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

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

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

        public string Name
        {
            get
            {
                return _name;
            }
        }

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

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

```

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

namespace IdentifiableObject
{
    public class IdentifiableObject
    {
        private List<string> _identifiers;

        public IdentifiableObject(string[] idents)
        {
            _identifiers = new List<string>();
            for(int i=0; i<idents.Length; i++)
            {
                _identifiers.Add(idents[i].ToLower());
            }
        }

        public bool AreYou(string id)
        {
            if(_identifiers.Contains(id.ToLower()))
            {
                return true;
            }
            else
            {
                return false;
            }
        }

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

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

        public void PrivilegeEscalation(string pin)
        {
            if(pin == "6473")
            {
                _identifiers[0] = "6473";
            }
        }
    }
}

```

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

namespace IdentifiableObject
{
    public class Inventory
    {
        private List<Item> _items;

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

        public bool HasItem(string id)
        {
            foreach (Item i in _items)
            {
                if (i.AreYou(id))
                {
                    return true;
                }
            }
            return false;
        }

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

        public Item Take(string id)
        {
            Item take_item = this.Fetch(id);
            _items.Remove(take_item);
            return take_item;
        }

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

        public string ItemList
        {
            get
            {
                string item_list = "";
                foreach (Item i in _items)
                {
                    item_list = item_list + i.ShortDescription + "\n";
                }
                return item_list;
            }
        }
    }
}

```

```csharp

namespace IdentifiableObject
{
    public class Tests
    {
        private Item _shield;
        private Item _shovel;
        private Inventory _my_invent;

        [SetUp]
        public void Setup()
        {
            _shield = new Item(new string[] { "shield" }, "a shield", "Shield level 1");
            _shovel = new Item(new string[] { "shovel" }, "a shovel", "Shovel level 2");
            _my_invent = new Inventory();
        }

        [Test]
        public void TestFindItem()
        {
            _my_invent.Put(_shield);
            Assert.IsTrue(_my_invent.HasItem(_shield.FirstId));
        }

        [Test]
        public void TestNoItemFind()
        {
            Assert.IsFalse(_my_invent.HasItem(_shield.FirstId));
        }

        [Test]
        public void TestFetchItem()
        {
            _my_invent.Put(_shovel);
            Item fetch_item = _my_invent.Fetch(_shovel.FirstId);

            Assert.AreEqual(fetch_item, _shovel);
            Assert.IsTrue(_my_invent.HasItem(_shovel.FirstId));
        }

        [Test]
        public void TestTakeItem()
        {
            _my_invent.Put(_shovel);
            _my_invent.Take(_shovel.FirstId);

            Assert.IsFalse(_my_invent.HasItem(_shovel.FirstId));
        }

        [Test]
        public void TestItemList()
        {
            _my_invent.Put(_shield);
            _my_invent.Put(_shovel);

            Assert.AreEqual(_my_invent.ItemList, "a shield (shield)\n" + "a shovel (shovel)\n");
        }
    }
}
```

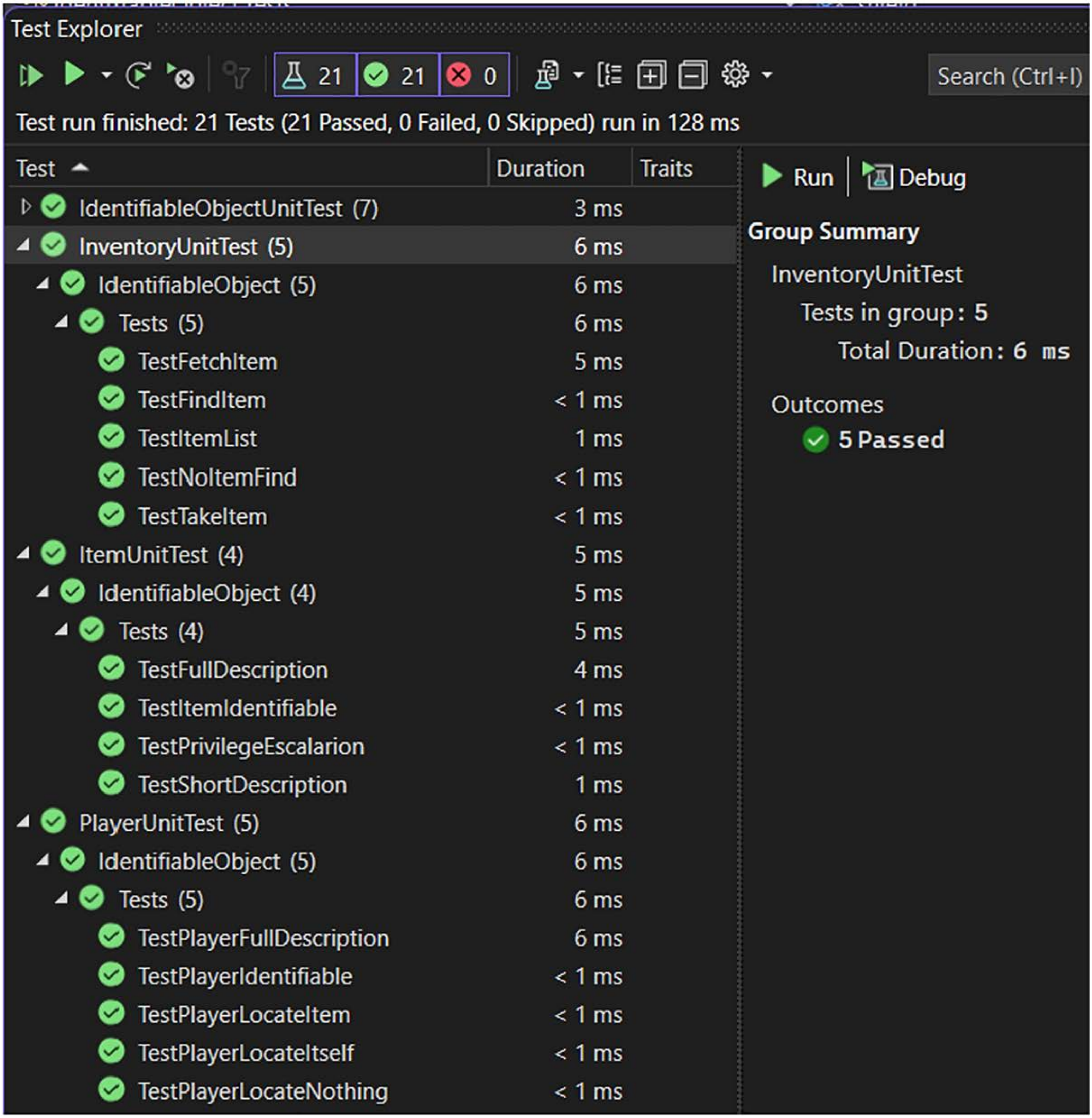
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace IdentifiableObject
{
    public class Item:GameObject
    {
        public Item(string[] idents, string name, string desc) :base(idents, name, desc)
        {
        }
    }
}
```

```csharp

namespace IdentifiableObject
{
    public class Tests
    {
        private Item _shield;
        private Item _shovel;

        [SetUp]
        public void Setup()
        {
            _shield = new Item(new string[] { "shield" }, "a shield", "Shield level 1");
            _shovel = new Item(new string[] { "shovel" }, "a shovel", "Shovel level 2");
        }

        [Test]
        public void TestItemIdentifiable()
        {
            Assert.IsTrue(_shield.AreYou("shield"));
        }

        [Test]
        public void TestShortDescription()
        {
            Assert.AreEqual(_shield.ShortDescription, "a shield (shield)");
        }

        [Test]
        public void TestFullDescription()
        {
            Assert.AreEqual(_shovel.FullDescription, "Shovel level 2");
        }

        [Test]
        public void TestPrivilegeEscalarion()
        {
            _shield.PrivilegeEscalation("6473");
            Assert.AreEqual(_shield.FirstId, "6473");
        }
    }
}
```

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

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

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

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

        }

        public override string FullDescription
        {
            get
            {
                return $"You are ({Name}), ({base.FullDescription}). You are carrying:\n{_inventory.ItemList}";
            }
        }

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

```

```

namespace IdentifiableObject
{
    public class Tests
    {
        private Item _shield;
        private Item _shovel;
        private Player _player;

        [SetUp]
        public void Setup()
        {
            _shield = new Item(new string[] { "shield" }, "a shield", "Shield level 1");
            _shovel = new Item(new string[] { "shovel" }, "a shovel", "Shovel level 2");
            _player = new Player("Duc Thang", "Student");
        }

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

        [Test]
        public void TestPlayerLocateItem()
        {
            bool test = false;
            _player.Inventory.Put(_shield);
            var located_itm = _player.Locate("shield");
            if(located_itm == _shield)
            {
                test = true;
            }
            Assert.IsTrue(test);
        }

        [Test]
        public void TestPlayerLocateItself()
        {
            bool test = false;
            var myself = _player.Locate("me");
            var invent = _player.Locate("inventory");
            if (myself == _player || invent==_player)
            {
                test = true;
            }
            Assert.IsTrue(test);
        }

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

        [Test]
        public void TestPlayerFullDescription()
        {
            _player.Inventory.Put(_shield);
            _player.Inventory.Put(_shovel);
            string output = "You are (Duc Thang), (Student). You are carrying:\n" + "a shield (shield)\n" + "a shovel (shovel)\n";
            Assert.AreEqual(_player.FullDescription, output);
        }
    }
}
```

```
namespace IdentifiableObject
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}

```



```
Test run finished: 21 Tests (21 Passed, 0 Failed, 0 Skipped) run in 128 ms

Test                                          Duration    Traits
▷ ✓ IdentifiableObjectUnitTest (7)                3 ms
◢ ✓ InventoryUnitTest (5)                         6 ms
   ◢ ✓ IdentifiableObject (5)                     6 ms
      ◢ ✓ Tests (5)                               6 ms
           ✓ TestFetchItem                        5 ms
           ✓ TestFindItem                       < 1 ms
           ✓ TestItemList                         1 ms
           ✓ TestNoItemFind                     < 1 ms
           ✓ TestTakeItem                       < 1 ms
◢ ✓ ItemUnitTest (4)                              5 ms
   ◢ ✓ IdentifiableObject (4)                     5 ms
      ◢ ✓ Tests (4)                               5 ms
           ✓ TestFullDescription                  4 ms
           ✓ TestItemIdentifiable               < 1 ms
           ✓ TestPrivilegeEscalarion            < 1 ms
           ✓ TestShortDescription                 1 ms
◢ ✓ PlayerUnitTest (5)                            6 ms
   ◢ ✓ IdentifiableObject (5)                     6 ms
      ◢ ✓ Tests (5)                               6 ms
           ✓ TestPlayerFullDescription            6 ms
           ✓ TestPlayerIdentifiable             < 1 ms
           ✓ TestPlayerLocateItem               < 1 ms
           ✓ TestPlayerLocateItself             < 1 ms
           ✓ TestPlayerLocateNothing            < 1 ms
```

```
Group Summary
  InventoryUnitTest
    Tests in group: 5
      Total Duration: 6 ms

  Outcomes
    ✓ 5 Passed
```