0 y un objeto de tipo Turno que contenga: una opción de jugador que corresponde a tijeras y una opción de computadora que corresponde a papel

CUANDO: Se llama a la función Jugar()

ENTONCES: Se debe checar quien gana y cambiar el puntaje de la partida

Instrucciones:

1. Crear objeto de tipo Partida con puntajes en 0
2. Crear objeto de tipo Turno con opcionJugador = “Tijeras” y opcionComputadora = “Papel”
3. Crear objeto de tipo JugarTurno con los parámetros necesarios (partida y turno)
4. Llamar la función Jugar()

Entradas:

-Un objeto de tipo partida con puntajes en 0

-Un objeto de tipo turno con opciones: opcionJugador = “Tijeras” y opcionComputadora = “Papel”

Salidas Esperadas:

-String representando el ganador: “Jugador”

-Puntajes cambiados correctamente en el objeto de tipo Partida (puntajeJugador + 1 = 1)

**Identificador: TEST-09**

Nombre: Test Jugar Turno Empate

Escenario:

DADO: Un objeto de tipo Partida con puntajes en 0 y un objeto de tipo Turno que contenga: una opción de jugador que corresponde a tijeras y una opción de computadora que corresponde a tijeras

CUANDO: Se llama a la función Jugar()

ENTONCES: Se debe checar quien gana y cambiar el puntaje de la partida

Instrucciones:

1. Crear objeto de tipo Partida con puntajes en 0
2. Crear objeto de tipo Turno con opcionJugador = “Tijeras” y opcionComputadora = “Tijeras”
3. Crear objeto de tipo JugarTurno con los parámetros necesarios (partida y turno)
4. Llamar la función Jugar()

Entradas:

-Un objeto de tipo partida con puntajes en 0

-Un objeto de tipo turno con opciones: opcionJugador = “Tijeras” y opcionComputadora = “Tijeras”

Salidas Esperadas:

-String representando el ganador: “Empate”

-Sin cambios en los puntajes (ambos puntajes = 0)

**Test Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Nombre | Salida Esperada | Salida Obtenida | P/F |
| 01 | Presionar botón papel | Papel | Papel | SI |
| 02 | Presionar botón piedra | Piedra | Piedra | SI |
| 03 | Presionar botón tijeras | Tijeras | Tijeras | SI |
| 04 | Test Init Partida | objeto de tipo Partida con partida.puntajeJugador = 0 y partida.puntajeComputadora = 1 | objeto de tipo Partida con partida.puntajeJugador = 0 y partida.puntajeComputadora = 1 | SI |
| 05 | Test Init Turno | objeto de tipo Turno con turno.opcionJugador = “Papel” y turno.opcionComputadora = “Tijeras” | objeto de tipo Turno con turno.opcionJugador = “Papel” y turno.opcionComputadora = “Tijeras” | SI |
| 06 | Test Opcion Random | un String que corresponda a alguna de las opciones | un String que corresponda a alguna de las opciones | SI |
| 07 | Test Jugar Turno Piedra vs Papel | -String representando el ganador: “Computadora”  -Puntajes cambiados correctamente en el objeto de tipo Partida (puntajeComputadora + 1 = 1) | -String representando el ganador: “Computadora”  -Puntajes cambiados correctamente en el objeto de tipo Partida (puntajeComputadora + 1 = 1) | SI |
| 08 | Test Jugar Turno Tijeras vs Papel | -String representando el ganador: “Jugador”  -Puntajes cambiados correctamente en el objeto de tipo Partida (puntajeJugador + 1 = 1) | -String representando el ganador: “Jugador”  -Puntajes cambiados correctamente en el objeto de tipo Partida (puntajeJugador + 1 = 1) | SI |
| 09 | Test Jugar Turno Empate | -String representando el ganador: “Empate”  -Sin cambios en los puntajes (ambos puntajes = 0) | -String representando el ganador: “Empate”  -Sin cambios en los puntajes (ambos puntajes = 0) | SI |