COS 20007
Task 6.1

Duc Thuan Tran
104330455

I. Swin-Adventure

1. IdentifiableObject.cs

```
using System;
namespace Swin_Adventure
{
	public class IdentifiableObject
	{
		private List<string> _identifiers;

		public IdentifiableObject(string[] idents)
		{
			_identifiers = new List<string>(idents);
			_identifiers.AddRange(idents);
		}

		public bool AreYou(string id)
		{
			return _identifiers.Contains(id.ToLower());
		}

		public string FirstId
		{
			get
			{
				if (_identifiers.Count == 0)
				{
					return "";
				}
				return _identifiers[0];
			}
		}

		public void AddIdentifier(string id)
		{
			_identifiers.Add(id.ToLower());
		}
	}
}
```

2. GameObject.cs

```
using System;
using System.Xml.Linq;
```

```
namespace Swin_Adventure
{
        public class GameObject : IdentifiableObject
        {
    private string _description;
    private string _name;

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

    public string Name
                {
                        get { return _name.ToLower(); }
                }

                public string ShortDescription
                {
      get { return $"a {_name.ToLower()} ({FirstId.ToLower()})"; }
    }

    public virtual string FullDescription
                {
                        get { return _description; }
                }
        }
}
```

3. Inventory.cs

```
using System;
namespace Swin_Adventure
{
        public class Inventory
        {
                private List<Item> _items;
                public Inventory()
                {
                        _items = new List<Item>();
                }

                public bool HasItem(string id)
                {
                        foreach (Item itm in _items)
                        {
```

```
                        if (itm.AreYou(id))
                        {
                                return true;
                        }
                }
                return false;
        }

        public void Put(Item itm)
        {
                _items.Add(itm);
        }

public Item Take(string id)
{
   Item itm = Fetch(id);

   if (itm != null)
   {
      _items.Remove(itm);
   }

   return itm;
}

public Item Fetch(string id)
        {
                foreach (Item itm in _items)
                {
                        if (itm.AreYou(id))
                        {
                                return itm;
                        }
                }
                return null;
        }

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

```
        }
                }
         }
}
```

4. Item.cs

```
using System;

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

      }
   }
}
```

5. Player.cs

```
using System;
namespace Swin_Adventure
{
          public class Player : GameObject
          {
      private Inventory _inventory;

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

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

                public override string FullDescription
                {
                        get
                        {
            return "You are " + Name + ", " + base.FullDescription + ".\n"
                                + "You are carrying:\n" + Inventory.ItemList;
```

```
            }
                    }

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

6. Bag.cs

```
using System;
namespace Swin_Adventure
{
   public class Bag : Item
   {
      private Inventory _inventory;

      public Bag(string[] ids, string name, string description) : base(ids, name,
description)
      {
         _inventory = new Inventory();
      }

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

      public override string FullDescription
      {
         get { return $"In the {Name} you can see:\n" + _inventory.ItemList; }
      }

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

7. Command.cs

```
using System;
namespace Swin_Adventure
{
```

```
    public abstract class Command : IdentifiableObject
    {
        public Command(string[] ids) : base(ids)
        {
        }
         public abstract string Execute(Player player, string[] text);
    }
}
```

8. LookCommand.cs

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

namespace Swin_Adventure
{
    public class LookCommand : Command
    {
        public LookCommand() : base(new string[] { "look" }) { }

        public override string Execute(Player player, string[] text)
        {
            IHaveInventory container = null;
            string itemId;

            if (text.Length != 3 && text.Length != 5)
            {
                return "I don't know how to look like that";
            }
            else
            {
                if (text[0] != "look")
                {
                    return "Error in look input";
                }
                if (text[1] != "at")
                {
                    return "What do you want to look at?";
                }
                if (text.Length == 5 && text[3] != "in")
                {
                    return "What do you want to look in?";
                }

                switch (text.Length)
                {
```

```
                case 3:
                    container = player;
                    break;

                case 5:
                    container = FetchContainer(player, text[4]);

                    if (container == null)
                    {
                        return $"I can't find the {text[4]}";
                    }
                    break;
            }
            itemId = text[2];
            return LookAtIn(itemId, container);
        }
    }

    private IHaveInventory FetchContainer(Player player, string containerId)
    {
        return player.Locate(containerId) as IHaveInventory;
    }

    private string LookAtIn(string thingId, IHaveInventory container)
    {
        GameObject locatedObject = container.Locate(thingId);

        if (locatedObject != null)
        {
            return locatedObject.FullDescription;
        }
        else
        {
            return $"I can't find the {thingId}";
        }
    }
  }
}
```

9. IHaveInventory.cs

```
using System;
namespace Swin_Adventure
{
  public interface IHaveInventory
  {
    GameObject Locate(string id);
    string Name { get; }
  }
```

```
}
```

10. Program.cs

```
namespace Swin_Adventure;

class Program
{
   static void Main(string[] args)
   {
      Console.WriteLine("Duc Thuan Tran - 104330455");
   }
}
```

II. Swin-Adventure Test

1. IdentifiableObjectTest.cs

```
using NUnit.Framework;
using Swin_Adventure;

namespace IdentifiableObjectTest
{

   internal class Tests
   {
      private IdentifiableObject _test1;
      private IdentifiableObject _test2;
      private IdentifiableObject _test3;
      private IdentifiableObject _test4;
      private IdentifiableObject _test5;
      private IdentifiableObject _test6;

      [SetUp]
      public void Setup()
      {
         _test1 = new IdentifiableObject(new string[] { "fred", "bob" });
         _test2 = new IdentifiableObject(new string[] { "fred", "bob" });
         _test3 = new IdentifiableObject(new string[] { "fred", "bob" });
         _test4 = new IdentifiableObject(new string[] { "fred", "bob" });
         _test5 = new IdentifiableObject(new string[] { });
         _test6 = new IdentifiableObject(new string[] { "fred", "bob" });
      }

      [Test]
      public void TestAreYou()
      {
```

```
            Assert.IsTrue(_test1.AreYou("fred"));
            Assert.IsTrue(_test1.AreYou("bob"));
        }

        [Test]
        public void TestNotAreYou()
        {
            Assert.IsFalse(_test2.AreYou("wilma"));
            Assert.IsFalse(_test2.AreYou("boby"));
        }

        [Test]
        public void TestCaseSensitive()
        {
            Assert.IsTrue(_test3.AreYou("FRED"));
            Assert.IsTrue(_test3.AreYou("bOB"));
        }

        [Test]
        public void TestFirstID()
        {
            Assert.AreEqual("fred", _test4.FirstId);
        }

        [Test]
        public void TestFirstIdWithNoIDs()
        {
            Assert.AreEqual("", _test5.FirstId);
        }

        [Test]
        public void TestAddID()
        {
            _test6.AddIdentifier("wilma");
            Assert.IsTrue(_test6.AreYou("fred"));
            Assert.IsTrue(_test6.AreYou("bob"));
            Assert.IsTrue(_test6.AreYou("wilma"));
        }
    }
}
```

2. Inventory.cs

```
using System;
using Swin_Adventure;

namespace SwinAdventureTest
{
    [TestFixture]
```

```
public class InventoryTest
        {
                private Inventory _inventoryTest;
                private Item _weaponTest;
                private Item _armorTest;

                [SetUp]
                public void SetUp()
                {
                        _inventoryTest = new Inventory();
                        _weaponTest = new Item(new string[] { "weapon" }, "sword",
"this is a Excalibur");
        _armorTest = new Item(new string[] { "armor" }, "shield", "this is a shield");

        _inventoryTest.Put(_weaponTest);
        _inventoryTest.Put(_armorTest);
    }

    [Test]
    public void TestFindItem()
    {
      Assert.IsTrue(_inventoryTest.HasItem("weapon"));
      Assert.IsTrue(_inventoryTest.HasItem("armor"));
    }

    [Test]
    public void TestNoItemFind()
    {
      Assert.IsFalse(_inventoryTest.HasItem("axe"));
      Assert.IsFalse(_inventoryTest.HasItem("helmet"));
    }

    [Test]
    public void TestFetchItem()
    {
      Assert.IsTrue(_weaponTest == _inventoryTest.Fetch("weapon"));
      Assert.IsTrue(_inventoryTest.HasItem("weapon"));

      Assert.IsTrue(_armorTest == _inventoryTest.Fetch("armor"));
      Assert.IsTrue(_inventoryTest.HasItem("armor"));
    }

    [Test]
    public void TestTakeItem()
    {
      Assert.IsTrue(_weaponTest == _inventoryTest.Take("weapon"));
      Assert.IsFalse(_inventoryTest.HasItem("weapon"));
```

```
            Assert.IsTrue(_armorTest == _inventoryTest.Take("armor"));
            Assert.IsFalse(_inventoryTest.HasItem("armor"));
        }

        [Test]
        public void TestItemList()
        {
            Assert.IsTrue(_inventoryTest.ItemList.Replace("\t", "") == "a sword
(weapon)\na shield (armor)\n");
        }
    }
}
```

3. Item.cs

```
using System;
using Swin_Adventure;

namespace SwinAdventureTest
{
    [TestFixture]
    public class ItemTest
          {
                private Item _itemTest;


        [SetUp]
        public void Setup()
        {
            _itemTest = new Item(new string[] { "weapon" }, "sword", "This is an
Excalibur");

        }

        [Test]
        public void TestItemIsIdentifiable()
        {
            Assert.IsTrue(_itemTest.AreYou("weapon"));
        }

        [Test]
        public void TestShortDescription()
        {
            Assert.IsTrue(_itemTest.ShortDescription == "a sword (weapon)");
        }

        [Test]
```

```
        public void TestFullDescription()
        {
            Assert.IsTrue(_itemTest.FullDescription == "This is an Excalibur");
        }
    }
}
```

4. Player.cs

```
using System;
using Swin_Adventure;

namespace SwinAdventureTest
{
    [TestFixture]
    public class PlayerTest
    {
        private Player _playerTest;
        private Item _weaponTest;
        private Item _armorTest;

        [SetUp]
        public void Setup()
        {
            _playerTest = new Player("thuan", "dan choi");
            _weaponTest = new Item(new string[] { "weapon" }, "sword", "this is an
Excalibur");
            _armorTest = new Item(new string[] { "armor" }, "shield", "this is a shield");



            _playerTest.Inventory.Put(_weaponTest);
            _playerTest.Inventory.Put(_armorTest);
        }

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

        [Test]
        public void TestPlayerLocateItems()
        {
            Assert.IsTrue(_playerTest.Locate("weapon") == _weaponTest);
            Assert.IsTrue(_playerTest.Locate("armor") == _armorTest);
```

```
            Assert.IsTrue(_playerTest.Inventory. HasItem("weapon"));
            Assert.IsTrue(_playerTest.Inventory.HasItem("armor"));
        }

        [Test]
        public void TestPlayerLocateItself()
        {
            Assert.IsTrue(_playerTest == _playerTest.Locate("me"));
            Assert.IsTrue(_playerTest == _playerTest.Locate("inventory"));
        }

        [Test]
        public void TestPlayerLocateNothing()
        {
            Assert.IsTrue(_playerTest.Locate("helmet") == null);
        }

        [Test]
        public void TestPlayerFullDescription()
        {
            Assert.IsTrue(_playerTest.FullDescription == "You are thuan, dan choi.\nYou
are carrying:\n\ta sword (weapon)\n\ta shield (armor)\n");
        }
    }
}
```

5. BagTest.cs

```
using System;
namespace Swin_Adventure
{
            [TestFixture]
            public class BagTest
            {
                        private Bag _bagTest1;
        private Bag _bagTest2;
        private Item _weaponTest;
        private Item _armorTest;

        [SetUp]
        public void SetUp()
                    {
                                _bagTest1 = new Bag(new string[] { "bag1" }, "backpack", "It's
spacious");
            _bagTest2 = new Bag(new string[] { "bag2" }, "suitcase", "It's compact");
            _weaponTest = new Item(new string[] { "weapon" }, "sword", "this is an
Excalibur");
            _armorTest = new Item(new string[] { "armor" }, "shield", "this is a shield");
```

```
            _bagTest1.Inventory.Put(_bagTest2);
            _bagTest1.Inventory.Put(_weaponTest);
            _bagTest2.Inventory.Put(_armorTest);
        }

        [Test]
        public void TestBagLocatesItems()
        {
          Assert.AreSame(_weaponTest, _bagTest1.Locate("weapon"));
        }

        [Test]
        public void TestBagLocatesitself()
        {
          Assert.AreSame(_bagTest1, _bagTest1.Locate("bag1"));
        }

        [Test]
        public void TestBagLocatesnothing()
        {
          Assert.IsNull(_bagTest1.Locate("bag3"));
        }

        [Test]
        public void TestBagFullDescription()
        {
          Assert.AreEqual("In the backpack you can see:\n\ta suitcase (bag2)\n\ta sword
    (weapon)\n", _bagTest1.FullDescription);
        }

        [Test]
        public void TestBaginBag()
        {
           Assert.AreSame(_bagTest2, _bagTest1.Locate("bag2"));
           Assert.AreSame(_weaponTest, _bagTest1.Locate("weapon"));
           Assert.IsNull(_bagTest1.Locate("armor"));
        }
      }
    }
```

6. LookCommandTest.cs

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

namespace SwinAdventureTest
```

```
{
    [TestFixture]
    public class TestLookCommand
    {
        private LookCommand _lookCommandTest;
        private Player _playerTest;
        private Bag _bagTest;
        private Item _gemTest;

        [SetUp]
        public void Setup()
        {
            _lookCommandTest = new LookCommand();
            _playerTest = new Player("thuan", "dan choi");
            _bagTest = new Bag(new string[] { "duffelbag" }, "duffelbag", "it's small-sized");
            _gemTest = new Item(new string[] { "gem" }, "gem", "a beautiful gem");
        }

        [Test]
        public void TestLookAtMe()
        {
            Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
"at", "inventory" }), Is.EqualTo("You are thuan, dan choi.\nYou are carrying:\n"));
        }

        [Test]
        public void TestLookAtGem()
        {
            _playerTest.Inventory.Put(_gemTest);

            Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
"at", "gem" }), Is.EqualTo("a beautiful gem"));
        }

        [Test]
        public void TestLookAtUnk()
        {
            Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
"at", "unknown" }), Is.EqualTo("I can't find the unknown"));
        }

        [Test]
        public void TestLookAtGemInMe()
        {
            _playerTest.Inventory.Put(_gemTest);
```

```
            Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
"at", "gem", "in", "inventory" }), Is.EqualTo("a beautiful gem"));
        }

        [Test]
        public void TestLookAtGemInBag()
        {
            _bagTest.Inventory.Put(_gemTest);
            _playerTest.Inventory.Put(_bagTest);

            Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
"at", "gem", "in", "duffelbag" }), Is.EqualTo("a beautiful gem"));
        }

        [Test]
        public void TestLookAtGemInNoBag()
        {
            Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
"at", "gem", "in", "duffelbag" }), Is.EqualTo("I can't find the duffelbag"));
        }

        [Test]
        public void TestLookAtNoGemInBag()
        {
            _playerTest.Inventory.Put(_bagTest);

            Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
"at", "gem", "in", "duffelbag" }), Is.EqualTo("I can't find the gem"));
        }

        [Test]
        public void TestInvalidLook()
        {
            Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
"around" }), Is.EqualTo("I don't know how to look like that"));
            Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "hello" }),
Is.EqualTo("I don't know how to look like that"));
            Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
"at", "a", "at", "b" }), Is.EqualTo("What do you want to look in?"));
            Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "hello",
"at", "a" }), Is.EqualTo("Error in look input"));
            Assert.That(_lookCommandTest.Execute(_playerTest, new string[] { "look",
"by", "a" }), Is.EqualTo("What do you want to look at?"));
        }
    }
}
```

III. Image

1. NUnit test run