**4.2. SwinAdventure Iteration 2 (Other files are included in my submission as .cs files)**

**I fixed \_identifiers.Add(identifiers[i].ToLower()); into addIdentifier(identifiers[i].ToLower());**

**Identifiable Object**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SwinAdventure

{

public abstract class IdentifiableObject

{

private List<string> \_identifiers = new List<string>();

public IdentifiableObject(string[] idents)

{

foreach (string id in idents)

{

AddIdentifier(id.ToLower());

}

}

public bool AreYou(string id)

{

return \_identifiers.Contains(id.ToLower());

}

public string FirstId

{

get

{

if (\_identifiers.Count > 0)

{

return \_identifiers[0];

}

else

{

return "";

}

}

}

public void AddIdentifier(string id)

{

\_identifiers.Add(id.ToLower());

}

}

}

**Test File**

using SwinAdventure;

namespace TestQueue

{

public class Tests

{

Item item1 = new Item(new string[] { "sword" }, "sword", "a sword");

Item item2 = new Item(new string[] { "shield" }, "shield", "a shield");

[SetUp]

public void Setup()

{

}

[Test]

public void ItemIdentifiable()

{

Assert.IsTrue(item1.AreYou("sword"));

}

[Test]

public void ShortDescription()

{

Assert.That(item1.ShortDescription, Is.EqualTo("sword (sword)"));

}

[Test]

public void FullDescription()

{

Assert.That(item1.FullDescription, Is.EqualTo("a sword"));

}

[Test]

public void FindItem()

{

Inventory inventory = new Inventory();

inventory.Put(item1);

Assert.IsTrue(inventory.HasItem("sword"));

}

[Test]

public void NoItem()

{

Inventory inventory = new Inventory();

Assert.IsFalse(inventory.HasItem("sword"));

}

[Test]

public void FetchItem()

{

Inventory inventory = new Inventory();

inventory.Put(item1);

Assert.That(item1, Is.EqualTo(inventory.Fetch("sword")));

Assert.IsTrue(inventory.HasItem("sword"));

}

[Test]

public void TakeItem()

{

Inventory inventory = new Inventory();

inventory.Put(item1);

Assert.That(item1, Is.EqualTo(inventory.Take("sword")));

Assert.IsFalse(inventory.HasItem("sword"));

}

[Test]

public void ItemList()

{

Inventory inventory = new Inventory();

inventory.Put(item1);

inventory.Put(item2);

//the list string below is the expected output, consisting of every item in the following format: name ( first id)

Assert.That(inventory.ItemList, Is.EqualTo("sword (sword)\nshield (shield)\n"));

}

[Test]

public void PlayerIdentifiable()

{

Player player = new Player("Tan", "A player");

Assert.IsTrue(player.AreYou("me"));

Assert.IsTrue(player.AreYou("inventory"));

}

[Test]

public void PlayerLocate()

{

Player player = new Player("Tan", "A player");

player.Inventory.Put(item1);

Assert.That(item1, Is.EqualTo(player.Locate("sword")));

}

[Test]

public void PlayerLocateItself()

{

Player player = new Player("Tan", "A player");

Assert.That(player, Is.EqualTo(player.Locate("me")));

Assert.That(player, Is.EqualTo(player.Locate("inventory")));

}

[Test]

public void PlayerLocateNothing()

{

Player player = new Player("Tan", "A player");

Assert.That(player.Locate("sword"), Is.Null);

}

[Test]

public void PlayerFullDescription()

{

Player player = new Player("Tan", "A player");

player.Inventory.Put(item1);

player.Inventory.Put(item2);

//the list string below is the expected output, consisting of every item in the following format: name ( first id)

Assert.That(player.FullDescription, Is.EqualTo("You are Tan A player\nYou are carrying:\nsword (sword)\nshield (shield)\n"));

}

}

}

**Test Results**

A screenshot of a computer

Description automatically generated