GameObject code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{

    public class GameObject : IdentifiableObject
    {
        private string _description;
        private string _name;



        public GameObject(string[] ids, string name, string description) : base(ids)
        {
            _name = name;
            _description = description;
        }



        public string Name
        {
            get { return _name; }
        }



        public string ShortDescription
        {
```

```
            get { return _description; }
        }

        public virtual string GetFullDescription()
        { return $"{_name}: {_description}"; }
    }
}
```

Item code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SwinAdventure
{
    public class Item : GameObject
    {
        public Item(string[]idents,string name,string description) : base(idents,name,description)

        {


        }
    }
}
```

Inventory code

```
using System.Collections.Generic;
using System.Linq;

namespace SwinAdventure
{
```

```
public class Inventory
{
    private List<Item> _items;

    public Inventory()
    {
        _items = new List<Item>();
    }

    public void Put(Item item)
    {
        if (item == null)
        {
            throw new ArgumentNullException(nameof(item), "Item cannot be null");
        }

        _items.Add(item);
        Console.WriteLine($"Item '{item.Name}' added to inventory.");
    }

    public Item Take(string id)
    {
        Item itemToTake = Fetch(id);
        if (itemToTake != null)
        {
            _items.Remove(itemToTake);
        }
        return itemToTake;
    }

    public Item Fetch(string id)
```

```
        {
            foreach (Item item in _items)
            {
                if (item.AreYou(id))
                {
                    return item;
                }
            }
            return null;
        }

        public bool HasItem(string id)
        {
            return Fetch(id) != null;
        }

        public string ItemList
        {
            get
            {
                string itemList = "";
                foreach (Item item in _items)
                {
                    itemList += $"\t{item.ShortDescription} ({item.FirstId})\n";
                }
                return itemList;
            }
        }
    }
}
```

Player code

```
using MiNET.Utils.Skins;
using System;

namespace SwinAdventure
{
    public class Player : GameObject
    {
        private Inventory _inventory;


        public Player(string name, string description) : base(new string[] { "me", "inventory" }, name, description)
        {
            _inventory = new Inventory();
        }

        public override string GetFullDescription()
        {

            return $" You are {Name}, a brave adventurer. You are holding" + _inventory.ItemList;
        }


        public GameObject Locate(string id)
        {
            if (AreYou(id))
            {
                return this;
            }
            return _inventory.Fetch(id);
```

```
        }



        public Inventory Inventory
        {
            get { return _inventory; }
        }
    }
}
```

Test Item code

```
using NUnit.Framework;
using SwinAdventure;
namespace SwinAdventure.Tests
{
    [TestFixture]
    public class ItemTests
    {
        private Item _item;

        [SetUp]
        public void Setup()
        {
            _item = new Item(new string[] { "sword", "weapon" }, "bronze sword", "a sharp bronze sword");
        }

        [Test]
        public void TestItemIsIdentifiable()
        {
            Assert.IsTrue(_item.AreYou("sword"));
            Assert.IsTrue(_item.AreYou("weapon"));
```

```
            Assert.IsFalse(_item.AreYou("shield"));
        }

        [Test]
        public void TestShortDescription()
        {
            Assert.AreEqual("a sharp bronze sword", _item.ShortDescription);
        }

        [Test]
        public void TestFullDescription()
        {
            Assert.AreEqual("bronze sword: a sharp bronze sword", _item.GetFullDescription());
        }
    }
}
```

Test inventory

```
using NUnit.Framework;
using SwinAdventure;

namespace SwinAdventure.Tests
{
    [TestFixture]
    public class InventoryTests
    {
        private Inventory _inventory;
        private Item _item;

        [SetUp]
        public void Setup()
```

```
    {
        _inventory = new Inventory();
        _item = new Item(new string[] { "sword", "weapon" }, "bronze sword", "a sharp bronze sword");
        _inventory.Put(_item);
    }

    [Test]
    public void TestFindItem()
    {
        Assert.IsTrue(_inventory.HasItem("sword"));
    }

    [Test]
    public void TestNoFindItem()
    {
        Assert.IsFalse(_inventory.HasItem("shield"));
    }

    [Test]
    public void TestFetchItem()
    {
        Item fetchedItem = _inventory.Fetch("sword");
        Assert.AreEqual(_item, fetchedItem);
        Assert.IsTrue(_inventory.HasItem("sword"));  // Fetch should not remove the item
    }

    [Test]
    public void TestTakeItem()
    {
        Item takenItem = _inventory.Take("sword");
```

```
            Assert.AreEqual(_item, takenItem);
            Assert.IsFalse(_inventory.HasItem("sword"));  // Item should be removed
        }

        [Test]
        public void TestItemList()
        {
            string expectedList = "\ta sharp bronze sword (sword)\n";
            Assert.AreEqual(expectedList, _inventory.ItemList);
        }
    }
}
```

Test Player code

```
using NUnit.Framework;
using System.Numerics;
using SwinAdventure;

namespace SwinAdventure.Tests
{
    [TestFixture]
    public class PlayerTests
    {
        private Player _player;
        private Item _item;

        [SetUp]
        public void Setup()
        {
            _player = new Player("John", "A brave adventurer");
            _item = new Item(new string[] { "sword", "weapon" }, "bronze sword", "a sharp bronze sword");
```

```
    _player.Inventory.Put(_item);
}

[Test]
public void TestPlayerIsIdentifiable()
{
    Assert.IsTrue(_player.AreYou("me"));
    Assert.IsTrue(_player.AreYou("inventory"));
}

[Test]
public void TestPlayerLocatesItems()
{
    Assert.AreEqual(_item, _player.Locate("sword"));
}

[Test]
public void TestPlayerLocatesItself()
{
    Assert.AreEqual(_player, _player.Locate("me"));
    Assert.AreEqual(_player, _player.Locate("inventory"));
}

[Test]
public void TestPlayerLocatesNothing()
{
    Assert.IsNull(_player.Locate("shield"));
}

[Test]
public void TestPlayerFullDescription()
```

```
        {
            string actualDescription = _player.GetFullDescription();
            string expectedDescription = " You are John, a brave adventurer. You are holding\ta sharp bronze sword (sword)\n";

            // Temporarily print both actual and expected descriptions to compare visually
            Console.WriteLine("Actual: " + actualDescription);
            Console.WriteLine("Expected: " + expectedDescription);

            Assert.AreEqual(expectedDescription, actualDescription);
        }
    }
}
```

Screenshot of test passed