```
1  using System;
2
3  public abstract class GameObject : IdentifiableObject
4  {
5      private string _name;
6      private string _description;
7
8      public GameObject(string[] ids, string name, string desc) : base(ids)
9      {
10         _name = name;
11         _description = desc;
12     }
13
14     public string Name
15     {
16         get { return _name; }
17     }
18
19     public virtual string ShortDescription
20     {
21         get { return $"{Name} ({FirstId})"; }
22     }
23
24     public virtual string FullDescription
25     {
26         get { return _description; }
27     }
28 }
29
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4
5  public class IdentifiableObject
6  {
7      private List<string> _identifiers;
8
9      public IdentifiableObject(string[] idents)
10     {
11         _identifiers = new List<string>();
12         foreach (var id in idents)
13         {
14             AddIdentifier(id);
15         }
16     }
17
18     public bool AreYou(string id)
19     {
20         return _identifiers.Contains(id.ToLower());
21     }
22
23     public string FirstId
24     {
25         get
26         {
27             return _identifiers.FirstOrDefault() ?? string.Empty;
28         }
29     }
30
31     public void AddIdentifier(string id)
32     {
33         _identifiers.Add(id.ToLower());
34     }
35
36     public void PrivilegeEscalation(string pin)
37     {
38         if (pin == "2184")
39         {
40             _identifiers[0] = "1";
41         }
42     }
43 }
44
```

```
1  using System.Collections.Generic;
2  using System.Linq;
3
4  public class Inventory
5  {
6      private List<Item> _items;
7
8      public Inventory()
9      {
10         _items = new List<Item>();
11     }
12
13     public bool HasItem(string id)
14     {
15         foreach (var item in _items)
16         {
17             if (item.AreYou(id))
18             {
19                 return true; // Return true if the item matches the id
20             }
21         }
22         return false; // Return false if no match is found
23     }
24
25     public void Put(Item itm)
26     {
27         _items.Add(itm);
28     }
29
30     public Item Take(string id)
31     {
32         Item item = Fetch(id);
33         if (item != null)
34         {
35             _items.Remove(item);
36         }
37         return item;
38     }
39
40     public Item Fetch(string id)
41     {
42         foreach (var item in _items)
43         {
44             if (item.AreYou(id))
45             {
46                 return item; // Return the first item that matches the id
47             }
48         }
49         return null; // Return null if no match is found
```

```
50      }
51
52
53      public string ItemList
54      {
55          get
56          {
57              var itemDescriptions = new System.Text.StringBuilder();
58
59              foreach (var item in _items)
60              {
61                  itemDescriptions.AppendLine("\t" + item.ShortDescription);
62              }
63
64              return itemDescriptions.ToString().TrimEnd();
65          }
66      }
67  }
68
69
70
```

```
1  using NUnit.Framework;
2
3  [TestFixture]
4  public class InventoryTests
5  {
6      private Inventory _inventory;
7      private Item _sword;
8      private Item _shield;
9
10     [SetUp]
11     public void Setup()
12     {
13         _inventory = new Inventory();
14         _sword = new Item(new string[] { "sword" }, "a sword", "A sharp
             blade.");
15         _shield = new Item(new string[] { "shield" }, "a shield", "A strong
              shield.");
16         _inventory.Put(_sword);
17         _inventory.Put(_shield);
18     }
19
20     [Test]
21     public void TestInventoryHasItems()
22     {
23         Assert.IsTrue(_inventory.HasItem("sword"));
24         Assert.IsTrue(_inventory.HasItem("shield"));
25         Assert.IsFalse(_inventory.HasItem("gem"));
26     }
27
28     [Test]
29     public void TestInventoryFetchItem()
30     {
31         Assert.AreEqual(_sword, _inventory.Fetch("sword"));
32         Assert.AreEqual(_shield, _inventory.Fetch("shield"));
33     }
34
35     [Test]
36     public void TestInventoryTakeItem()
37     {
38         Assert.AreEqual(_sword, _inventory.Take("sword"));
39         Assert.IsFalse(_inventory.HasItem("sword"));
40     }
41
42     [Test]
43     public void TestInventoryItemList()
44     {
45         string expectedItemList = "\ta sword (sword)\n\ta shield (shield)";
46         Assert.AreEqual(expectedItemList, _inventory.ItemList);
47     }
```

```
48  }
49
```

```
1 public class Item : GameObject
2 {
3     public Item(string[] idents, string name, string desc) : base(idents,
        name, desc) { }
4 }
5
```

```
using NUnit.Framework;

[TestFixture]
public class ItemTests
{
    private Item _sword;
    private Item _shield;

    [SetUp]
    public void Setup()
    {
        _sword = new Item(new string[] { "sword", "blade" }, "a sword", "A sharp blade.");
        _shield = new Item(new string[] { "shield" }, "a shield", "A strong shield.");
    }

    [Test]
    public void TestItemIsIdentifiable()
    {
        Assert.IsTrue(_sword.AreYou("sword"));
        Assert.IsTrue(_sword.AreYou("blade"));
        Assert.IsFalse(_sword.AreYou("shield"));
    }

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

    [Test]
    public void TestFullDescription()
    {
        Assert.AreEqual("A sharp blade.", _sword.FullDescription);
    }
}

```

```
1  public class Player : GameObject
2  {
3      private Inventory _inventory;
4
5      public Player(string name, string desc) : base(new string[] { "me",
         "inventory" }, name, desc)
6      {
7          _inventory = new Inventory();
8      }
9
10     public Inventory Inventory
11     {
12         get { return _inventory; }
13     }
14
15     public override string FullDescription
16     {
17         get
18         {
19             return $"You are {Name}, {base.FullDescription}\nYou are
                 carrying:\n{_inventory.ItemList}";
20         }
21     }
22
23     public GameObject Locate(string id)
24     {
25         if (AreYou(id))
26         {
27             return this;
28         }
29
30         return _inventory.Fetch(id);
31     }
32 }
33
34
```

```
1  using NUnit.Framework;
2
3  [TestFixture]
4  public class PlayerTests
5  {
6      [Test]
7      public void TestPlayerIsIdentifiable()
8      {
9          Player player = new Player("Fred", "a mighty programmer");
10         Assert.IsTrue(player.AreYou("me"));
11         Assert.IsTrue(player.AreYou("inventory"));
12     }
13
14     [Test]
15     public void TestPlayerLocatesItems()
16     {
17         Player player = new Player("Fred", "a mighty programmer");
18         Item item = new Item(new string[] { "sword" }, "a bronze sword", "a ↪
               shiny sword");
19         player.Inventory.Put(item);
20         Assert.AreEqual(item, player.Locate("sword"));
21     }
22
23     [Test]
24     public void TestPlayerLocatesItself()
25     {
26         Player player = new Player("Fred", "a mighty programmer");
27         Assert.AreEqual(player, player.Locate("me"));
28         Assert.AreEqual(player, player.Locate("inventory"));
29     }
30
31     [Test]
32     public void TestPlayerLocatesNothing()
33     {
34         Player player = new Player("Fred", "a mighty programmer");
35         Assert.IsNull(player.Locate("nonexistent"));
36     }
37
38     [Test]
39     public void TestPlayerFullDescription()
40     {
41         Player player = new Player("Fred", "a mighty programmer");
42         Item item1 = new Item(new string[] { "shovel" }, "a shovel", "a     ↪
             mighty fine shovel");
43         Item item2 = new Item(new string[] { "sword" }, "a bronze sword",   ↪
             "a shiny sword");
44         player.Inventory.Put(item1);
45         player.Inventory.Put(item2);
46
```

```
47          string expectedDescription = "You are Fred, a mighty programmer
              \nYou are carrying:\n\ta shovel (shovel)\n\ta bronze sword
              (sword)";
48          Assert.AreEqual(expectedDescription, player.FullDescription);
49      }
50  }
51
```

```
1  namespace swinadventure
2  {
3      internal class Program
4      {
5          static void Main(string[] args)
6          {
7              Console.WriteLine("Hello, World!");
8              IdentifiableObject id = new IdentifiableObject(new string[]
                 { "007", "Ved", "Makhijani" });
9              Console.WriteLine(
10             id.AreYou("Ved"));
11         }
12     }
13 }
14
15
```

```
1  using NUnit.Framework;
2
3  [TestFixture]
4  public class IdentifiableObjectTests
5  {
6      private IdentifiableObject obj;
7
8      [SetUp]
9      public void Setup()
10     {
11         obj = new IdentifiableObject(new string[] { "184", "Ved",
             "Makhijani" });
12     }
13
14     [Test]
15     public void TestAreYou()
16     {
17         Assert.IsTrue(obj.AreYou("184"));
18         Assert.IsTrue(obj.AreYou("Ved"));
19         Assert.IsTrue(obj.AreYou("Makhijani"));
20     }
21
22     [Test]
23     public void TestNotAreYou()
24     {
25         Assert.IsFalse(obj.AreYou("481"));
26     }
27
28     [Test]
29     public void TestCaseSensitive()
30     {
31         Assert.IsTrue(obj.AreYou("Ved"));
32         Assert.IsTrue(obj.AreYou("MAKHIJANI"));
33     }
34
35     [Test]
36     public void TestFirstId()
37     {
38         Assert.AreEqual("184", obj.FirstId);
39     }
40
41     [Test]
42     public void TestFirstIdWithNoIDs()
43     {
44         obj = new IdentifiableObject(new string[] { });
45         Assert.AreEqual(string.Empty, obj.FirstId);
46     }
47
48     [Test]
```

```
49      public void TestAddId()
50      {
51          obj = new IdentifiableObject(new string[] { "Seekers", "Athol",
              "Keith", "Bruce" });
52          obj.AddIdentifier("Mary");
53          Assert.IsTrue(obj.AreYou("Mary"));
54      }
55
56      [Test]
57      public void TestPrivilegeEscalation()
58      {
59          obj = new IdentifiableObject(new string[] { "184", "Ved" });
60          obj.PrivilegeEscalation("2184");
61          Assert.AreEqual("1", obj.FirstId);
62      }
63  }
64
```