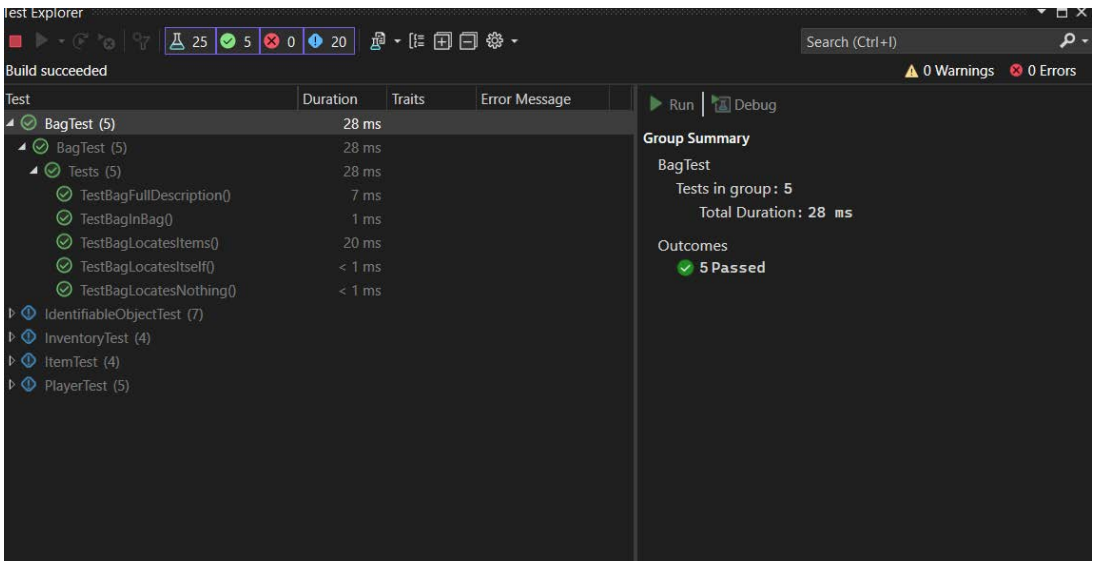
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace _5._2P
8  {
9      public class Bag : Item
10     {
11         private Inventory _inventory;
12
13         public Bag(string[] ids, string name, string description) : base
             (ids, name, description)
14         {
15             _inventory = new Inventory();
16         }
17
18         public GameObject Locate(string id)
19         {
20             if (AreYou(id))
21             {
22                 return this;
23             }
24             return _inventory.Fetch(id);
25         }
26
27         public string FullDescription
28         {
29             get
30             {
31                 return $"In the {Name} you can see: {string.Join(", ",
                     _inventory.ItemList)}"; //add "," between every
                     elements
32             }
33         }
34
35         public Inventory Inventory
36             { get { return _inventory; } }
37
38     }
39 }
40
```



```
1  using _5._2P;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Threading.Tasks;
6  using NUnit.Framework;
7  using System.Numerics;
8
9
10
11 namespace BagTest
12 {
13     [TestFixture]
14     public class Tests
15     {
16         private Item _item1;
17         private Item _item2;
18         private Bag _bag;
19
20         [SetUp]
21         public void Setup()
22         {
23             _item1 = new Item(new string[] { "sword" }, "Sword", "very
                 stronk sword");
24             _item2 = new Item(new string[] { "shield" }, "Shield", "very
                  stronk shield");
25             _bag = new Bag(new string[] { "backpack" }, "Backpack",
                 "very gud backpack");
26             _bag.Inventory.Put(_item1);
27             _bag.Inventory.Put(_item2);
28         }
29
30         [TestCase]
31         public void TestBagLocatesItems()
32         {
33             Assert.Pass();
34             var locatedItem = _bag.Locate("sword"); //reflect the return
                  type of the method
35             Assert.IsNotNull(locatedItem);
36             Assert.AreEqual(locatedItem, _item1);
37         }
38
39         [TestCase]
40         public void TestBagLocatesItself()
41         {
42             var locatedBag = _bag.Locate("backpack");
43             Assert.IsNotNull(locatedBag);
44             Assert.AreEqual(locatedBag, _bag);
45         }
46
47         [TestCase]
48         public void TestBagLocatesNothing()
49         {
```

```
50              var locatedItem = _bag.Locate("money");
51              Assert.IsNull(locatedItem);
52          }
53
54          [TestCase]
55          public void TestBagFullDescription()
56          {
57              string expectedDescription = "In the Backpack you can see:
                  \tSword (sword)\n\tShield (shield)\n";
58              Assert.AreEqual(expectedDescription, _bag.FullDescription);
59          }
60
61          [TestCase]
62          public void TestBagInBag()
63          {
64              Bag innerBag = new Bag(new string[] { "innerBag" }, "Inner
                  Bag", "smaller bad");
65              Item _item3 = new Item(new string[] { "diamond" },
                  "Diamond", "very rare diamond");
66              innerBag.Inventory.Put(_item3);
67              _bag.Inventory.Put(innerBag);
68
69              //Test outer bag can locate inner bag
70              var locatedInnerBag = _bag.Locate("innerBag");
71              Assert.IsNotNull(locatedInnerBag);
72              Assert.AreEqual(innerBag, locatedInnerBag);
73
74              //Test outer bag can locate its item
75              var locatedItemInOuterBag = _bag.Locate("sword");
76              Assert.IsNotNull(locatedItemInOuterBag);
77              Assert.AreEqual(_item1, locatedItemInOuterBag);
78
79              //Test outer bag cannot locate inner bag's item
80              var locatedItemInInnerBag = _bag.Locate("diamond");
81              Assert.IsNull(locatedItemInInnerBag);
82          }
83      }
84  }
```

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

namespace _4._2P
{
    public class GameObject : IdentifiableObject
    {
        private string _name;
        private string _description;

        public GameObject(string[] ids, string name, string description) : base(ids)    //call constructor of the base class
        {
            _name = name;
            _description = description;
        }

        public string Name
        {
            get { return _name; }
        }

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

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

}

```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace _4._2P
8  {
9      public class IdentifiableObject
10     {
11         private List<string> _identifiers = new List<string>();
12         public IdentifiableObject(string[] identifiers)
13         {
14
15
16             foreach (string id in identifiers)
17             {
18                 AddIdentifier(id);
19             }
20         }
21
22         public void AddIdentifier(string identifier)
23         {
24             _identifiers.Add(identifier.ToLower());
25         }
26
27         public bool AreYou(string identifier)
28         { return _identifiers.Contains(identifier.ToLower()); }
29
30         public string FirstId
31         {
32             get
33             {
34                 if (_identifiers.Count > 0)
35                 {
36                     return _identifiers[0];
37                 }
38                 else
39                 {
40                     return "";
41                 }
42             }
43         }
44
45
46
47         public void PrivilegeEscalation(string pin)
48         {
49             if (pin == "2183" && _identifiers.Count > 0)
50             {
51                 _identifiers[0] = "7";
52             }
53         }
```

```
54      }
55  }
56
```

```
1  using _4._2P;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Threading.Tasks;
6  using NUnit.Framework;
7  using System.Numerics;
8
9
10 namespace NUnitTests
11 {
12     [TestFixture]
13     public class TestIdentifiableObject
14     {
15         private IdentifiableObject _objectIdentifier;
16         private IdentifiableObject _emptyObjectIdentifier;
17         private IdentifiableObject _objectIdentifierNoStuId;
18
19         [SetUp]
20         public void SetUp()
21         {
22             _objectIdentifier = new IdentifiableObject(new string[]
                 { "104772183", "Khoi Nguyen", "Pham" });
23             _emptyObjectIdentifier = new IdentifiableObject(new string[]
                  { });
24             _objectIdentifierNoStuId = new IdentifiableObject(new string
                 [] { "1111", "Khoi Nguyen", "Pham" });
25         }
26
27         [TestCase]
28         public void TestAreYou()
29         {
30             Assert.IsTrue(_objectIdentifier.AreYou("104772183"));
31             Assert.IsTrue(_objectIdentifier.AreYou("Khoi Nguyen"));
32         }
33
34         [TestCase]
35         public void TestNotAreYou()
36         {
37             Assert.IsFalse(_objectIdentifier.AreYou("nonexistent
                 identifier"));
38         }
39
40
41
42         [TestCase]
43         public void TestFirstId()
44         {
45             Assert.AreEqual("104772183", _objectIdentifier.FirstId);
46         }
47
48         [TestCase]
49         public void TestFirstIdWithNoId()
```

```
50          {
51              Assert.AreEqual("", _emptyObjectIdentifier.FirstId);
52          }
53
54          [TestCase]
55          public void TestAddId()
56          {
57              _objectIdentifier.AddIdentifier("Nevan");
58              Assert.IsTrue(_objectIdentifier.AreYou("Nevan"));
59          }
60
61          [TestCase]
62          public void TestCaseSensitivity()
63          {
64              Assert.IsTrue(_objectIdentifier.AreYou("khOI NGUyEN"));
65              Assert.IsTrue(_objectIdentifier.AreYou("PhAM"));
66          }
67
68          [TestCase]
69          public void TestPrivilegeEscalation()
70          {
71
72
73              string correctPin = "2183";
74              string expectedFirstId = "7";
75
76
77              _objectIdentifierNoStuId.PrivilegeEscalation(correctPin);
78
79              // Assert
80              Assert.AreEqual(expectedFirstId,                              ↵
                  _objectIdentifierNoStuId.FirstId);
81          }
82      }
83
84
85
86
87  }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace _4._2P
8  {
9      public class Inventory
10     {
11         private List<Item> _items = new List<Item>();
12         public Inventory() { }
13         public bool HasItem (string id)
14         {
15             return Fetch(id) != null;
16         }
17         public void Put(Item itm)
18         {
19             _items.Add(itm);
20         }
21         public Item Take(string id)
22         {
23             Item item = Fetch(id);
24             if (item != null)
25             {
26                 _items.Remove(item);
27             }
28             return item;
29         }
30
31         public Item Fetch(string id)
32         {
33             foreach (Item item in _items)
34             {
35                 if (item.AreYou(id))
36                 {
37                     return item;
38                 }
39             }
40             return null;
41         }
42
43         public string ItemList
44         {
45             get
46             {
47                 string itemList = "";
48                 foreach (Item item in _items)
49                 {
50                     itemList += "\t" + item.ShortDescription + "\n";
51                 }
52                 return itemList;
53             }
```

```
54          }
55
56  }
57  }
58
```

```
1  using _4._2P;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Threading.Tasks;
6  using NUnit.Framework;
7  using System.Numerics;
8
9  namespace InventoryTest
10 {
11     [TestFixture]
12     public class Tests
13     {
14         private Inventory _inventory;
15         private Item _sword;
16         private Item _shield;
17
18         [SetUp]
19         public void Setup()
20         {
21             _inventory = new Inventory();
22             _sword = new Item(new string[] { "sword", "blade" },
                 "Excalibur", "A strong sword");
23             _shield = new Item(new string[] { "shield", "safeguard" },
                 "Aegis", "A strong shield");
24         }
25
26         [Test]
27         public void TestFindItem()
28         {
29             _inventory.Put(_sword);
30             Assert.IsTrue(_inventory.HasItem("sword"));
31         }
32
33         [Test]
34         public void TestNoItemFind()
35         {
36             Assert.IsFalse(_inventory.HasItem("spear"));
37         }
38
39         [Test]
40         public void TestFetchItem()
41         {
42             _inventory.Put(_sword);
43             Item fetchedItem = _inventory.Fetch("sword");
44             Assert.AreEqual(_sword, fetchedItem);
45             Assert.IsTrue(_inventory.HasItem("sword"));
46         }
47
48         [Test]
49         public void TestTakeItem()
50         {
51             _inventory.Put(_shield);
```

```
52              Item takenItem = _inventory.Take("shield");
53              Assert.AreEqual(takenItem, _shield);
54              Assert.IsFalse(_inventory.HasItem("shield"), "Inventory
                  should not contain the shield because it has been taken");
55
56          }
57
58          public void TestItemList()
59          {
60              _inventory.Put(_sword);
61              _inventory.Put(_shield);
62              string expectedList = "\tExcalibur\n\tAegis";
63              Assert.AreEqual(expectedList, _inventory.ItemList);
64          }
65      }
66  }
```

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using System.Text;
 5  using System.Threading.Tasks;
 6
 7  namespace _4._2P
 8  {
 9      public class Item : GameObject
10      {
11          public Item(string[] ids, string name, string description) :
              base(ids, name, description)
12          {
13
14          }
15      }
16  }
17
```

```
1  using _4._2P;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Threading.Tasks;
6  using NUnit.Framework;
7  using System.Numerics;
8
9  namespace ItemTest
10 {
11     [TestFixture]
12     public class Tests
13     {
14         private Item _sword;
15
16         [SetUp]
17         public void Setup()
18         {
19             _sword = new Item(new string[] { "sword", "blade" },
                 "Excalibur", "A strong sword");
20         }
21
22         [Test]
23         public void TestItemIsIdentifiable()
24         {
25             Assert.IsTrue(_sword.AreYou("sword"), "Item should be
                 identifiable as 'sword'");
26             Assert.IsTrue(_sword.AreYou("blade"), "Item should be
                 identifiable as 'blade'");
27         }
28
29         [Test]
30         public void TestShortDescription()
31         {
32             Assert.AreEqual("Excalibur (sword)",
                 _sword.ShortDescription);
33         }
34
35         [Test]
36         public void TestFullDescription()
37         {
38             Assert.AreEqual("A strong sword", _sword.FullDescription);
39         }
40
41         [Test]
42         public void TestPrivilegeEscalation()
43         {
44             string correctPin = "2183";
45             string expectedFirstId = "7";
46             _sword.PrivilegeEscalation(correctPin);
47             Assert.AreEqual(expectedFirstId, _sword.FirstId);
48
49         }
```

```
50      }
51  }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace _4._2P
8  {
9      public class Player : GameObject
10     {
11         private Inventory _inventory =  new Inventory();
12
13         public Player(string name, string description) : base (new
             string[] { "me", "inventory" }, name, description) { } //name
             and des gotten from GameObject
14         //help the class identify itself and its item, 3 batteries, 2
             from GO and 1 from IO
15         public GameObject Locate(string id)
16         {
17             if (AreYou(id))
18             {
19                 return this; //return then player object itself
20             }
21             return _inventory.Fetch(id);
22             //searches the inventory for an item with the given
                 identifier and returns it if found. If no item matches, it
                 returns null.
23         }
24
25         public override string FullDescription
26         {
27             get
28             {
29                 return $"You are {Name}, {base.FullDescription}\nYou are
                      carrying:\n{_inventory.ItemList}";
30             }
31         }
32
33         public Inventory Inventory { get { return _inventory; } }
34     }
35 }
36
```

```
1  using _4._2P;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Threading.Tasks;
6  using NUnit.Framework;
7  using System.Numerics;
8
9
10 namespace PlayerTest
11 {
12     [TestFixture] //mark a class that contains tests. It tells NUnit
         that this class should be treated as a test suite.
13     public class Tests
14     {
15         private Player _player;
16
17
18         [SetUp] //mark a method that should be run before each test
           method. It's useful for setting up common test data or state.
19         public void Setup()
20         {
21             _player = new Player("Nevan", "a human");
22         }
23
24         [Test] // mark a method inside a [TestFixture] class as a test
           method.
25         public void TestPlayerIsIdentifiable()
26         {
27             Assert.IsTrue(_player.AreYou("me"));
28             Assert.IsTrue(_player.AreYou("inventory"));
29         }
30
31         [Test]
32         public void TestPlayerLocatesItem()
33         {
34             Item sword = new Item(new string[] { "sword", "blade" },
               "Excalibur", "A strong sword");
35             _player.Inventory.Put(sword);
36             Assert.AreEqual(sword, _player.Locate("sword"));
37         }
38
39         [Test]
40         public void TestPlayerLocatesItself()
41         {
42             Assert.AreEqual(_player, _player.Locate("me"));
43         }
44
45         [Test]
46         public void TestLocatesNothing()
47         {
48             Assert.IsNull(_player.Locate("sth not exist"));
49         }
```

```
50
51          [Test]
52          public void TestPlayerFullDescription()
53          {
54              Item sword = new Item(new string[] { "sword", "blade" },
                  "Excalibur", "A strong sword");
55              _player.Inventory.Put(sword);
56              Assert.AreEqual("You are Nevan, a human\nYou are carrying:\n
                  \tExcalibur (sword)\n", _player.FullDescription);
57          }
58
59
60      }
61  }
```