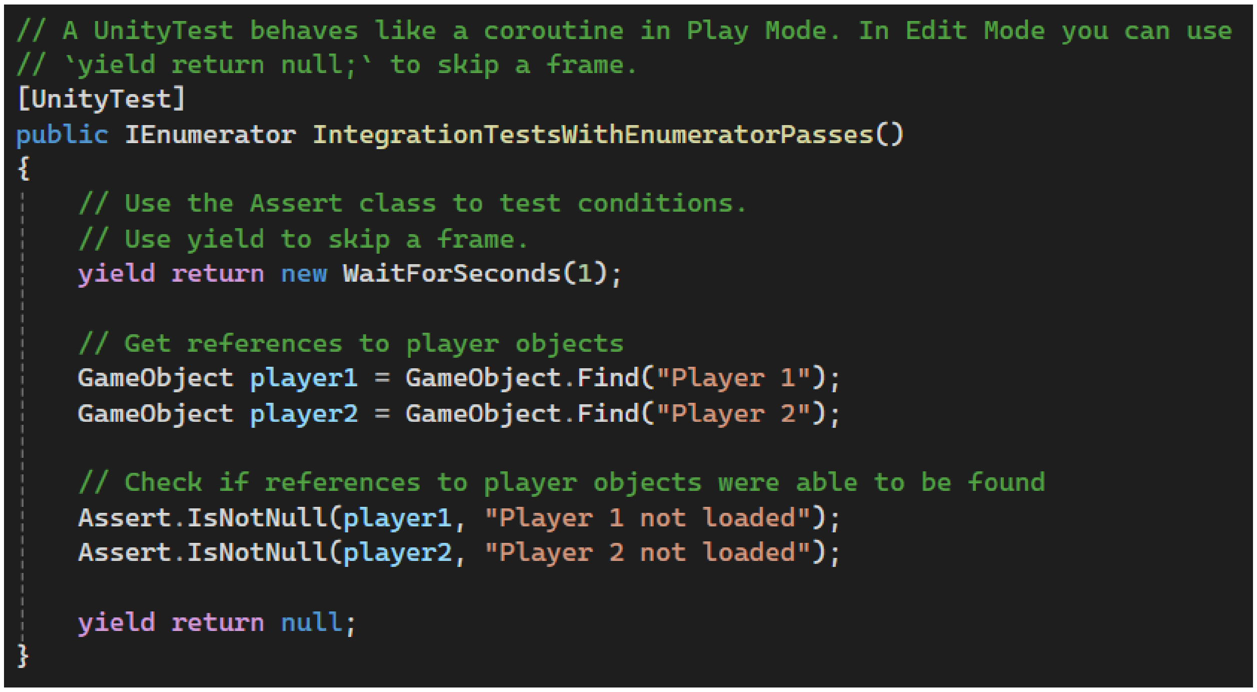
1.

Players loaded (Valid): not null
Players loaded (Invalid): null

Code:

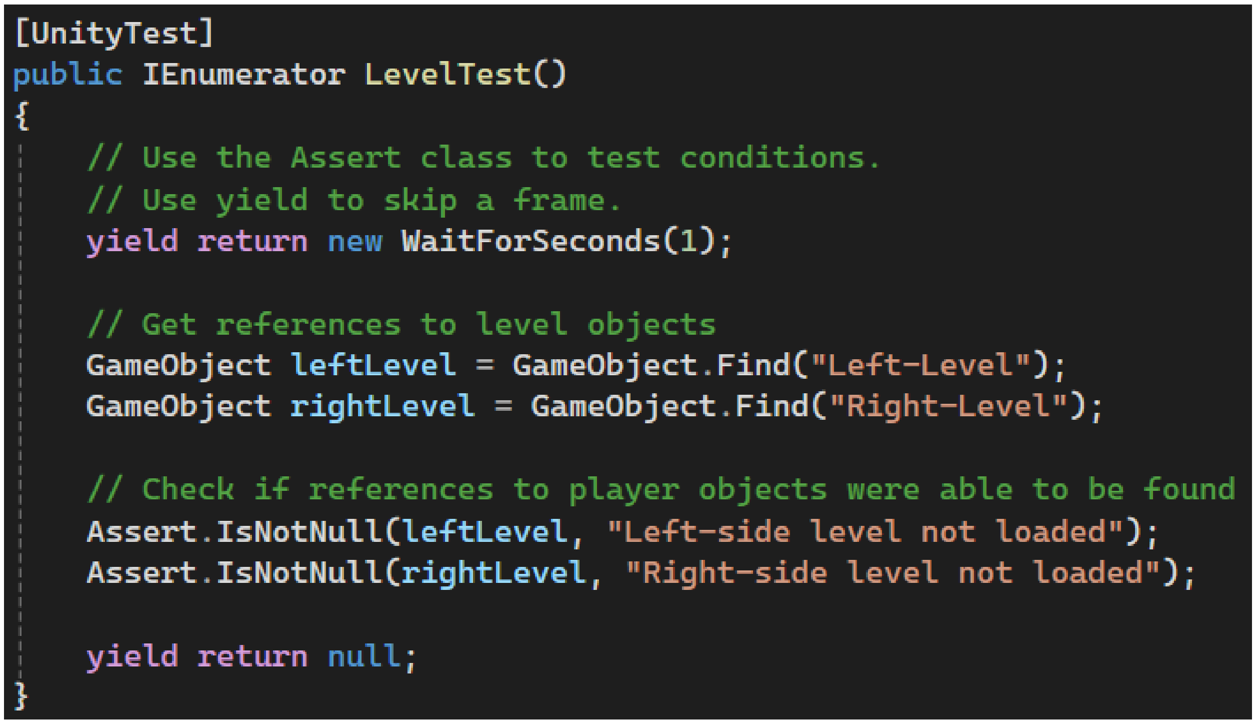


```
// A UnityTest behaves like a coroutine in Play Mode. In Edit Mode you can use
// `yield return null;` to skip a frame.
[UnityTest]
public IEnumerator IntegrationTestsWithEnumeratorPasses()
{
    // Use the Assert class to test conditions.
    // Use yield to skip a frame.
    yield return new WaitForSeconds(1);

    // Get references to player objects
    GameObject player1 = GameObject.Find("Player 1");
    GameObject player2 = GameObject.Find("Player 2");

    // Check if references to player objects were able to be found
    Assert.IsNotNull(player1, "Player 1 not loaded");
    Assert.IsNotNull(player2, "Player 2 not loaded");

    yield return null;
}
```

2.

Level loaded (Valid): not null
Level loaded (Invalid): null

Code:



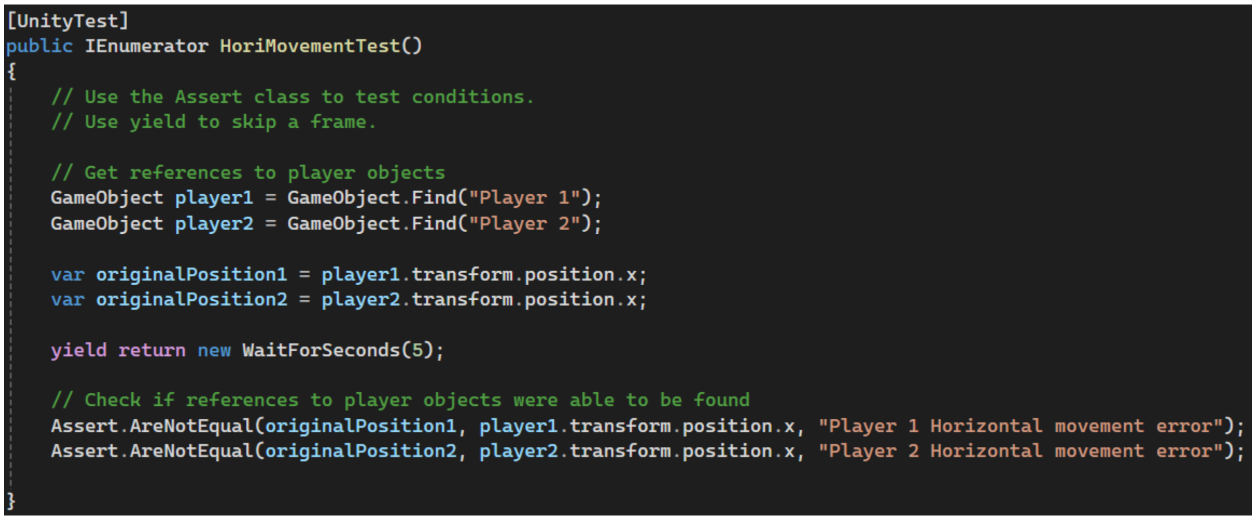
```
[UnityTest]
public IEnumerator LevelTest()
{
    // Use the Assert class to test conditions.
    // Use yield to skip a frame.
    yield return new WaitForSeconds(1);

    // Get references to level objects
    GameObject leftLevel = GameObject.Find("Left-Level");
    GameObject rightLevel = GameObject.Find("Right-Level");

    // Check if references to player objects were able to be found
    Assert.IsNotNull(leftLevel, "Left-side level not loaded");
    Assert.IsNotNull(rightLevel, "Right-side level not loaded");

    yield return null;
}
```

3.

Player 1 Horizontal movement (Valid): current x position != original x position
Player 2 Horizontal movement (Valid): current x position != original x position

Player 1 Horizontal movement (Invalid): current x position = original x position
Player 2 Horizontal movement (Invalid): current x position = original x position

Code:



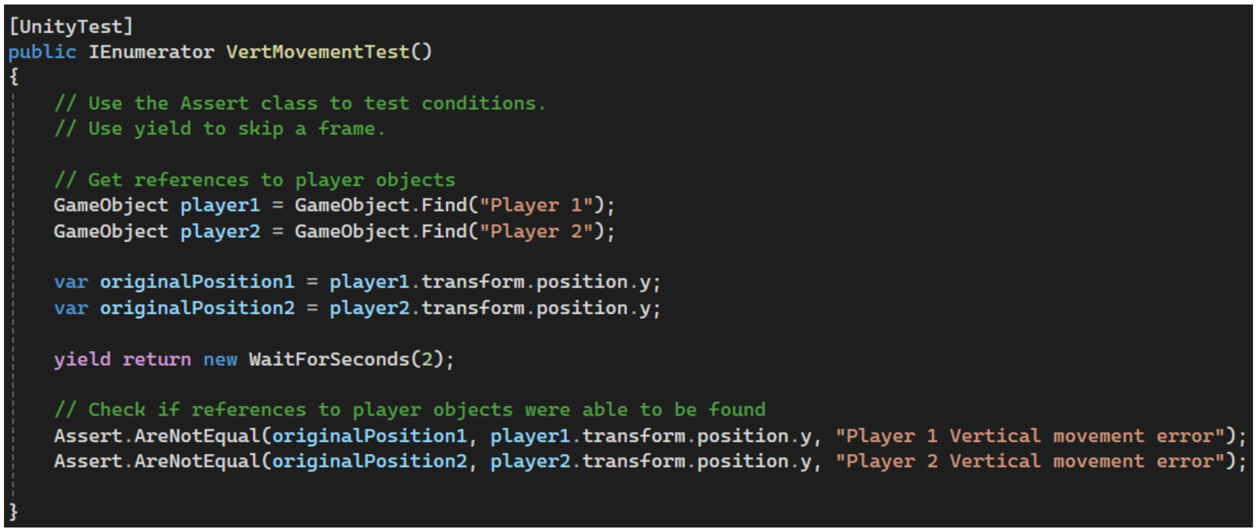
```
[UnityTest]
public IEnumerator HoriMovementTest()
{
    // Use the Assert class to test conditions.
    // Use yield to skip a frame.

    // Get references to player objects
    GameObject player1 = GameObject.Find("Player 1");
    GameObject player2 = GameObject.Find("Player 2");

    var originalPosition1 = player1.transform.position.x;
    var originalPosition2 = player2.transform.position.x;

    yield return new WaitForSeconds(5);

    // Check if references to player objects were able to be found
    Assert.AreNotEqual(originalPosition1, player1.transform.position.x, "Player 1 Horizontal movement error");
    Assert.AreNotEqual(originalPosition2, player2.transform.position.x, "Player 2 Horizontal movement error");

}
```

4.

Player 1 Vertical movement (Valid): current y position != original y position
Player 2 Vertical movement (Valid): current y position != original y position

Player 1 Vertical movement (Invalid): current y position = original y position
Player 2 Vertical movement (Invalid): current y position = original y position

Code:



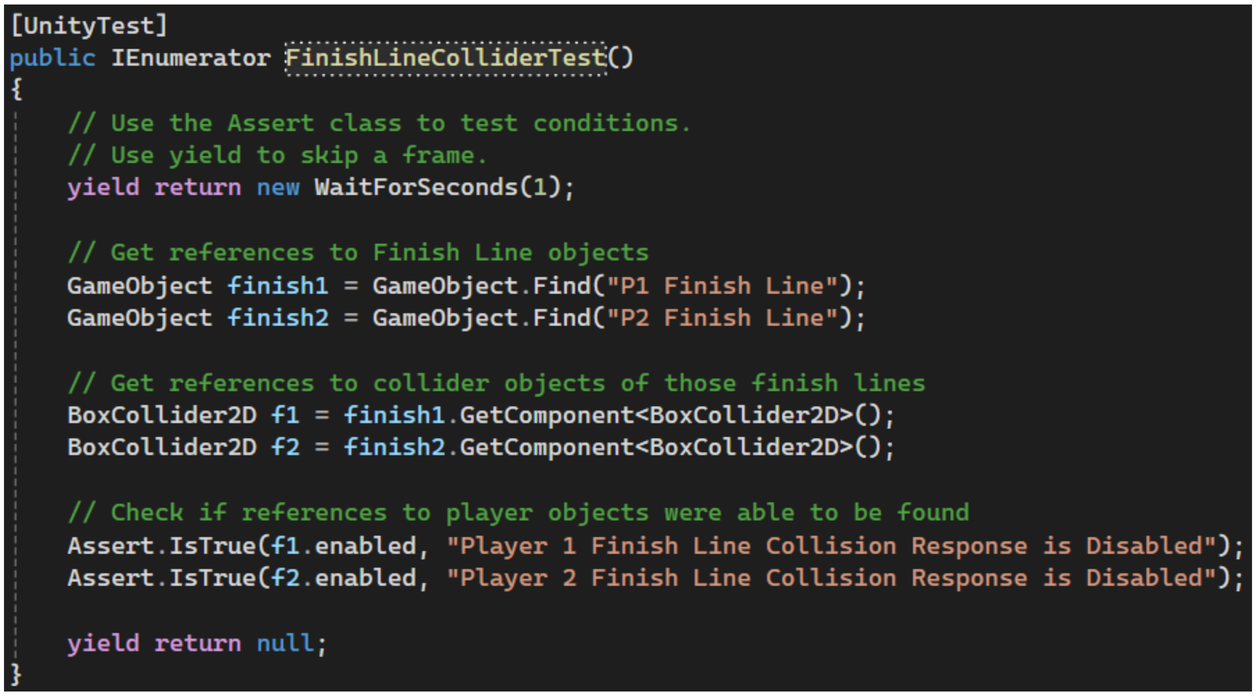
```
[UnityTest]
public IEnumerator VertMovementTest()
{
    // Use the Assert class to test conditions.
    // Use yield to skip a frame.

    // Get references to player objects
    GameObject player1 = GameObject.Find("Player 1");
    GameObject player2 = GameObject.Find("Player 2");

    var originalPosition1 = player1.transform.position.y;
    var originalPosition2 = player2.transform.position.y;

    yield return new WaitForSeconds(2);

    // Check if references to player objects were able to be found
    Assert.AreNotEqual(originalPosition1, player1.transform.position.y, "Player 1 Vertical movement error");
    Assert.AreNotEqual(originalPosition2, player2.transform.position.y, "Player 2 Vertical movement error");

}
```

5.

Player 1 Finish Line Collider Component Enabled (Valid): True
Player 2 Finish Line Collider Component Enabled (Valid): True

Player 1 Finish Line Collider Component Enabled (Invalid): False
Player 2 Finish Line Collider Component Enabled (Invalid): False

Code:



```
[UnityTest]
public IEnumerator FinishLineColliderTest()
{
    // Use the Assert class to test conditions.
    // Use yield to skip a frame.
    yield return new WaitForSeconds(1);

    // Get references to Finish Line objects
    GameObject finish1 = GameObject.Find("P1 Finish Line");
    GameObject finish2 = GameObject.Find("P2 Finish Line");

    // Get references to collider objects of those finish lines
    BoxCollider2D f1 = finish1.GetComponent<BoxCollider2D>();
    BoxCollider2D f2 = finish2.GetComponent<BoxCollider2D>();

    // Check if references to player objects were able to be found
    Assert.IsTrue(f1.enabled, "Player 1 Finish Line Collision Response is Disabled");
    Assert.IsTrue(f2.enabled, "Player 2 Finish Line Collision Response is Disabled");

    yield return null;
}
```

6.

Player Hit Response (Valid): current x position (after hit) = original x position
Player Hit Response (Invalid): current x position (after hit) != original x position

7.

Progress Bar 1 Matches Player 1 Position (Valid): progressBarValue = playerXPosition
Progress Bar 1 Matches Player 1 Position (Invalid): progressBarValue != playerXPosition

8.

Progress Bar 2 Matches Player 2 Position (Valid): progressBarValue = abs(playerXPosition)
Progress Bar 2 Matches Player 2 Position (Invalid): progressBarValue != abs(playerXPosition)

9.

Player Gravity (Valid): current y position < original y position
Player Gravity (Invalid): current y position < original y position

10.

Wall slide slow fall (Valid): current y position (touching wall) > current y position (not touching wall)
Wall slide slow fall (Invalid): current y position (touching wall) > current y position (not touching wall)