Pass Task 10 – The Spell Book

Related Learning Outcomes

ULO1 – Explain the OO Principles

This exercise demonstrated object encapsulation

ULO2 – Use OO Language and Library

Demonstrated class and constructor declaration, the use of conditional statements (e.g. “if”), and assigning values to parameters. Uses of List<> and index this[int i] by accessing library using system.Collection.Generic, use readonly field.

ULO3 – Design, Develop and Test using an IDE

The code was developed using Xamarin Studio to build and run the program, as well as integrated debugging features to step and inspect values.

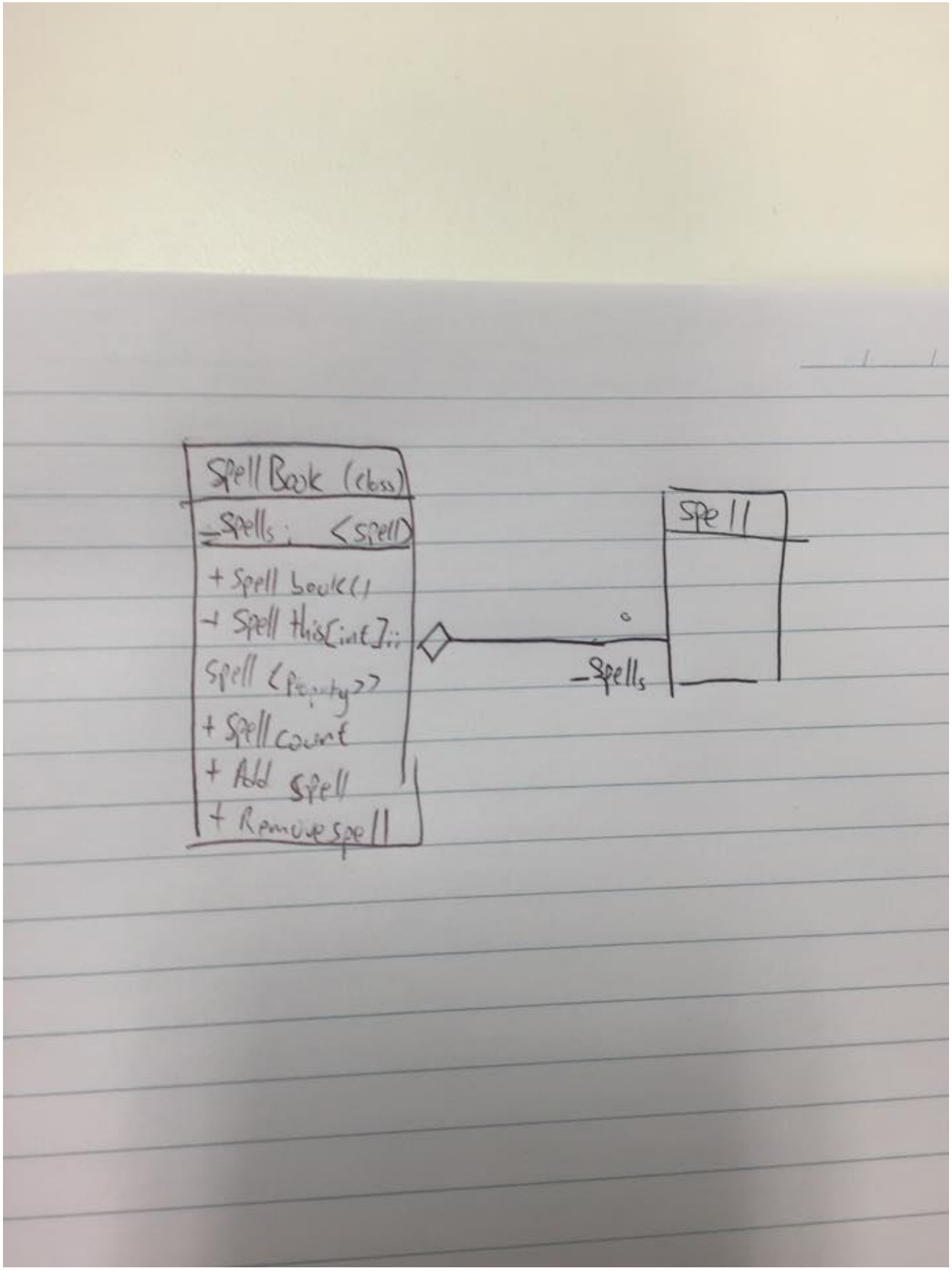
ULO4 – Communicate using UML Diagrams

I learned how to interpret a UML class diagram and write the related code.

ULO5 – Describe Elements of Good OO Design

The exercise demonstrated correct use of C# coding conventions.

Screenshots



[code running]

```
1  using System;
2  using System.Collections.Generic;
3
4  namespace SwinwardsSchoolMagic
5  {
6      public class Spellbook
7      {
8          private List<Spell> _spells;
9
10
11         public int SpellNumber
12         {
13             get
14             {
15                 return _spells.Count;
16             }
17         }
18
19         public Spell this [int i]
20         {
21             get
22             {
23                 return _spells [i];
24             }
25         }
26
27
28         public Spellbook ()
29         {
30             _spells = new List<Spell> ();
31         }
32
33         public void AddSpell (Spell s)
34         {
35             _spells.Add (s);
36         }
37
38         public void RemoveSpell (Spell s)
39         {
40             _spells.Remove (s);
41         }
42     }
43 }
44
```

No selection

```
using NUnit.Framework;
using System;

namespace SwinwardsSchoolMagic
{
    [TestFixture ()]
    public class TESTING
    {
        [Test ()]
        public void TestTeleport ()
        {
            Spell Spells = new Spell ("Mitch's mighty mover", SpellKind.Teleport);
            StringAssert.AreEqualIgnoringCase ("Poof... you appear somewhere else",  Spells.Castspell());
        }

        [Test ()]
        public void Healingtest ()
        {
            Spell Spells = new Spell ("Paul's potent poultice", SpellKind.Heal);
            StringAssert.AreEqualIgnoringCase ("Ahhh... you feel better",  Spells.Castspell());
        }

        [Test ()]
        public void INvisTest ()
        {
            Spell Spells = new Spell ("David's dashing disappearance", SpellKind.Invisibility);
            StringAssert.AreEqualIgnoringCase ("Zippp... where am I?",  Spells.Castspell());
        }

        [Test ()]
        public void Spellbook()
        {
            Spellbook newbook = new Spellbook ();

            Spell myTeleport = new Spell ("Mitch's mighty mover", SpellKind.Teleport);

            Spell myHeal = new Spell ("Paul's potent poultice", SpellKind.Heal);

            Spell myInvisibility = new Spell ("David's dashing disapperance", SpellKind.Invisibility);

            mybook.AddSpell (myTeleport);
            mybook.AddSpell (myHeal);
            mybook.AddSpell (myInvisibility);

            Assert.AreEqual (myTeleport, newbook [0]);
            Assert.AreEqual (myHeal, newbook [1]);
            Assert.AreEqual (myInvisibility, newbook [2]);

        }
    }
}

```