COS20007

Object-Orientated Programming

Learning Summary Report

VAISSHEENAVI PRABAKARAN
103508183

Self-Assessment Details

The following checklists provide an overview of my self-assessment for this unit.

	Pass (D)	Credit (C)	Distinction (B)	High Distinction (A)
Self-Assessment	/			

Self-Assessment Statement

	Included
Learning Summary Report	/
Test is Complete in Doubtfire	/
C# programs that demonstrate coverage of core concepts	/
Explanation of OO principles	/
All Pass Tasks are Complete on Doubtfire	/

Minimum Pass Checklist

	Included
All Credit Tasks are Complete on Doubtfire	

Minimum Credit Checklist (in addition to Pass Checklist)

	Included
Distinction tasks (other than Custom Program) are Complete	
Custom program meets Distinction criteria & Interview booked	
Design report has UML diagrams and screenshots of program	

Minimum Distinction Checklist (in addition to Credit Checklist)

	Included
HD Project included	
Custom project meets HD requirements	

Minimum High Distinction Checklist (in addition to Distinction Checklist)

Declaration

I declare that this portfolio is my individual work. I have not copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part of this submission been written for me by another person.

Signature: **P.Vaissheenavi**

Portfolio Overview

This portfolio includes work that demonstrates that I have achieve all Unit Learning Outcomes for COS20007 Object-Orientated Programming to a **Pass** level.

[Provide a justification for why you should receive this grade… Write this for the assessment panel – tell them why you should get this grade.

For Pass: you need to indicate how you have demonstrated all Unit Learning Outcomes to a minimal level.
For Credit: you need to indicate how you have demonstrated all Unit Learning Outcomes to a good level.
For Distinction: you need to indicate how you have been able to apply all of the Unit Learning Outcomes in achieving the distinction tasks.
For High Distinction: you need to indicate how you have been able to extend beyond the material presented in the unit.

In this section, refer to the tasks you have completed. These will be attached by Doubtfire after this summary. Do not try to demonstrate the outcomes here, this is just a summary.

Think of this like a cover letter to a job application – here it is a cover letter to your grade application.]

Justification

I have demonstrated all the Unit Learning Outcomes by having a solid understanding on the core principles of the object orientated programming paradigm which are encapsulation, abstraction, polymorphism, and inheritance. Some of the other contents that I learned in this unit includes the use of C# programming language, class libraries to develop OOP programs, framework classes to reuse the class to save time and doing NUnit tests to ensure there are no mistakes in the codes. Moreover, for some Pass tasks and the Week 8 Test, I learned how to properly construct a good UML diagram and textual descriptions that represents the program developed for the tasks accordingly. Along the way of this unit, I have designed, developed, tested, and debugged the tasks along with my tutor. My tutor gave me meaningful feedbacks which then really helped me to consolidate my understanding on some OOP good coding practices and concepts. Constantly working on the OOP tasks for 12 weeks strengthen my knowledge on the concepts and ‘when’/ ’how’ to use them efficiently. I like how this unit was designed as there were tasks created based on the core concepts. This helped me to apply the concepts that I learned over time from the online class/ tutorial session into the tasks assigned. On the side note, I have also referred to the unit online to demonstrate the learning outcomes. Now, regarding the grade, I should get a Pass because I have completed all the Pass tasks required. However, I am still trying my best on working with the credit tasks. I will upload what I have done for the credit tasks to Doubtfire before the deadline (5th June 2022) as you could assess my progress in the credit tasks too. Overall, this is an interesting unit and I can’t wait to implement what I have

learned from this unit in the future.

Reflection

The most important things I learnt:

The most important things I learnt is that encapsulation, abstraction, polymorphism, and inheritance. Encapsulation is the ability of an object to hide its data when it is not necessary to the users. Inheritance gives us an ability to create a new class from an existing class. Abstraction is used to hide the implementation information and only showing the users the required function. This is to hide unwanted information from the users and only shows the relevant information. Polymorphism refers to the idea of being able to access objects of many types using the same interface. This interface can be implements in a variety of ways by each type.

Referencing: https://www.programiz.com/csharp-programming/inheritance#:~:text=In%20C%23%2C%20inheritance%20allows%20us,derived%20class%20(child%20or%20subclass)

https://www.techopedia.com/definition/3787/encapsulation-c#:~:text=Encapsulation%2C%20in%20the%20context%20of,a%20single%20unit%20or%20object.

The things that helped me most were:

The things that helped me the most were the helpdesk sessions and my tutor. For example, when I had trouble understanding the tasks, I would often go to the helpdesks to ask for further clarifications from different tutors. All the tutors were so helpful, and they also ensured that I have completely understood the tasks before doing. Besides, the lecture recordings and materials were also helpful along the way.

I found the following topics particularly challenging:

At first, I found polymorphism was challenging for me grasp the concept of it. I managed to overcome it by doing the tasks. Besides, the NUnit Tests were also challenging for me but with the help of my tutor and helpdesk sessions, I managed to learn a lot from them.

I found the following topics particularly interesting:

I found inheritance interesting as we can reuse the codes for different classes by just inheriting the properties.

I feel I learnt these topics, concepts, and/or tools really well:

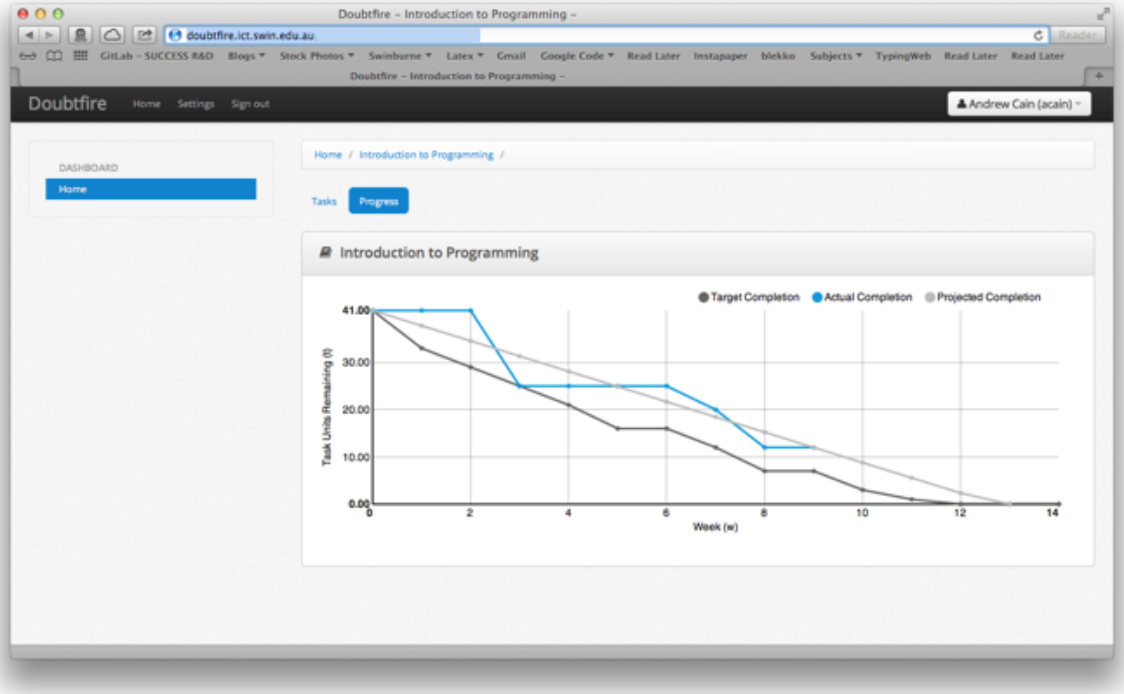
I feel I learned encapsulation and inheritance well. I perfectly understand how both encapsulation and inheritance. Moreover, I am able to confidently code the parts where this is required because I can understand it from the requirements given.

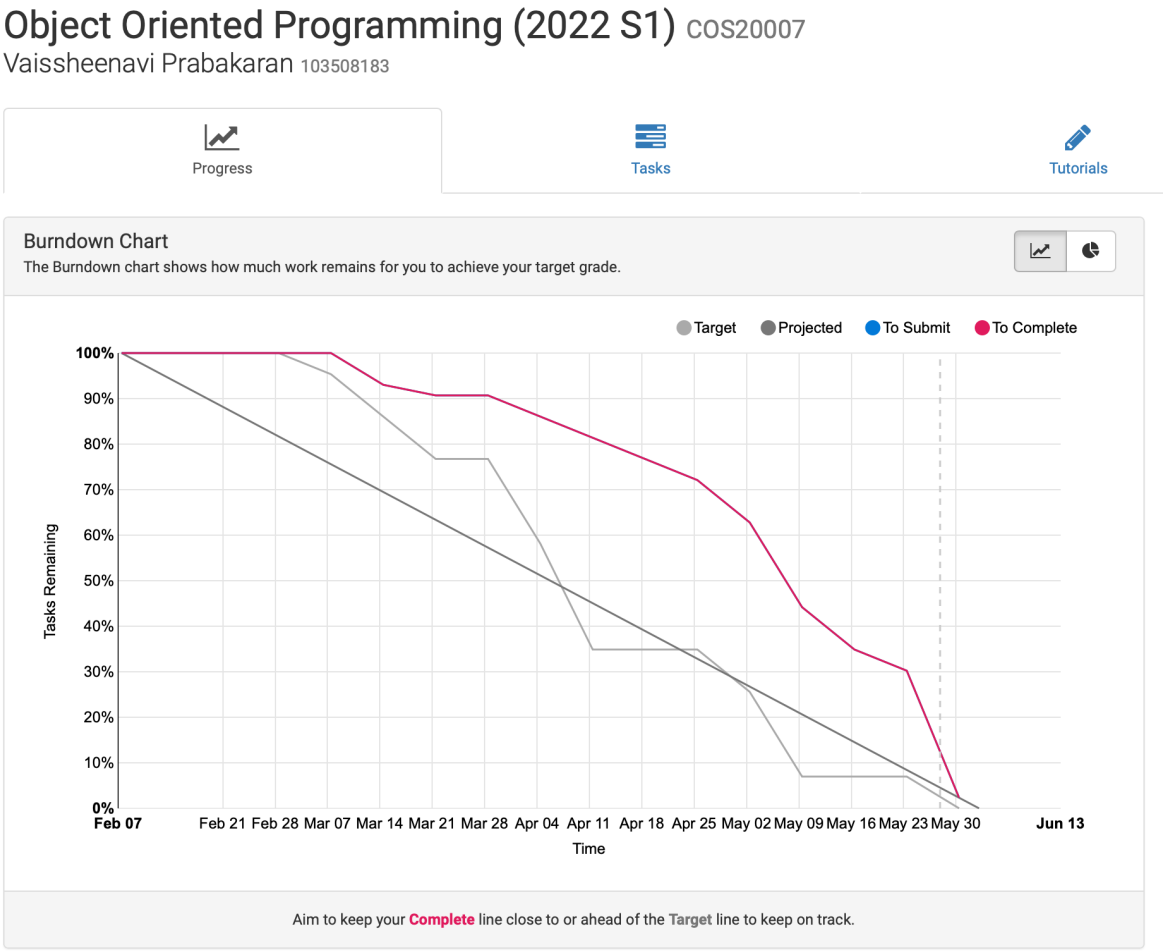
I still need to work on the following areas:

I still need to work on polymorphism as I still need time to wrap my head around this concept. I know how it works but I would like to be more confident while applying it.

My progress in this unit was …:

[Include a screenshot of your **progress graph** from **Doubtfire**, and comment on what happened from your perspective… what does the graph say about how you approached the unit? (Login to Doubtfire to get your graph https://doubtfire.ict.swin.edu.au)]





This unit will help me in the future:

The importance of encapsulation, abstraction, polymorphism, inheritance and all the topics that I have learnt in this unit, I believe that it will be valuable in the future. This is because this unit has successfully provided me with strong basics of object orientated programming and I am sure that I will be able to put all the knowledge I have learnt from this unit to a good use in the future.

If I did this unit again I would do the following things differently:

If I did this unit again, I would learn C# prior to learning this unit as most of the tasks (all Pass tasks) is in C#.

Other…:

If I have learnt C# early through the Internet, it would have been an easier process of understanding the tasks assigned. Fortunately, as the times goes, I got the hang of it and hopefully, I can do better next time.

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Vaissheenavi Prabakaran

Portfolio Submission

Submitted By:
Vaissheenavi Prabakaran
103508183

Tutor:
Jai Cornes

June 3, 2022

Contents

2 Overall Task Status

Task	Status	Times Assessed
Task 1.1P: Preparing for Object-Oriented Programming	Complete	1
Task 1.2P: Object Oriented Hello World	Complete	1
Task 1.3P: C# Language Reference Sheet	Complete	1
Task 2.1P: Counter Class	Complete	1
Task 2.2P: Drawing Program - A Basic Shape	Complete	1
Task 2.3P: Case Study Iteration 1 - Identifiable Object	Complete	1
Task 3.1P: The Stack and Heap	Complete	1
Task 3.2P: Drawing Program - A Drawing Class	Complete	1
Task 3.3P: Clock Class	Complete	1
Task 4.1P: Drawing Program: Multiple Shape Kinds	Complete	2
Task 4.2P: Case Study Iteration 2: Player Class and Inventory	Complete	1
Task 5.1P: Case Study Iteration 3: Bags	Complete	1
Task 5.2P: Case Study Iteration 4: Look Command	Complete	1
Task 6.1P: Case Study Iteration 5: Tying it Together	Complete	2
Task 6.2C: Case Study Iteration 6: Locations	Ready to Mark	1
Task 6.3D: D Level Custom Program Design	Not Started	
Task 6.4D: D Level Custom Program	Not Started	
Task 6.5HD: HD Level Custom Program Design	Not Started	
Task 6.6HD: HD Level Custom Program	Not Started	
Task 7.1P: Key Object Oriented Concepts	Complete	1
Task 7.2C: Case Study Iteration 7: Paths	Not Started	
Task 7.3C: Case Study Iteration 8: Command Processor	Not Started	
Task 8.1P: Semester TEST (Time-limited Evaluation of Skills Task)	Complete	1
Task 9.1HD: Research Project Plan	Not Started	
Task 9.2HD: Research Project	Not Started	
Task 11.1P: Clock in Another Language	Complete	2
Task 11.2P: Learning Summary Report Draft	Ready to Mark	1

3 Learning Outcomes

4 Task 1.1P: Preparing for Object-Oriented Programming

Coming into this unit we expect you to have some knowledge of programming (that's why there are pre-requisite units). This task will cover the core concepts we expect you to be familiar with.

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 1.1P: Preparing for Object-Oriented Programming

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/03/11 08:54

Tutor:
Jai Cornes

March 11, 2022

1.1P: Preparing for OOP – Answer Sheet

1. Explain the following terminal instructions:
    a. cd: To change the directory path, we use cd
    b. ls: ls is the list of content available in the directory
    c. pwd: pwd is the current directory that the computer is in

2. Consider the following kinds of information, and suggest the most appropriate data type to store or represent each:

Information	**Suggested Data Type**
A person's name	String
A person's age in years	Integer
A phone number	Integer
A temperature in Celsius	Float
The average age of a group of people	Integer/Float
Whether a person has eaten lunch	Boolean

3. Aside from the examples already given, come up with an example of information that could be stored as:

Data type	**Suggested Information**
String	A person's favorite food
Integer	A person's bank account number
Float	A person's height
Boolean	Whether a person has done their homework

4. Fill out the following table, evaluating the value of each expression and identifying the data type the value is most likely to be:

Expression	**Given**	**Value**	**Data Type**
5	5	5	Integer

`True`	`True`	True	Boolean
`a`	`a = 2.5`	2.5	Float
`1 + 2 * 3`	`1 + 2 * 3`	7	Integer
`a and False`	`a = True`	False	Boolean
`a or False`	`a = True`	True	Boolean
`a + b`	`a = 1` `b = 2`	3	Integer
`2 * a`	`a = 3`	6	Integer
`a * 2 + b`	`a = 1.5` `b = 2`	5	Integer
`a + 2 * b`	`a = 1.5` `b = 2`	5.5	Float

`(a + b) * c`	`a = 1`
	`b = 1`
	`c = 5`
		10	Integer

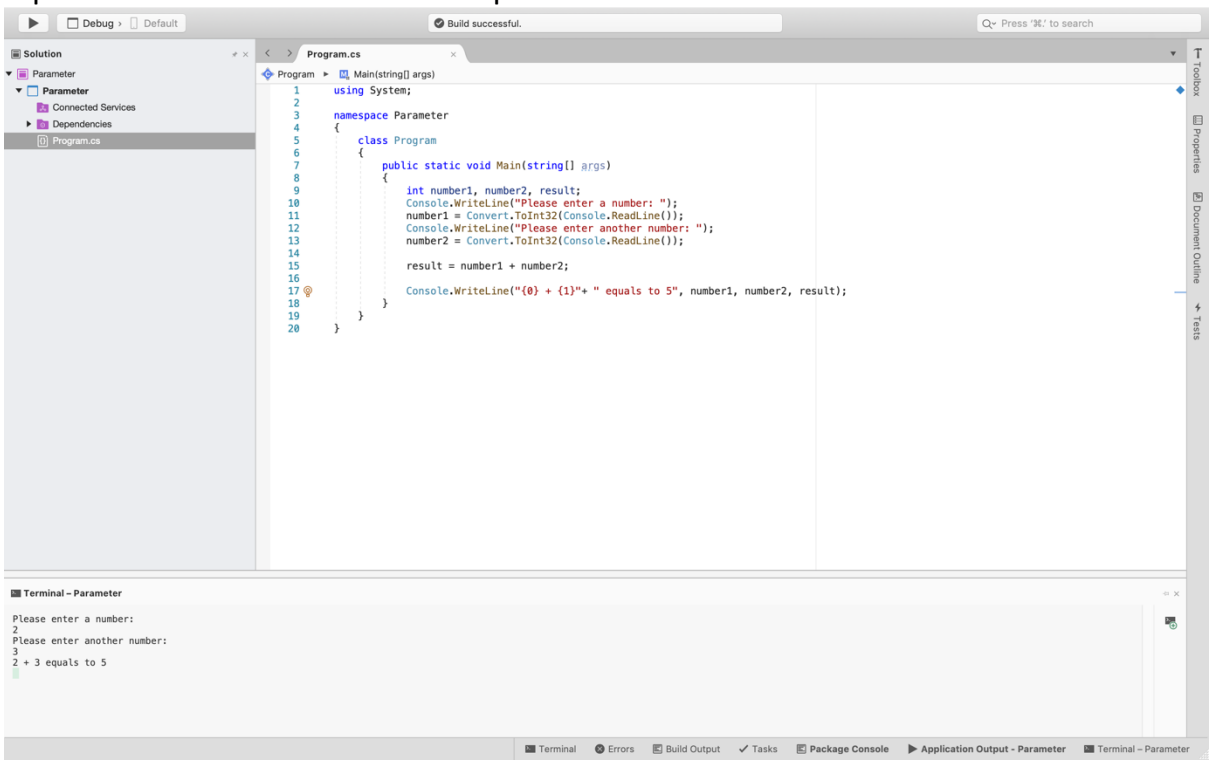
`"Fred" + " Smith"`	`"Fred" + " Smith"`	Fred Smith	String
`a + " Smith"`	`a = "Wilma"`	Wilma Smith	String

5. Explain the difference between **declaring** and **initialising** a variable.

The difference between the two is declaring means setting up and initialising means placing the value. For instance, if we say “declaring a variable”, it means that we are creating a variable while if we say “initialising a variable”, it means that we are putting the value in the variable.

6. Explain the term **parameter**. Write some code that demonstrates a simple of use of a parameter.

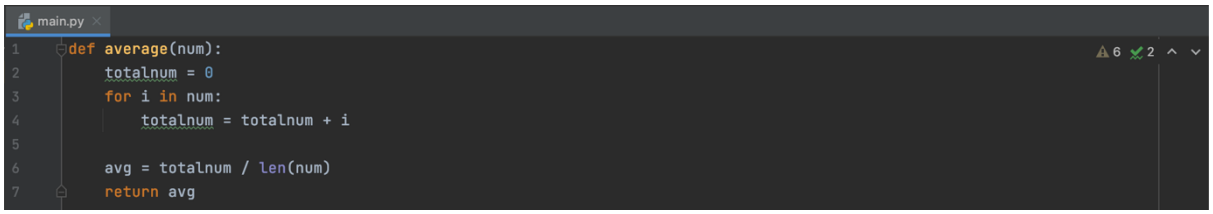
A parameter is a value that will be passed into the function.



7. Using an example, describe the term **scope**.

Scope defines how and where variables can be used and accessed.

8. In any procedural language you like, write a function called Average, which accepts an array of integers and returns the average of those integers.



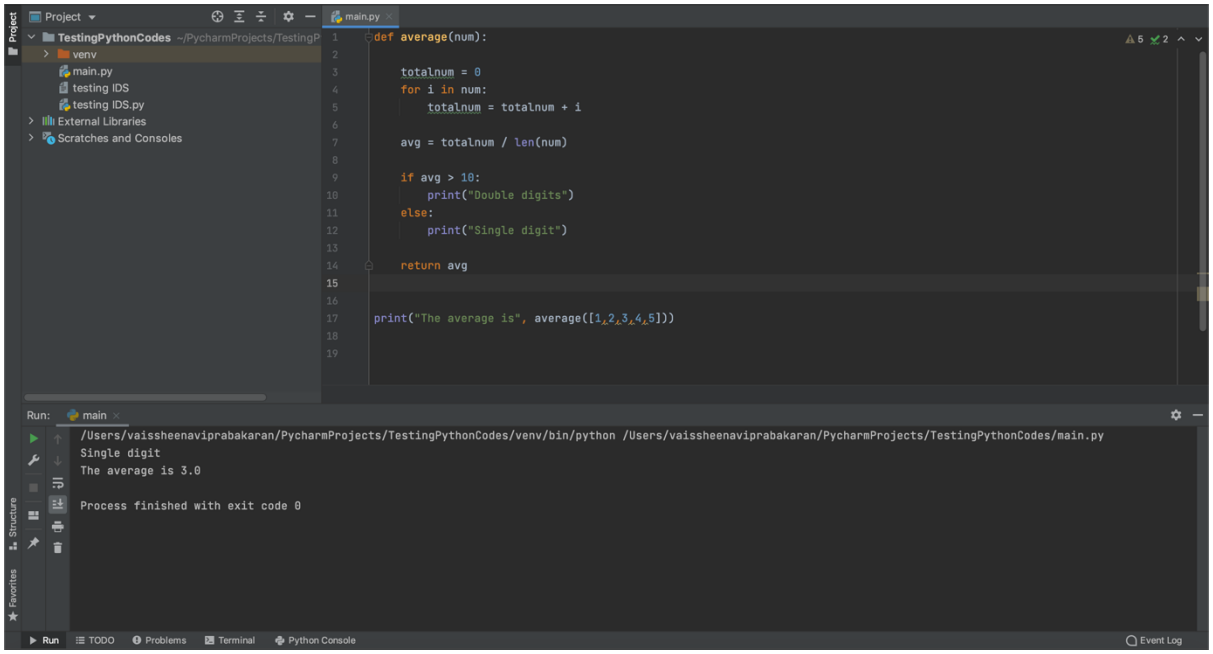
9. In the same language, write the code you would need to call that function and print out the result.

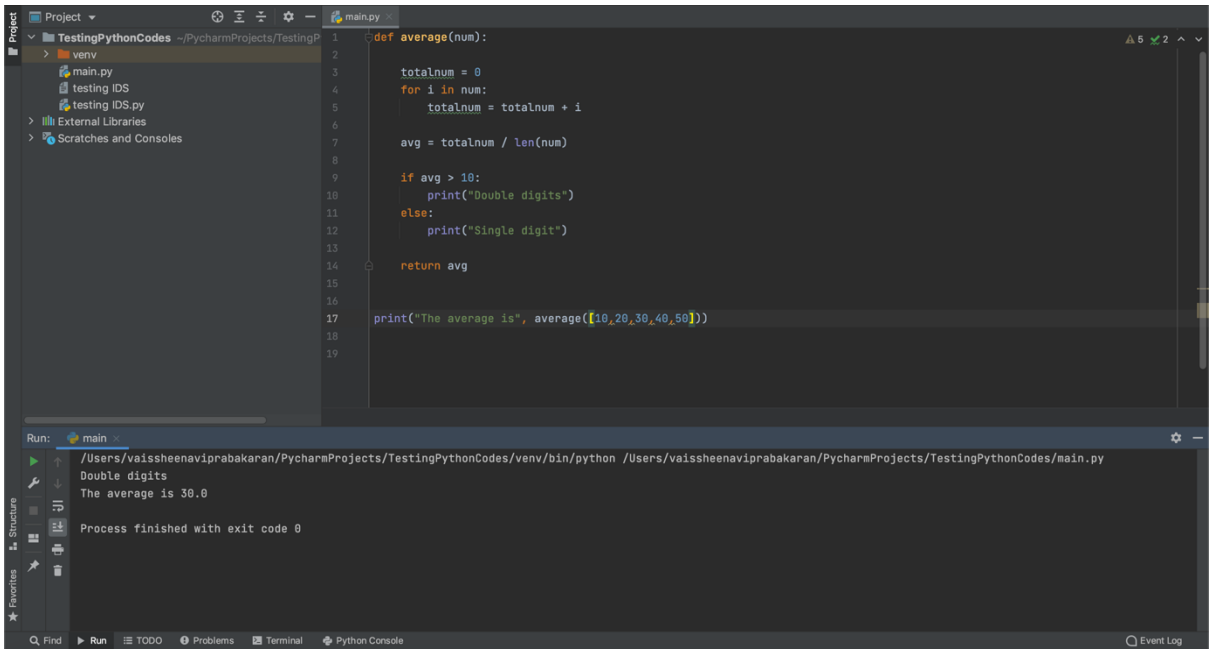
main.py

```
def average(num):
    totalnum = 0
    for i in num:
        totalnum = totalnum + i

    avg = totalnum / len(num)
    return avg

print("The average is", average([10,20,30,40,50]))
```

10. To the code from 9, add code to print the message “Double digits” if the average is above 10. Otherwise, print the message “Single digits”.

```python
def average(num):

    totalnum = 0
    for i in num:
        totalnum = totalnum + i

    avg = totalnum / len(num)

    if avg > 10:
        print("Double digits")
    else:
        print("Single digit")

    return avg


print("The average is", average([1,2,3,4,5]))
```




Project
TestingPythonCodes ~/PycharmProjects/TestingP
venv
main.py
testing IDS
testing IDS.py
External Libraries
Scratches and Consoles
main.py
def average(num):
totalnum = 0
for i in num:
totalnum = totalnum + i
avg = totalnum / len(num)
if avg > 10:
print("Double digits")
else:
print("Single digit")
return avg
print("The average is", average([10,20,30,40,50]))
Run:
main
/Users/vaissheenaviprabakaran/PycharmProjects/TestingPythonCodes/venv/bin/python /Users/vaissheenaviprabakaran/PycharmProjects/TestingPythonCodes/main.py
Double digits
The average is 30.0
Process finished with exit code 0
Find
Run
TODO
Problems
Terminal
Python Console
Event Log
Structure
Favorites

5 Task 1.2P: Object Oriented Hello World

As always, ”Hello World” is the first program you should write in a new language or with a new set of tools. In this task you will create an object oriented version of this classic program.

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 1.2P: Object Oriented Hello World

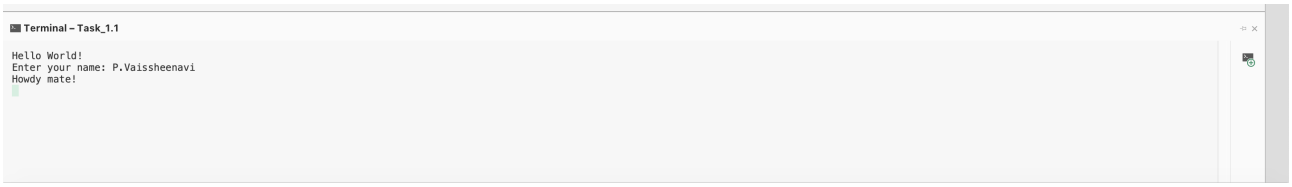
Submitted By:
Vaissheenavi Prabakaran
103508183
2022/03/11 09:09

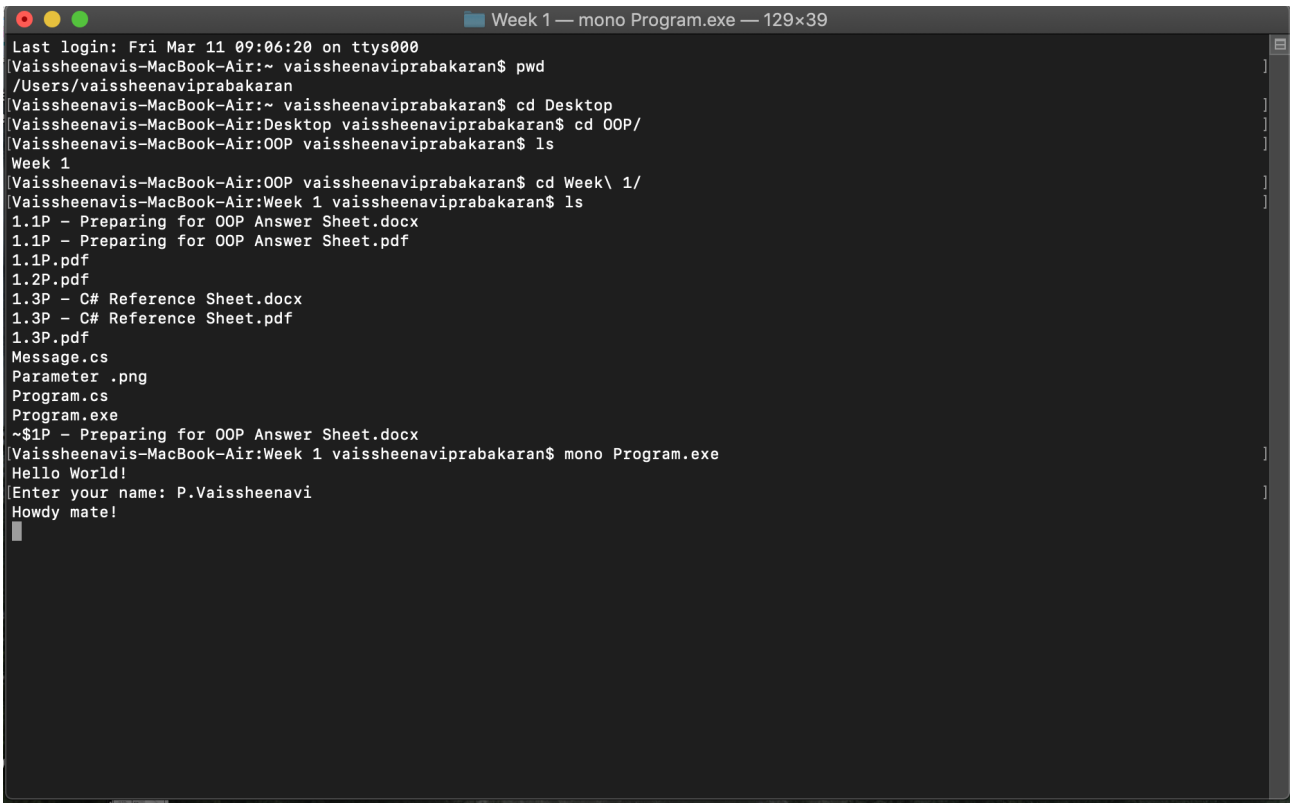
Tutor:
Jai Cornes

March 11, 2022

```
using System;

namespace Task_1._1
{
    class MainClass
    {
        public static void Main(string[] args)
        {

            Message myMessage;
            myMessage = new Message("Hello World!");
            //myMessage = new Message("Hello World - from Message Object");
            myMessage.Print();


            Message [] messageforname = new Message[4];
            messageforname[0] = new Message("Howdy mate!");
            messageforname[1] = new Message("That's a beautiful name");
            messageforname[2] = new Message("Great name");
            messageforname[3] = new Message("That's a silly name!");

            Console.Write("Enter your name: ");
            string name = Console.ReadLine();


            if (name.ToLower() == "p.vaissheenavi")
            {
                messageforname[0].Print();
            }

            else if (name.ToLower() == "p.jaihsreea")
            {
                messageforname[1].Print();
            }
            else if (name.ToLower() == "p.vajgenthra")
            {
                messageforname[2].Print();
            }

            else
            {
                messageforname[3].Print();
            }

                Console.ReadLine();
        }
    }
}
```

```
using System;
namespace Task_1._1
{
    public class Message
    {
        private string text;
        public Message(string txt)
        {
            text = txt;
        }

        public void Print()
        {
            Console.WriteLine(text);
        }
    }
}
```


Terminal – Task_1.1
Hello World!
Enter your name: P.Vaissheenavi
Howdy mate!


Week 1 — mono Program.exe — 129×39
Last login: Fri Mar 11 09:06:20 on ttys000
Vaissheenavis-MacBook-Air:~ vaissheenaviprabakaran$ pwd
/Users/vaissheenaviprabakaran
Vaissheenavis-MacBook-Air:~ vaissheenaviprabakaran$ cd Desktop
Vaissheenavis-MacBook-Air:Desktop vaissheenaviprabakaran$ cd OOP/
Vaissheenavis-MacBook-Air:OOP vaissheenaviprabakaran$ ls
Week 1
Vaissheenavis-MacBook-Air:OOP vaissheenaviprabakaran$ cd Week\ 1/
Vaissheenavis-MacBook-Air:Week 1 vaissheenaviprabakaran$ ls
1.1P - Preparing for OOP Answer Sheet.docx
1.1P - Preparing for OOP Answer Sheet.pdf
1.1P.pdf
1.2P.pdf
1.3P - C# Reference Sheet.docx
1.3P - C# Reference Sheet.pdf
1.3P.pdf
Message.cs
Parameter .png
Program.cs
Program.exe
~$1P - Preparing for OOP Answer Sheet.docx
Vaissheenavis-MacBook-Air:Week 1 vaissheenaviprabakaran$ mono Program.exe
Hello World!
Enter your name: P.Vaissheenavi
Howdy mate!

6 Task 1.3P: C# Language Reference Sheet

The key to any new language is getting familiar with its basic syntax. So let's do that!

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 1.3P: C# Language Reference Sheet

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/03/11 08:55

Tutor:
Jai Cornes

March 11, 2022

C# Programming Reference Sheet

Built In Data Types & Literals

Integers
```
Int (5)
```
Floating Point Numbers
```
float (2.4)
```
Strings and Characters
```
string ("good") char ('g')
```
Boolean
```
bool(True/False)
```

Working with Strings

Assignment (giving a string a value)
```
string favfood = "Subway"
```
Concatenation (joining strings)
```
Concatenation= "My fav food is " + favfood ;
```
Comparison
```
if (favfood == "Subway") {}
```
Construction from other types:
```
int inNum = 5;
string stNum = inNum.ToString();
```

Simple Programming Statements

Constant declaration
```
const int Pi = 3.14;
```
Variable declaration
```
string favfood; int quantity;
```
Assignment
```
favfood = "Subway" ; quantity = 1;
```
Method call
```
Console.WriteLine(string);
```
Sequence of statements - grouped
```
{}
```

Structured Programming Statements

If statement
```
if (age > 20) {}
```
Case statement
```
switch(expression){case x:b; case y:
b;default: b;} where b= break
```
While loop
```
while (age > 20) {}
```
Repeat loop
```
do {} while (age > 20)
```
For loop
```
for (statement 1;statement 2;statement 3) {}
```

Declaring Methods

Declare a method with parameters:
```
void Method (){}
```
Declare a method that returns data:
```
string Method(){return "Hello";}
```
Pass by reference:
```
int x = 14;
multiplication(ref x);
```

Boolean Operators and Other Statements

Comparison: equal, less, larger, not equal, less eq
```
==,  < ,  > ,  != ,  <=
```
Boolean: And, Or and Not
```
&& ,|| , !
```
Skip an iteration of a loop
```
continue;
```
End a loop early
```
break;
```
End a method:
```
return;
```

Custom Types

Classes
```
public class Message
{
  private string text;
  public Message(string txt){}
```
Enumerations
```
enum Speed{Fast, Moderate, Slow}…
Speed enumVar = Speed.Average;
```
Structs
```
Public struct type {
        Public type x;}
```

Arrays

Declaration
```
string[] bags = new string [size];
```
Access
```
string[] bags= {"Prada", "LV", "Gucci"}
Console.WriteLine(bags [0]); Output: Prada
```
Loop with index i
```
for( int i = 0; i < bags.Length; i +=1){}
```
For each loop
```
foreach (string i in bags){}
```

Programs and Modules

Creating a program
```
Static void Main(String[] args){}
```
Using a class from a library
```
Message myMessage;
myMessage = new Message(args);
```

Other Things

Reading from Terminal
```
Console.ReadLine();
```
Writing to Terminal
```
Console.WriteLine();
```
Comments
```
//Not parsed
```

7 Task 2.1P: Counter Class

Object oriented programming and design is probably feeling pretty new still. In this task you'll get some more practice and both by writing and designing a simple Counter class.

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 2.1P: Counter Class

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/03/19 21:44

Tutor:
Jai Cornes

March 19, 2022

```
using System;

namespace Task_2._1
{
    class Program
    {
        static void Main(string[] args)
        {
            Counter[] mycounters = new Counter [3];

            mycounters[0] = new Counter("Counter 1");
            mycounters[1] = new Counter("Counter 2");
            mycounters[2] = mycounters[0];

            for(int i = 0; i < 4; i++)
            {
                mycounters[0].Increment();
            }

            for (int i = 0; i < 9; i++)
            {
                mycounters[1].Increment();
            }

            PrintCounters(mycounters);

            mycounters[2].Reset();

            PrintCounters(mycounters);
        }

        static void PrintCounters(Counter[] counters)
        {
            foreach (Counter c in counters)
            {
                Console.WriteLine("{0} is {1}", c.Name, c.Ticks);
            }
        }

    }
}
```

```
using System;
namespace Task_2._1
{
    public class Counter
    {
        private int _count;
        private string _name;

        public string Name
        {
            get
            {
                return _name;
            }
            set
            {
                _name = value;
            }
        }

        public Counter(string name)
        {
            _name = name;
            _count = 0;
        }

        public void Increment()
        {
            _count++;
        }

        public void Reset()
        {
            _count = 0;

        }


        public int Ticks
        {
            get
            {
                return _count;
            }

        }
    }
}
```

Terminal – Task_2.1

```
Counter 1 is 4
Counter 2 is 9
Counter 1 is 4
Counter 1 is 0
Counter 2 is 9
Counter 1 is 0
```

8 Task 2.2P: Drawing Program - A Basic Shape

Let's get graphical! Drawing programs have a natural affinity with object oriented designs. In this task we'll get started with the basis of our own drawing program.

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 2.2P: Drawing Program - A Basic Shape

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/04/12 14:14

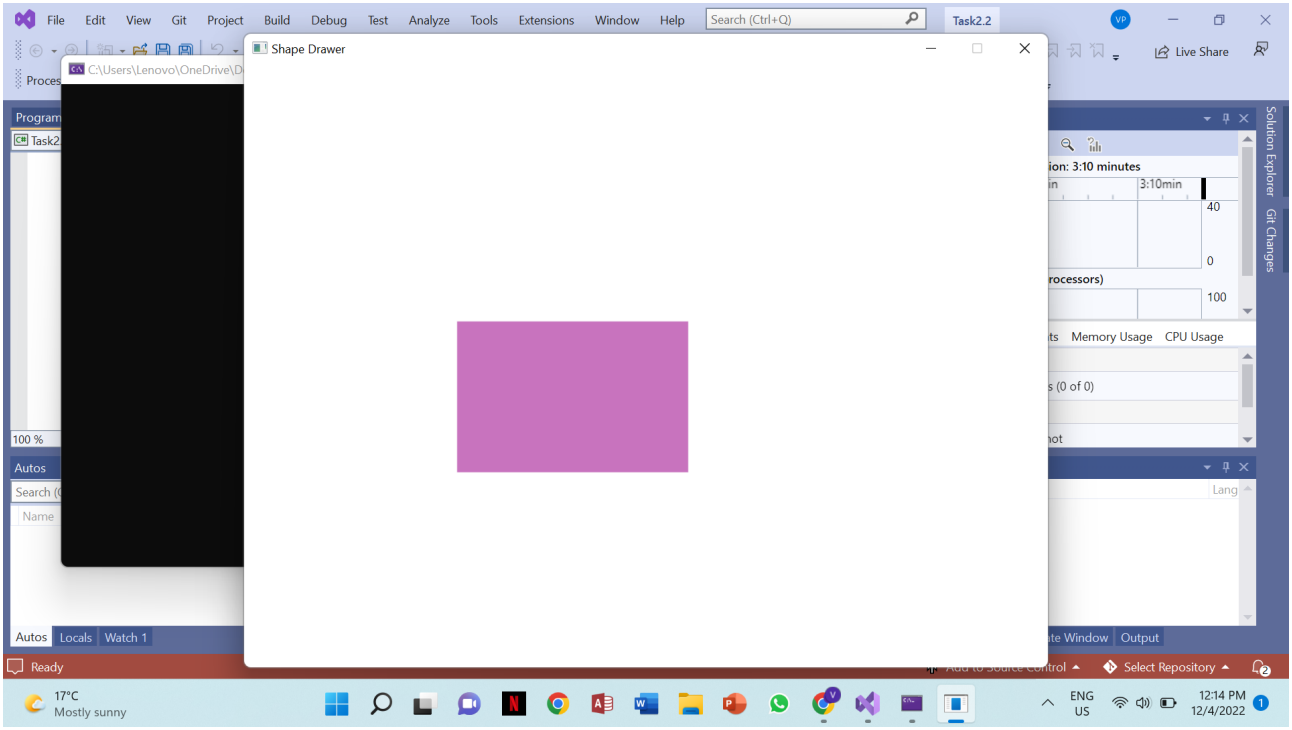
Tutor:
Jai Cornes

April 12, 2022

```
using System;
using SplashKitSDK;
using Task2._2;

namespace ShapeDrawer
{
    public class Program
    {
        public static void Main()
        {

            new Window("Shape Drawer", 800, 600);
            Shape myShape = new Shape();

            do
            {
                SplashKit.ProcessEvents();
                SplashKit.ClearScreen();

                myShape.Draw();


                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    myShape.X = SplashKit.MouseX();
                    myShape.Y = SplashKit.MouseY();
                }

                if (myShape.IsAt(SplashKit.MousePosition()))

                {
                    if (SplashKit.KeyTyped(KeyCode.SpaceKey))
                    {
                        myShape.Color= SplashKit.RandomRGBColor(255);
                    }
                }

                //SplashKit.ClearScreen();
                SplashKit.RefreshScreen();
            } while (!SplashKit.WindowCloseRequested("Shape Drawer"));
        }
    }
}
```

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

namespace Task2._2
{
    public class Shape
    {
        private Color _color;
        private float _x, _y;
        private int _width, _height;

        public Shape() //contructor can only return the reference of the obj
        {
            _color = Color.Green;
            _x = 0;
            _y = 0;
            _width = 230;
            _height = 150;
        }

        public void Draw()
        {
            SplashKit.FillRectangle(_color, _x, _y, _width, _height);
        }

        public float X
        {
            get { return _x; }

            set { _x = value; }
        }


        public Color Color
        {
            get { return _color; }
            set { _color = value; }

        }


        public float Y
        {
            get { return _y; }

            set { _y = value; }
        }

        public int Height
```

```
54          {
55              get { return _height; }
56              set { _height = value; }
57          }
58          public int Width {
59              get { return _width; }
60              set { _width = value; }
61          }
62
63
64          public bool IsAt(Point2D point)
65          {
66              if (point.X >= _x && point.X <= +_width && point.Y >= _x && +point.Y <=
                ↪  _y + _height)
67              {
68                  return true;
69              }
70              return false;
71          }
72          }
73
74
75      }
```


Shape Drawer
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Task2.2
Live Share
Memory Usage
CPU Usage
Autos
Locals
Watch 1
Output
Ready
Select Repository
17°C
Mostly sunny
ENG
US
12:14 PM
12/4/2022

9 Task 2.3P: Case Study Iteration 1 - Identifiable Object

Object oriented programming really shines with larger programs. In this task we'll get started with our case study, which we'll build up over time as we learn new objected oriented concepts and techniques.

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 2.3P: Case Study Iteration 1 - Identifiable Object

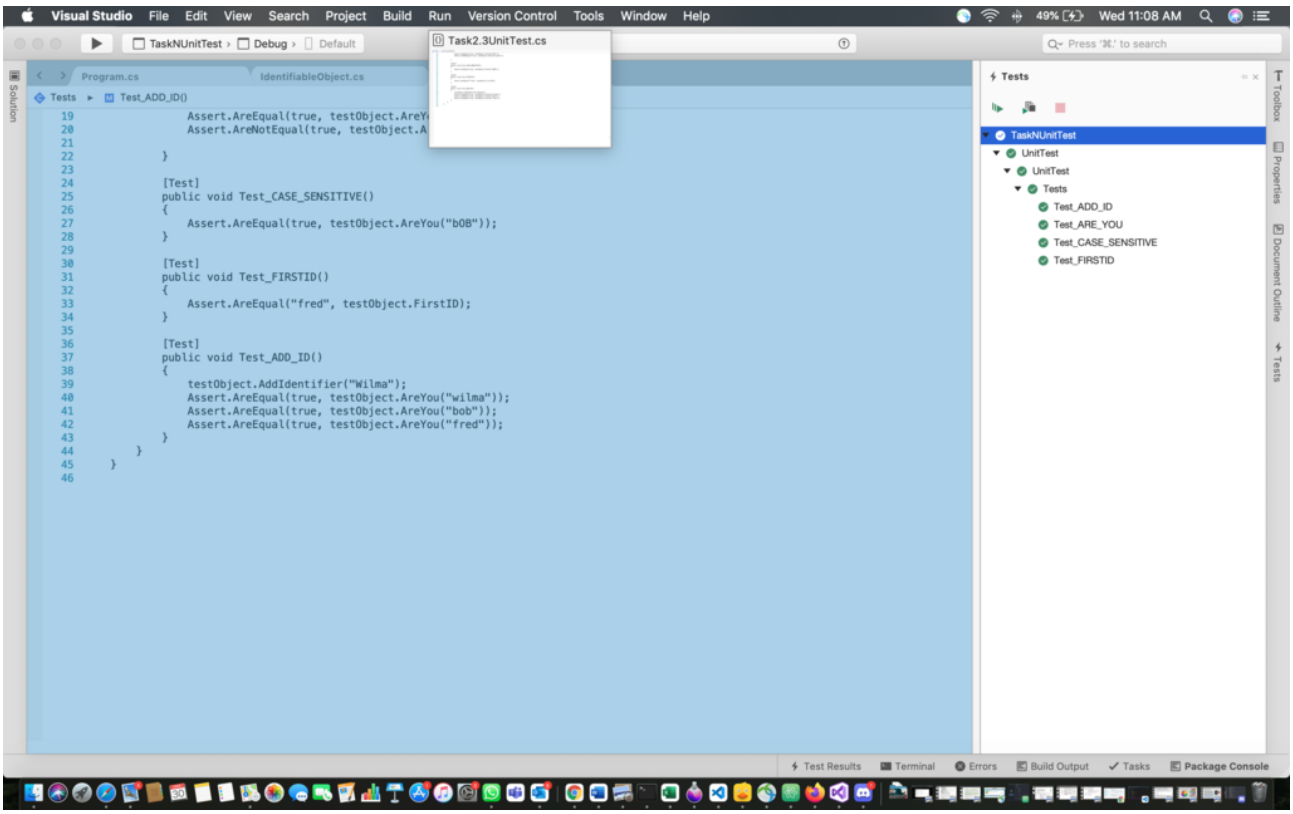
Submitted By:
Vaissheenavi Prabakaran
103508183
2022/03/30 11:08

Tutor:
Jai Cornes

March 30, 2022

```
using System;
using System.Collections.Generic;
namespace TaskNUnitTest
{
    public class IdentifiableObject
    {
        private List<string> _identifiers = new List<string>();

        public IdentifiableObject(string [] idents)
        {
            foreach (string id in idents)
            {
                _identifiers.Add(id.ToLower());
            }
        }

        //private List<string> _identifiers= new List<string>();




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

        public string FirstID
        {
            get
            {
                return _identifiers[0];
            }
        }

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


    }

}
```

```
using NUnit.Framework;
using TaskNUnitTest;
namespace UnitTest
{
    public class Tests
    {
        private IdentifiableObject testObject;

        [SetUp]
        public void Setup()
        {
            string[] array = new string[] { "fred", "bob" };
            testObject = new IdentifiableObject (array);
        }

        [Test]
        public void Test_ARE_YOU()
        {
            Assert.AreEqual(true, testObject.AreYou("bob"));
            Assert.AreNotEqual(true, testObject.AreYou("wilma"));

        }

        [Test]
        public void Test_CASE_SENSITIVE()
        {
            Assert.AreEqual(true, testObject.AreYou("bOB"));
        }

        [Test]
        public void Test_FIRSTID()
        {
            Assert.AreEqual("fred", testObject.FirstID);
        }

        [Test]
        public void Test_ADD_ID()
        {
            testObject.AddIdentifier("Wilma");
            Assert.AreEqual(true, testObject.AreYou("wilma"));
            Assert.AreEqual(true, testObject.AreYou("bob"));
            Assert.AreEqual(true, testObject.AreYou("fred"));
        }
    }
}
```


Visual Studio File Edit View Search Project Build Run Version Control Tools Window Help
Wed 11:08 AM
TaskNUnitTest › Debug › Default
Task2.3UnitTest.cs
Program.cs
IdentifiableObject.cs
Tests ▸ Test_ADD_ID()
Assert.AreEqual(true, testObject.AreYou("bOB"));
[Test]
public void Test_CASE_SENSITIVE()
Assert.AreEqual("fred", testObject.FirstID);
public void Test_FIRSTID()
testObject.AddIdentifier("Wilma");
Assert.AreEqual(true, testObject.AreYou("wilma"));
Assert.AreEqual(true, testObject.AreYou("bob"));
Assert.AreEqual(true, testObject.AreYou("fred"));
Tests
TaskNUnitTest
UnitTest
UnitTest
Tests
Test_ADD_ID
Test_ARE_YOU
Test_CASE_SENSITIVE
Test_FIRSTID
Test Results
Terminal
Errors
Build Output
Tasks
Package Console

10 Task 3.2P: Drawing Program - A Drawing Class

In this task we are going to keep building our drawing program. By adding collaborations we will be able to draw multiple objects to the screen at once!

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 3.2P: Drawing Program - A Drawing Class

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/04/15 08:54

Tutor:
Jai Cornes

April 15, 2022

```
using System;
using SplashKitSDK;



namespace ShapeDrawer
{
    public class Program
    {
        public static void Main()
        {

            new Window("Shape Drawer", 800, 600);
            //_ = new ShapeDrawer();
            Drawing myDrawing = new Drawing();

            do
            {
                SplashKit.ProcessEvents();
                //SplashKit.ClearScreen();

                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape myShape = new Shape();

                    myShape.X = SplashKit.MouseX();
                    myShape.Y = SplashKit.MouseY();


                    myDrawing.AddShapes(myShape);
                }




                if (SplashKit.MouseClicked(MouseButton.RightButton))
                {
                    myDrawing.SelectShapesAt(SplashKit.MousePosition());
                }




                if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
                ↪  (SplashKit.KeyTyped(KeyCode.DeleteKey)))
                {
                    foreach (Shape Shape in myDrawing.SelectedShapes)
                    {
                        myDrawing.RemoveShapes(Shape);
                    }
                }

                myDrawing.Draw();
```

```
53
54                  //SplashKit.ClearScreen();
55                  SplashKit.RefreshScreen();
56              } while (!SplashKit.WindowCloseRequested("Shape Drawer"));
57          }
58      }
59  }
```

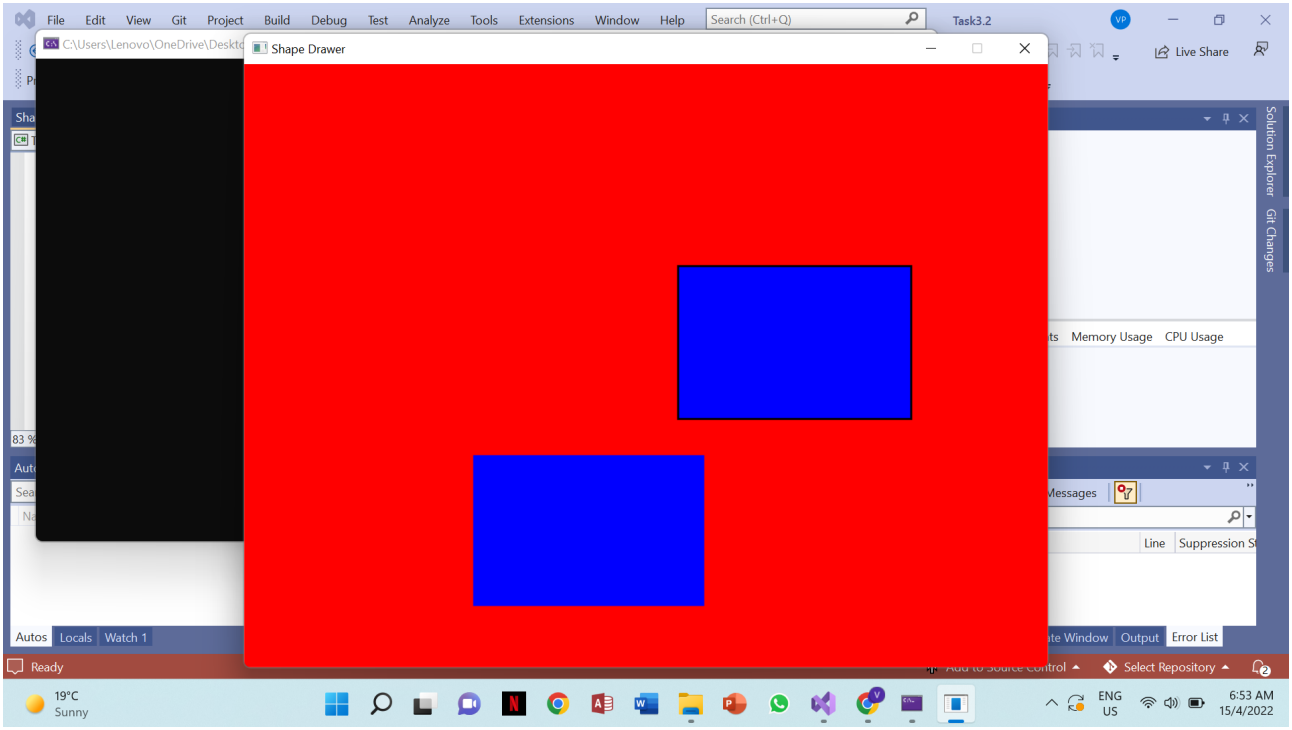
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SplashKitSDK;



//using System.Collections.Generic.List;

namespace ShapeDrawer
{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;

        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> NewShapeSelected = new List<Shape>();

                foreach (Shape shape in _shapes)
                {
                    if (shape.Selected)
                    {
                        NewShapeSelected.Add(shape);
                    }
                }
                return NewShapeSelected;
            }

        }


        public Drawing() : this(Color.Red)
        { }
            //foreach (Shape shape in _shapes)
            //{
              //  shape.Draw();
            //}
        //}



        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;

        }
```

```
54
55
56          public int ShapeCount
57          {
58              get
59              {
60                  return _shapes.Count;
61              }
62          }
63
64
65          public void Draw()
66          {
67              SplashKit.ProcessEvents();
68              //SplashKit.ClearScreen(_background);
69              SplashKit.FillRectangle(_background, 0, 0, SplashKit.ScreenWidth(),
                ↪  SplashKit.ScreenHeight());
70
71              foreach (Shape shape in _shapes) shape.Draw();
72
73          }
74
75
76          public void AddShapes(Shape shape)
77          {
78              _shapes.Add(shape);
79          }
80
81
82          public Color Background
83          {
84              get
85              {
86                  return Background;
87              }
88              set
89              {
90                  Background = value;
91              }
92          }
93
94
95          public void SelectShapesAt(Point2D pt)
96          {
97              foreach (Shape Shapes in _shapes)
98              {
99                  if (Shapes.IsAt(pt))
100                 {
101                     Shapes.Selected = true;
102                 }
103                 else
104                 {
105                     Shapes.Selected = false;
```

```
106                 }
107             }
108         }
109 
110             public void RemoveShapes(Shape shape)
111         {
112             _shapes.Remove(shape);
113         }
114 
115 
116         }
117 
118 
119 
120 
121 
122 
123 
124 }
```

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

namespace ShapeDrawer
{
    public class Shape
    {
        private Color _color;
        private float _x, _y;
        private int _width, _height;
        private bool _selected;

        public Shape() //contructor can only return the reference of the obj
        {
            _color = Color.Blue;
            _x = 0;
            _y = 0;
            _width = 230;
            _height = 150;
        }

        public void Draw()
        {
            if (Selected)
            {
                DrawOutline();
            }

            SplashKit.FillRectangle(_color, _x, _y, _width, _height);
        }

        public void DrawOutline()
        {
            SplashKit.FillRectangle(Color.Black, _x - 2, _y - 2, _width + 4,
            ↪  _height + 4);
        }

        public float X
        {
            get { return _x; }

            set { _x = value; }
        }


        public Color Color
        {
            get { return _color; }
            set { _color = value; }
```

```
53
54          }
55
56
57          public float Y
58          {
59              get { return _y; }
60
61              set { _y = value; }
62          }
63
64          public int Height
65          {
66              get { return _height; }
67              set { _height = value; }
68          }
69          public int Width
70          {
71              get { return _width; }
72              set { _width = value; }
73          }
74
75          public bool Selected
76          {
77              get { return _selected; }
78              set { _selected = value; }
79          }
80
81          public bool IsAt(Point2D point)
82          {
83              if (point.X >= _x && point.X <= _x + _width && point.Y >= _y &&
                ↪  +point.Y <= _y + _height)
84              {
85                  return true;
86              }
87              return false;
88          }
89      }
90
91
92  }
```


Shape Drawer
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Task3.2
Live Share
Memory Usage
CPU Usage
Solution Explorer
Git Changes
Autos
Locals
Watch 1
Error List
Ready
Select Repository
19°C
Sunny
ENG
US
6:53 AM
15/4/2022

11 Task 3.1P: The Stack and Heap

An important part of programming is understanding what is going on behind the scenes. For OO programming, this means thinking about memory, and what is happening on the stack and heap as your program runs. In this task you will have practice explaining and visualising these concepts, using a simple program you wrote in a previous task. This can be challenging, but is well worth it!

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 3.1P: The Stack and Heap

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/04/05 20:56

Tutor:
Jai Cornes

April 5, 2022

Task 3.1P Answer Sheet

Name: Vaissheenavi Prabakaran
Student ID: 103508183

1. How many Counter objects were created?

A total of 2 counter objects.

2. Variables declared in main() are different to the objects created when we call new. What is the relationship between the declared variables in main and the objects created?

Variables declared in main() contains the connection to the objects.

3. Resetting the counter in myCounters[2] also changes the value of the counter in myCounters[0]. Why does this happen?

myCounter[2] and myCounter[0] would have the same memory.

4. The key difference between memory on the heap compared to the stack and the heap is that the heap holds dynamically allocated memory. What does this mean ?

Dynamic memory allocation means the process of assigning memory during run time.

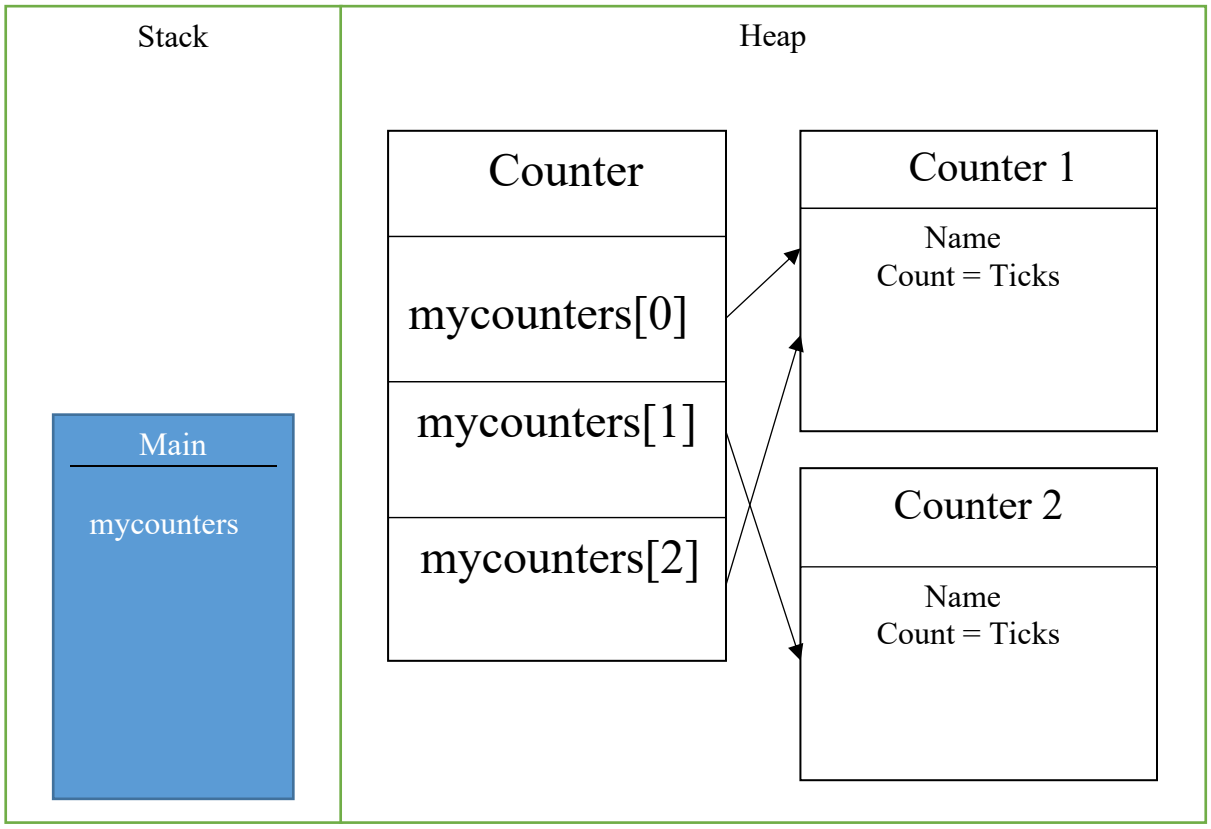
5. On which are objects allocated (heap or stack) ? On which are local variables allocated (heap or stack) ?

Objects are allocated on the heap.
Local variables are allocated on the stack.

6. What does the new() method do when called for a particular class What does it do and what does it return?

When new is called on a class, it generates a new instance type and initialises it into a class object before returning the object's address memory.

7. Draw a diagram showing the locations of the variables and objects in main.



12 Task 3.3P: Clock Class

Reuse is an important part of programming. Design a class well and it can be used in more ways than you originally intended. In this class you will reuse your Counter class to make a clock. You will also practice designing and implementing your own unit tests.

Date	Author	Comment
2022/04/27 11:40	Jai Cornes	demoed

Swinburne University of Technology

Object Oriented Programming (2022 S1)

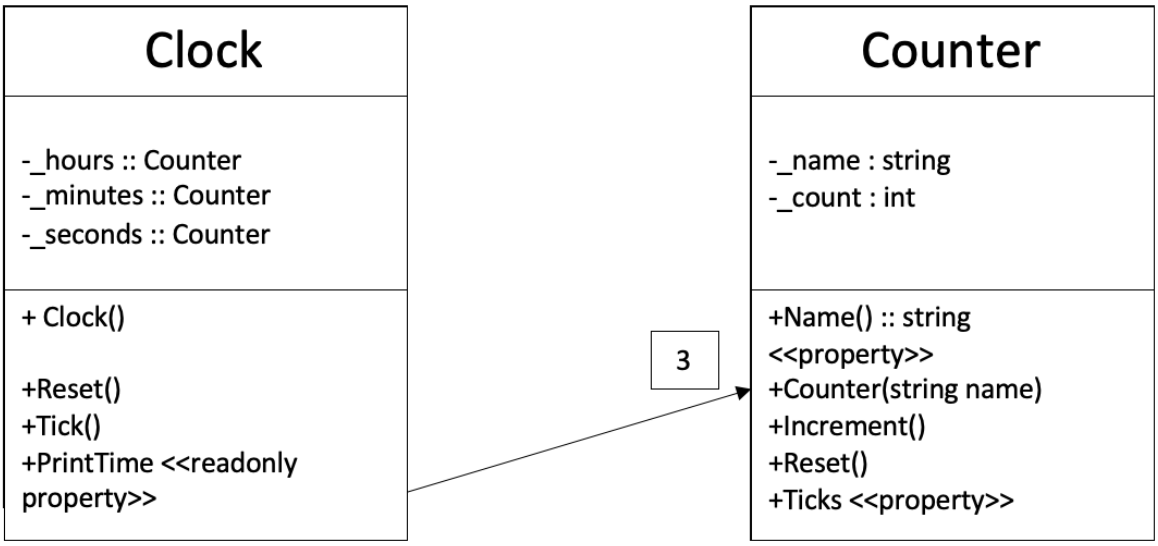
Doubtfire Submission

Task 3.3P: Clock Class

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/04/22 20:02

Tutor:
Jai Cornes

April 22, 2022


Clock
-_hours :: Counter
-_minutes :: Counter
-_seconds :: Counter
+ Clock()
+Reset()
+Tick()
+PrintTime <<readonly property>>
3
Counter
-_name : string
-_count : int
+Name() :: string <<property>>
+Counter(string name)
+Increment()
+Reset()
+Ticks <<property>>

```
using System;
namespace ClockTask
{
    public class Program
    {
        public static void Main(string[] args)
        {
            Clock clock = new Clock();

            //string readtime = "";

            for (int i = 0; i < 87000; i++)
            {
                clock.Tick();
                Console.WriteLine("Clock time is: " + clock.PrintTime);
            }

            //Console.WriteLine("Clock time is: " + clock.PrintTime);
        }
    }
}
```

```
using System;
namespace ClockTask
{
    public class Clock
    {
        private Counter _hours;
        private Counter _minutes;
        private Counter _seconds;


        public Clock()
        {
            _hours = new Counter("hours");
            _minutes = new Counter("minutes");
            _seconds = new Counter("seconds");
        }

        public void Tick()
        {
            _seconds.Increment();
            if (_seconds.Ticks == 60)
            {
                _seconds.Reset();
                _minutes.Increment();
            }

            if (_minutes.Ticks == 60)
            {
                _minutes.Reset();
                _hours.Increment();
            }

             if (_hours.Ticks == 24)
             {
                _hours.Reset();
             }

        }



        public void Reset()
        {
            _hours.Reset();
            _minutes.Reset();
            _seconds.Reset();
        }


        public string PrintTime
        {
            get
            {
```

```
54                  return _hours.Ticks.ToString("00") + ":" +
                     ↪  _minutes.Ticks.ToString("00") + ":" +
                     ↪  _seconds.Ticks.ToString("00");
55
56              }
57
58          }
59
60
61
62
63      }
64  }
```

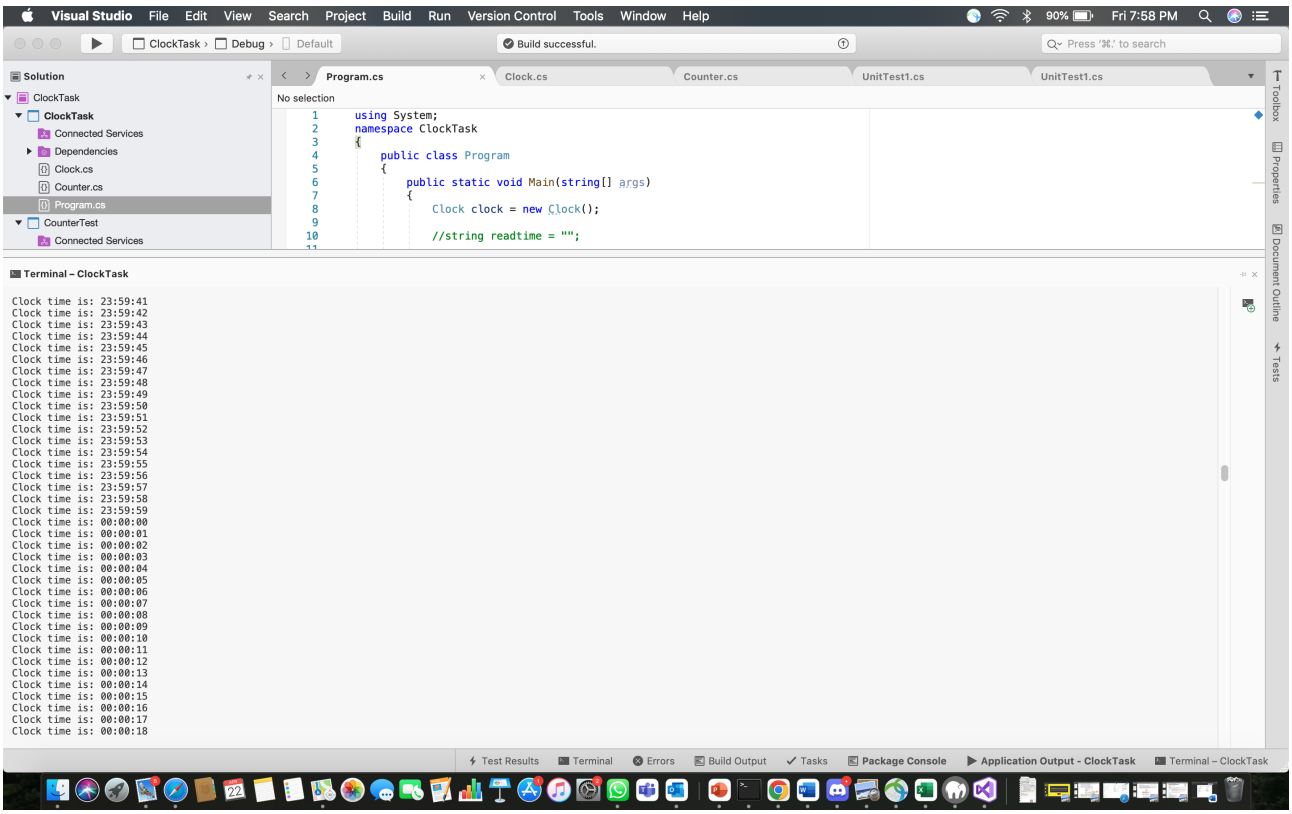
```
using System;
namespace ClockTask
{
    public class Counter
    {
        private int _count;
        private string _name;

        public string Name
        {
            get
            {
                return _name;
            }
            set
            {
                _name = value;
            }
        }

        public Counter(string name)
        {
            _name = name;
            _count = 0;
        }

        public void Increment()
        {
            _count++;
        }

        public void Reset()
        {
            _count = 0;

        }


        public int Ticks
        {
            get
            {
                return _count;
            }

            set
            {
                _count = value;
            }

        }
    }
}
```

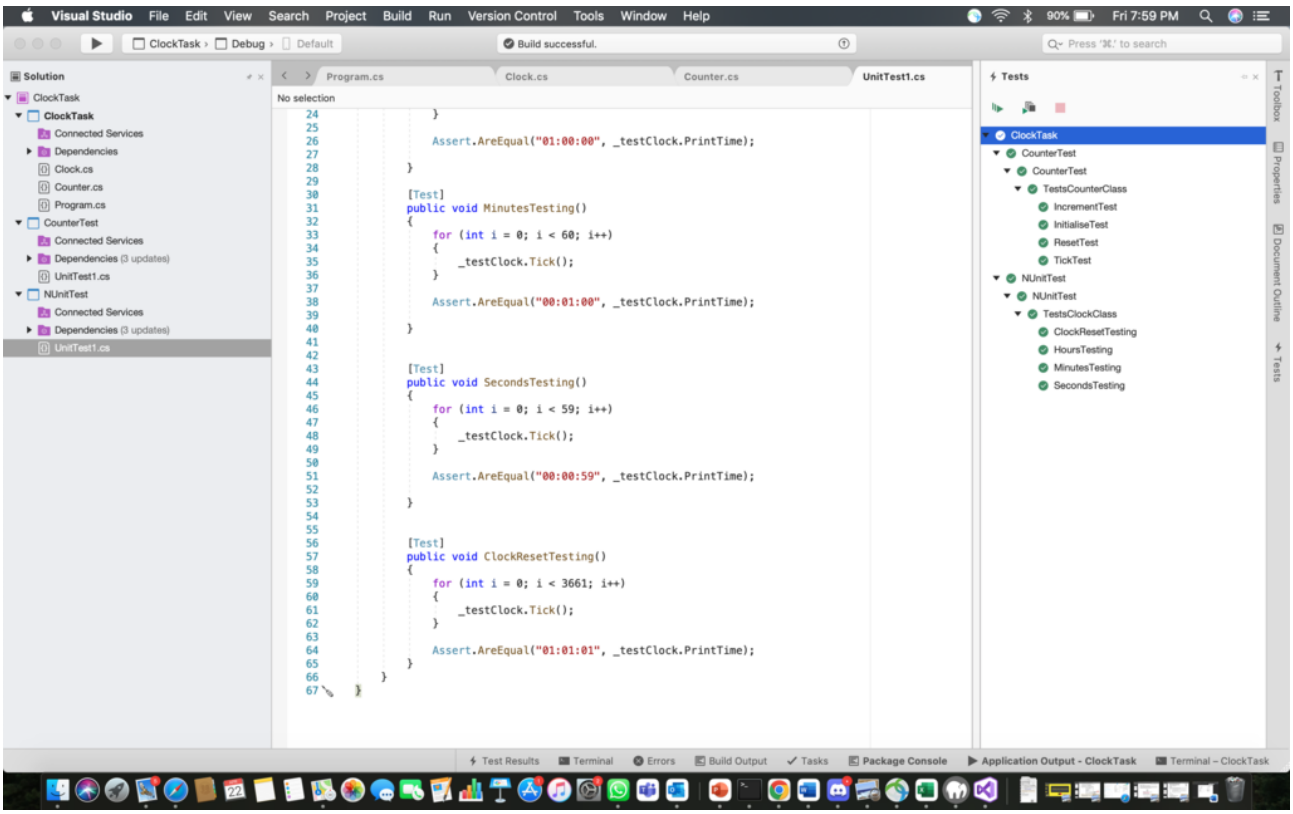
```
using NUnit.Framework;
using ClockTask;

namespace NUnitTest
{
    public class TestsClockClass
    {
        private Clock _testClock;


        [SetUp]
        public void Setup()
        {
            _testClock = new Clock();
        }


        [Test]
        public void HoursTesting()
        {
            for (int i = 0; i < 3600; i++)
            {
                _testClock.Tick();
            }

            Assert.AreEqual("01:00:00", _testClock.PrintTime);

        }

        [Test]
        public void MinutesTesting()
        {
            for (int i = 0; i < 60; i++)
            {
                _testClock.Tick();
            }

            Assert.AreEqual("00:01:00", _testClock.PrintTime);

        }


        [Test]
        public void SecondsTesting()
        {
            for (int i = 0; i < 59; i++)
            {
                _testClock.Tick();
            }

            Assert.AreEqual("00:00:59", _testClock.PrintTime);

        }
```

```
54
55
56          [Test]
57          public void ClockResetTesting()
58          {
59              for (int i = 0; i < 3661; i++)
60              {
61                  _testClock.Tick();
62              }
63
64              Assert.AreEqual("01:01:01", _testClock.PrintTime);
65          }
66      }
67  }
```

```
using NUnit.Framework;
using ClockTask;

namespace CounterTest
{
    public class TestsCounterClass
    {
        private Counter _testCounter;


        [SetUp]
        public void Setup()
        {
            _testCounter = new Counter("testCounter");
        }

        [Test]
        public void ResetTest()
        {
            ClockTask.Counter testCounter = new ClockTask.Counter("First Counter");

            testCounter.Reset();

            Assert.AreEqual(0, testCounter.Ticks);
        }


        [Test]
        public void InitialiseTest()
        {
            ClockTask.Counter testCounter = new ClockTask.Counter("First Counter");

            int expect = 0;
            int actual = testCounter.Ticks;

            Assert.AreEqual(expect, actual);
        }


        [Test]
        public void IncrementTest()
        {
            ClockTask.Counter testCounter = new ClockTask.Counter("First Counter");

            testCounter.Increment();

            testCounter.Increment();

            Assert.AreEqual(2, testCounter.Ticks);
        }


        [Test]
```

```
54          public void TickTest()
55          {
56              ClockTask.Counter testCounter = new ClockTask.Counter("First Counter");
57
58              testCounter.Ticks = 5;
59
60              Assert.AreNotEqual(4, testCounter.Ticks);
61          }
62      }
63  }
```


Visual Studio File Edit View Search Project Build Run Version Control Tools Window Help
90% Fri 7:58 PM
ClockTask › Debug › Default
Build successful.
Press '⌘,' to search
Solution
ClockTask
ClockTask
Connected Services
Dependencies
Clock.cs
Counter.cs
Program.cs
CounterTest
Connected Services
Program.cs
Clock.cs
Counter.cs
UnitTest1.cs
UnitTest1.cs
No selection
1 using System;
2 namespace ClockTask
3 {
4 public class Program
5 {
6 public static void Main(string[] args)
7 {
8 Clock clock = new Clock();
9
10 //string readtime = "";
Toolbox
Properties
Document Outline
Tests
Terminal – ClockTask
Clock time is: 23:59:41
Clock time is: 23:59:42
Clock time is: 23:59:43
Clock time is: 23:59:44
Clock time is: 23:59:45
Clock time is: 23:59:46
Clock time is: 23:59:47
Clock time is: 23:59:48
Clock time is: 23:59:49
Clock time is: 23:59:50
Clock time is: 23:59:51
Clock time is: 23:59:52
Clock time is: 23:59:53
Clock time is: 23:59:54
Clock time is: 23:59:55
Clock time is: 23:59:56
Clock time is: 23:59:57
Clock time is: 23:59:58
Clock time is: 23:59:59
Clock time is: 00:00:00
Clock time is: 00:00:01
Clock time is: 00:00:02
Clock time is: 00:00:03
Clock time is: 00:00:04
Clock time is: 00:00:05
Clock time is: 00:00:06
Clock time is: 00:00:07
Clock time is: 00:00:08
Clock time is: 00:00:09
Clock time is: 00:00:10
Clock time is: 00:00:11
Clock time is: 00:00:12
Clock time is: 00:00:13
Clock time is: 00:00:14
Clock time is: 00:00:15
Clock time is: 00:00:16
Clock time is: 00:00:17
Clock time is: 00:00:18
Test Results Terminal Errors Build Output Tasks Package Console Application Output - ClockTask Terminal – ClockTask


Visual Studio File Edit View Search Project Build Run Version Control Tools Window Help
Build successful.
Solution
ClockTask
Connected Services
Dependencies
Clock.cs
Counter.cs
Program.cs
CounterTest
Dependencies (3 updates)
UnitTest1.cs
NUnitTest
Program.cs
Clock.cs
Counter.cs
UnitTest1.cs
No selection
Assert.AreEqual("01:00:00", _testClock.PrintTime);
[Test]
public void MinutesTesting()
for (int i = 0; i < 60; i++)
_testClock.Tick();
Assert.AreEqual("00:01:00", _testClock.PrintTime);
[Test]
public void SecondsTesting()
for (int i = 0; i < 59; i++)
_testClock.Tick();
Assert.AreEqual("00:00:59", _testClock.PrintTime);
[Test]
public void ClockResetTesting()
for (int i = 0; i < 3661; i++)
_testClock.Tick();
Assert.AreEqual("01:01:01", _testClock.PrintTime);
Tests
ClockTask
CounterTest
CounterTest
TestsCounterClass
IncrementTest
InitialiseTest
ResetTest
TickTest
NUnitTest
NUnitTest
TestsClockClass
ClockResetTesting
HoursTesting
MinutesTesting
SecondsTesting
Test Results
Terminal
Errors
Build Output
Tasks
Package Console
Application Output - ClockTask
Terminal – ClockTask

13 Task 5.1P: Case Study Iteration 3: Bags

Object oriented programming really shines with larger programs. Now that we have added the concepts of players, items, and inventories, we'll build on that work in this task by adding a class to represent a bag or holdable container of items.

Date	Author	Comment
2022/04/26 11:47	Jai Cornes	Bag stuff looks OK. For Player we do want the "You are carrying" that you have commented out though, I think.

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 5.1P: Case Study Iteration 3: Bags

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/04/25 23:06

Tutor:
Jai Cornes

April 26, 2022

```
using System;
using System.Collections.Generic;

namespace Task_5._1
{
    public class IdentifiableObject
    {
        private List<string> _identifiers = new List<string>();



        public IdentifiableObject(string[] idents)
        {
            foreach (string id in idents)
            {
                _identifiers.Add(id.ToLower());
            }
        }

        //private List<string> _identifiers= new List<string>();




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

            foreach (string idAY in _identifiers)
            {
                if (id.ToLower() == idAY)
                {
                    return true;
                }

                //return false;
            }

            return false;
        }


        public string FirstID
        {
            get
            {
                if (_identifiers.Count > 0)
                {
                    return _identifiers[0];
                }

                return "";
            }
```

```
54          }
55
56          public void AddIdentifier(string id)
57          {
58              //id = id.ToLower();
59              _identifiers.Add(id.ToLower());
60
61              return;
62          }
63      }
64  }
```

```
using System;

namespace Task_5._1
{
    public abstract class GameObject : IdentifiableObject
    {
        private string _description;
        private string _name;




        public string Name
        {
            get
            {
                return _name;
            }

        }

        public string ShortDescription
        {
            get
            {
                return _name + " (" + FirstID + ")";
            }

        }


        public virtual string FullDescription
        {
            get
            {
                return _description;
            }

        }


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


    }
}
```

```
using System;
namespace Task_5._1
{
    public class Player : GameObject
    {
        //already in GameObject
        //private string _description;
        //private string _name;
        private Inventory inventory;


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


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

            else if (inventory.HasItem(id))
            {
                return inventory.Fetch(id);
            }

            return null;
        }

        public override string FullDescription
        {
            get
            {
                return inventory.ItemList;
                //string inventorydescription = "You're carrying: " +
                ↪  Inventory.ItemList;
                //return inventorydescription;
            }
        }

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

```
using System;
using System.Collections.Generic;
namespace Task_5._1
{
    public class Inventory
    {
        private List<Item> _items = new List<Item>();


        public Inventory()
        {
            //foreach (Item i in _items)
            //{
            //if (i.AreYou(id))
            //{
            //return true;
            //}

            //else
            //{
            //return false;
            //}

            //}


            //_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);
        }


```

```
54          public Item Fetch(string id)
55          {
56              foreach (Item i in _items)
57              {
58                  if (i.AreYou(id))
59                  {
60                      Item itemToFetch = i;
61
62                      //return true;
63                      return itemToFetch;
64                  }
65
66                  //return null;
67
68              }
69
70              return null;
71          }
72
73
74          public Item Take(string id)
75          {
76              Item i = Fetch(id);
77              if (i != null)
78              {
79                  _items.Remove(i);
80                  return i;
81              }
82
83              return null;
84          }
85
86
87          public string ItemList
88          {
89              get
90              {
91                  string iList = "";
92                  foreach (Item i in _items)
93                  {
94                      iList += "\t" + i.ShortDescription + "\n";
95                  }
96
97                  if (iList == null)
98                  {
99                      return "Item not found!";
100                 }
101
102                 return iList;
103             }
104
105         }
106     }
```

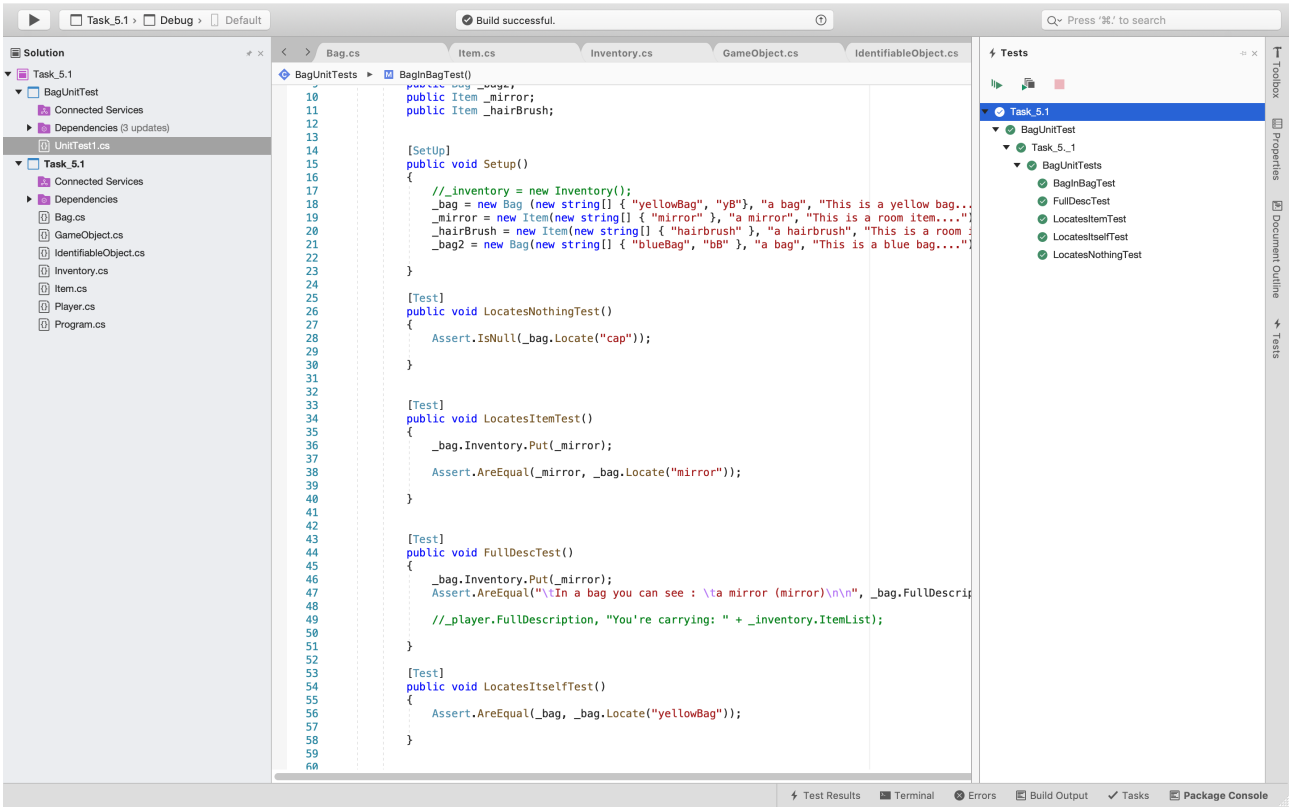
```
107 }
```

```
using System;
namespace Task_5._1
{
    public class Item : GameObject
    {
        //already in GameObject
        //private string _description;
        //private string _name;


        public Item(string[] idents, string name, string desc) : base(idents, name,
        ↪ desc)
        {
            //_name = name;
            //_description = desc;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
namespace Task_5._1
{
    public class Bag : Item
    {
        private Inventory _inventory;

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

        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }

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

        }

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

            else if (_inventory.HasItem(id))
            {
                return _inventory.Fetch(id);
            }

            return null;
        }

    }
}
```

```
using NUnit.Framework;

namespace Task_5._1
{
    public class BagUnitTests
    {

        public Bag _bag;
        public Bag _bag2;
        public Item _mirror;
        public Item _hairBrush;


        [SetUp]
        public void Setup()
        {
            //_inventory = new Inventory();
            _bag = new Bag (new string[] { "yellowBag", "yB"}, "a bag", "This is a yellow bag....");
            _mirror = new Item(new string[] { "mirror" }, "a mirror", "This is a room item....");
            _hairBrush = new Item(new string[] { "hairbrush" }, "a hairbrush", "This is a room item....");
            _bag2 = new Bag(new string[] { "blueBag", "bB" }, "a bag", "This is a blue bag....");

        }

        [Test]
        public void LocatesNothingTest()
        {
            Assert.IsNull(_bag.Locate("cap"));

        }


        [Test]
        public void LocatesItemTest()
        {
            _bag.Inventory.Put(_mirror);

            Assert.AreEqual(_mirror, _bag.Locate("mirror"));

        }


        [Test]
        public void FullDescTest()
        {
            _bag.Inventory.Put(_mirror);
            Assert.AreEqual("\tIn a bag you can see : \ta mirror (mirror)\n\n", _bag.FullDescription);

```

```
49              //_player.FullDescription, "You're carrying: " + _inventory.ItemList);
50
51          }
52
53          [Test]
54          public void LocatesItselfTest()
55          {
56              Assert.AreEqual(_bag, _bag.Locate("yellowBag"));
57
58          }
59
60
61          [Test]
62          public void BagInBagTest()
63          {
64              //Bag firstbag = _bag = new Bag(new string[] { "yellowBag", "yB" }, "a
                ↪  bag", "This is a yellow bag....");
65              //Bag secondbag = new Bag(new string[] { "blueBag", "bB" }, "a bag",
                ↪  "This is a blue bag....");
66
67              _bag.Inventory.Put(_bag2);
68
69              Assert.AreEqual(_bag2, _bag.Locate("blueBag"));
70
71              _bag.Inventory.Put(_hairBrush);
72
73              Assert.AreEqual(_hairBrush, _bag.Locate("hairbrush"));
74
75              _bag2.Inventory.Put(_mirror);
76
77              Assert.AreNotEqual(_mirror, _bag.Locate("_mirror"));
78          }
79      }
80  }
```


Task_5.1 › Debug › Default
Build successful.
Press '⌘.' to search
Solution
Task_5.1
BagUnitTest
Connected Services
Dependencies (3 updates)
UnitTest1.cs
Task_5.1
Connected Services
Dependencies
Bag.cs
GameObject.cs
IdentifiableObject.cs
Inventory.cs
Item.cs
Player.cs
Program.cs
Bag.cs
Item.cs
Inventory.cs
GameObject.cs
IdentifiableObject.cs
BagUnitTests ▸ BagInBagTest()
public Item _mirror;
public Item _hairBrush;
[SetUp]
public void Setup()
{
//_inventory = new Inventory();
_bag = new Bag (new string[] { "yellowBag", "yB"}, "a bag", "This is a yellow bag...
_mirror = new Item(new string[] { "mirror" }, "a mirror", "This is a room item...."
_hairBrush = new Item(new string[] { "hairbrush" }, "a hairbrush", "This is a room
_bag2 = new Bag(new string[] { "blueBag", "bB" }, "a bag", "This is a blue bag...."
}
[Test]
public void LocatesNothingTest()
{
Assert.IsNull(_bag.Locate("cap"));
}
[Test]
public void LocatesItemTest()
{
_bag.Inventory.Put(_mirror);
Assert.AreEqual(_mirror, _bag.Locate("mirror"));
}
[Test]
public void FullDescTest()
{
_bag.Inventory.Put(_mirror);
Assert.AreEqual("\tIn a bag you can see : \ta mirror (mirror)\n\n", _bag.FullDescrip
//_player.FullDescription, "You're carrying: " + _inventory.ItemList);
}
[Test]
public void LocatesItselfTest()
{
Assert.AreEqual(_bag, _bag.Locate("yellowBag"));
}
Tests
Task_5.1
BagUnitTest
Task_5._1
BagUnitTests
BagInBagTest
FullDescTest
LocatesItemTest
LocatesItselfTest
LocatesNothingTest
Toolbox
Properties
Document Outline
Tests
Test Results
Terminal
Errors
Build Output
Tasks
Package Console

14 Task 5.2P: Case Study Iteration 4: Look Command

Object oriented programming really shines with larger programs. Our case study is getting pretty complex now, but it's not yet very interactive. In this task we'll add the basis for some interactions by implementing a command that allows the player to look at their environment and the objects within it.

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 5.2P: Case Study Iteration 4: Look Command

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/05/13 22:54

Tutor:
Jai Cornes

May 13, 2022

```
using System;
using System.Collections.Generic;

namespace Task_5._2
{
    public class IdentifiableObject
    {
        private List<string> _identifiers = new List<string>();



        public IdentifiableObject(string[] idents)
        {
            foreach (string id in idents)
            {
                _identifiers.Add(id.ToLower());
            }
        }

        //private List<string> _identifiers= new List<string>();




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

            foreach (string idAY in _identifiers)
            {
                if (id.ToLower() == idAY)
                {
                    return true;
                }

                //return false;
            }

            return false;
        }


        public string FirstID
        {
            get
            {
                if (_identifiers.Count > 0)
                {
                    return _identifiers[0];
                }

                return "";
            }
```

```
54          }
55
56          public void AddIdentifier(string id)
57          {
58              //id = id.ToLower();
59              _identifiers.Add(id.ToLower());
60
61              return;
62          }
63      }
64  }
```

```
using System;

namespace Task_5._2
{
    public abstract class GameObject : IdentifiableObject
    {
        private string _description;
        private string _name;




        public string Name
        {
            get
            {
                return _name;
            }

        }

        public string ShortDescription
        {
            get
            {
                return _name + " (" + FirstID + ")";
            }

        }


        public virtual string FullDescription
        {
            get
            {
                return _description;
            }

        }


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


    }
}
```

```
using System;
namespace Task_5._2
{
    public class Player : GameObject, IHaveInventory
    {
        //already in GameObject
        //private string _description;
        //private string _name;
        private Inventory inventory;


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


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

            else if (inventory.HasItem(id))
            {
                return inventory.Fetch(id);
            }

            return null;
        }

        public override string FullDescription
        {
            get
            {
                //return inventory.ItemList;
                return "You are carrying: " + Name + inventory.ItemList;

                //return inventorydescription;
            }
        }

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

```
using System;
using System.Collections.Generic;
namespace Task_5._2
{
    public class Inventory
    {
        private List<Item> _items = new List<Item>();


        public Inventory()
        {
            //foreach (Item i in _items)
            //{
            //if (i.AreYou(id))
            //{
            //return true;
            //}

            //else
            //{
            //return false;
            //}

            //}


            //_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);
        }


```

```
54          public Item Fetch(string id)
55          {
56              foreach (Item i in _items)
57              {
58                  if (i.AreYou(id))
59                  {
60                      Item itemToFetch = i;
61
62                      //return true;
63                      return itemToFetch;
64                  }
65
66                  //return null;
67
68              }
69
70              return null;
71          }
72
73
74          public Item Take(string id)
75          {
76              Item i = Fetch(id);
77              if (i != null)
78              {
79                  _items.Remove(i);
80                  return i;
81              }
82
83              return null;
84          }
85
86
87          public string ItemList
88          {
89              get
90              {
91                  string iList = "";
92                  foreach (Item i in _items)
93                  {
94                      iList += "\t" + i.ShortDescription + "\n";
95                  }
96
97                  if (iList == null)
98                  {
99                      return "Item not found!";
100                 }
101
102                 return iList;
103             }
104
105         }
106     }
```

```
107 }
```

```
using System;
namespace Task_5._2
{
    public class Item : GameObject
    {
        //already in GameObject
        //private string _description;
        //private string _name;


        public Item(string[] idents, string name, string desc) : base(idents, name,
        ↪ desc)
        {
            //_name = name;
            //_description = desc;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
namespace Task_5._2
{
    public class Bag : Item, IHaveInventory
    {
        private Inventory _inventory;

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

        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }

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

        }

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

            else if (_inventory.HasItem(id))
            {
                return _inventory.Fetch(id);
            }

            return null;
        }

    }
}
```

```
using System;
namespace Task_5._2
{
    public interface IHaveInventory
    {
        GameObject Locate(string id);

        public string Name
        {
            get;
        }


    }
}
```

```
using System;
namespace Task_5._2
{
    public abstract class Command : IdentifiableObject
    {
        public Command(string [] ids) : base(ids)
        {
        }

        public abstract string Execute(Player p, string[] text);
    }
}
```

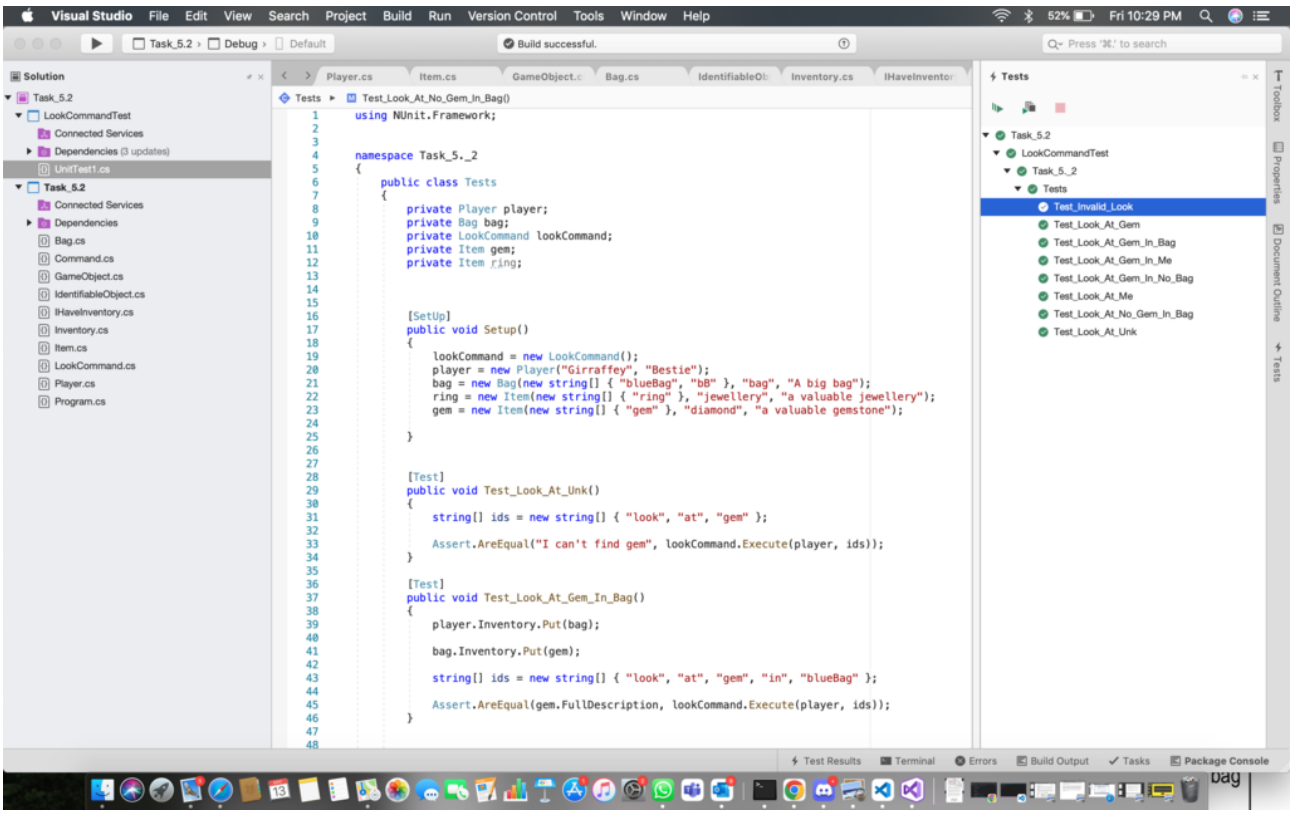
```
using System;
namespace Task_5._2
{
    public class LookCommand : Command
    {
        public LookCommand() : base (new string[] {"look"})
        {
        }

        public override string Execute(Player p, string[] text)
        {
            if(text.Length == 3 || text.Length == 5)
            {
                if (text[0] == "look")
                {
                    if (text[1] == "at")
                    {
                        if(text.Length == 3)
                        {
                            return LookAtIn(text[2], p);
                        }

                        else
                        {
                            if (text[3] == "in")
                            {
                                if(FetchContainer(p, text[4]) == null)
                                {
                                    return "I can't find " + text[2] + " in " +
                                    ↪ text[4];
                                }

                                else
                                {
                                    return LookAtIn(text[2], FetchContainer(p,
                                    ↪ text[4]));
                                }
                            }

                            else
                            {
                                return "What do you want to look in?";
                            }
                        }
                    }

                    else
                    {
                        return "What do you want to look at?";
                    }
                }

                else
```

```
52                  {
53                      return "Error look input";
54                  }
55              }
56
57              else
58              {
59                  return "I don't know how to look like that";
60              }
61          }
62
63
64          private string LookAtIn(string ThingID, IHaveInventory container)
65          {
66              if(container.Locate(ThingID) == null)
67              {
68                  return "I can't find " + ThingID;
69              }
70
71              else
72              {
73                  return container.Locate(ThingID).FullDescription;
74              }
75          }
76
77          private IHaveInventory FetchContainer(Player p, string ContainerID)
78          {
79              return p.Locate(ContainerID) as IHaveInventory;
80          }
81      }
82  }
```

```
using NUnit.Framework;


namespace Task_5._2
{
    public class Tests
    {
        private Player player;
        private Bag bag;
        private LookCommand lookCommand;
        private Item gem;
        private Item ring;



        [SetUp]
        public void Setup()
        {
            lookCommand = new LookCommand();
            player = new Player("Girraffey", "Bestie");
            bag = new Bag(new string[] { "blueBag", "bB" }, "bag", "A big bag");
            ring = new Item(new string[] { "ring" }, "jewellery", "a valuable
            ↪  jewellery");
            gem = new Item(new string[] { "gem" }, "diamond", "a valuable
            ↪  gemstone");

        }


        [Test]
        public void Test_Look_At_Unk()
        {
            string[] ids = new string[] { "look", "at", "gem" };

            Assert.AreEqual("I can't find gem", lookCommand.Execute(player, ids));
        }

        [Test]
        public void Test_Look_At_Gem_In_Bag()
        {
            player.Inventory.Put(bag);

            bag.Inventory.Put(gem);

            string[] ids = new string[] { "look", "at", "gem", "in", "blueBag" };

            Assert.AreEqual(gem.FullDescription, lookCommand.Execute(player, ids));
        }


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

```
52              string[] ids = new string[] { "look", "at", "inventory" };
53
54              Assert.AreEqual(player.FullDescription, lookCommand.Execute(player,
                ↪  ids));
55          }
56
57          [Test]
58          public void Test_Look_At_Gem()
59          {
60              player.Inventory.Put(gem);
61
62              string[] ids = new string[] { "look", "at", "gem" };
63
64              Assert.AreEqual(gem.FullDescription, lookCommand.Execute(player, ids));
65          }
66
67
68
69          [Test]
70          public void Test_Look_At_Gem_In_Me()
71          {
72              player.Inventory.Put(gem);
73
74              string[] ids = new string[] { "look", "at", "gem", "in", "inventory" };
75
76              Assert.AreEqual(gem.FullDescription, lookCommand.Execute(player, ids));
77          }
78
79          [Test]
80          public void Test_Look_At_No_Gem_In_Bag()
81          {
82              bag = new Bag(new string[] { "blueBag", "bB" }, "bag", "A big bag");
83
84              player.Inventory.Put(bag);
85
86              player.Inventory.Put(gem);
87
88              string[] ids = new string[] { "look", "at", "gem", "in", "bag" };
89
90              Assert.AreEqual("I can't find gem in bag", lookCommand.Execute(player,
                ↪  ids));
91          }
92
93
94          [Test]
95          public void Test_Look_At_Gem_In_No_Bag()
96          {
97              bag = new Bag(new string[] { "blueBag", "bB" }, "bag", "A big bag");
98
99              string[] ids = new string[] { "look", "at", "bag" };
100
101             Assert.AreEqual("I can't find bag",lookCommand.Execute(player, ids));
102         }
```

```
103
104
105
106         [Test]
107         public void Test_Invalid_Look()
108         {
109             Assert.AreEqual("Error look input" , lookCommand.Execute(player, new
                ↪  string[] { "pick", "up", "ring" }));
110             Assert.AreEqual("I don't know how to look like that" ,
                ↪  lookCommand.Execute(player, new string[] { "look", "at"}));
111             Assert.AreEqual("What do you want to look in?",
                ↪  lookCommand.Execute(player, new string[] { "look", "at", "table",
                ↪  "and", "chair" }));
112             Assert.AreEqual("What do you want to look at?",
                ↪  lookCommand.Execute(player, new string[] { "look", "test", "gem"
                ↪  }));
113         }
114
115     }
116
117 }
```


Visual Studio File Edit View Search Project Build Run Version Control Tools Window Help
Build successful.
Solution
Task_5.2
LookCommandTest
Connected Services
Dependencies (3 updates)
UnitTest1.cs
Task_5.2
Connected Services
Dependencies
Bag.cs
Command.cs
GameObject.cs
IIdentifiableObject.cs
IHaveInventory.cs
Inventory.cs
Item.cs
LookCommand.cs
Player.cs
Program.cs
Player.cs
Item.cs
GameObject.c
Bag.cs
IdentifiableOb
Inventory.cs
IHaveInventor
Tests
Test_Look_At_No_Gem_In_Bag()
using NUnit.Framework;
namespace Task_5._2
{
public class Tests
{
private Player player;
private Bag bag;
private LookCommand lookCommand;
private Item gem;
private Item ring;
[SetUp]
public void Setup()
{
lookCommand = new LookCommand();
player = new Player("Girraffey", "Bestie");
bag = new Bag(new string[] { "blueBag", "bB" }, "bag", "A big bag");
ring = new Item(new string[] { "ring" }, "jewellery", "a valuable jewellery");
gem = new Item(new string[] { "gem" }, "diamond", "a valuable gemstone");
}
[Test]
public void Test_Look_At_Unk()
{
string[] ids = new string[] { "look", "at", "gem" };
Assert.AreEqual("I can't find gem", lookCommand.Execute(player, ids));
}
[Test]
public void Test_Look_At_Gem_In_Bag()
{
player.Inventory.Put(bag);
bag.Inventory.Put(gem);
string[] ids = new string[] { "look", "at", "gem", "in", "blueBag" };
Assert.AreEqual(gem.FullDescription, lookCommand.Execute(player, ids));
}
Tests
Task_5.2
LookCommandTest
Task_5._2
Tests
Test_Invalid_Look
Test_Look_At_Gem
Test_Look_At_Gem_In_Bag
Test_Look_At_Gem_In_Me
Test_Look_At_Gem_In_No_Bag
Test_Look_At_Me
Test_Look_At_No_Gem_In_Bag
Test_Look_At_Unk
Test Results
Terminal
Errors
Build Output
Tasks
Package Console

15 Task 4.1P: Drawing Program: Multiple Shape Kinds

Time to add some extra fanciness to our drawing program by adding mutiple kinds of shapes. We can do this elegantly with our newfound knowledge of inheritance and polymorphism.

Date	Author	Comment
2022/04/23 22:07	Jai Cornes	Good. Take a look though at the constructors. Your Drawing class has good use of ‘this‘, and passes the defaults so you’re not duplicating code. Try the same thing in the rectangle, circle and line.
2022/04/27 11:45	Jai Cornes	demoed
2022/04/27 11:45	Jai Cornes	Maths for radius IsAt is not quite right :)

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 4.1P: Drawing Program: Multiple Shape Kinds

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/04/25 22:37

Tutor:
Jai Cornes

April 26, 2022

```
using System;
using SplashKitSDK;



namespace Task4._1
{
    public class Program
    {
        private enum ShapeKind
        {
            Rectangle,
            Line,
            Circle
        }
        public static void Main()
        {

            new Window("Shape Drawer", 800, 600);
            //_ = new ShapeDrawer();
            Drawing myDrawing = new Drawing();

            ShapeKind kindToAdd = ShapeKind.Rectangle;

            do
            {
                SplashKit.ProcessEvents();
                //SplashKit.ClearScreen();

                if (SplashKit.KeyTyped(KeyCode.LKey))
                {
                    kindToAdd = ShapeKind.Line;
                }


                if (SplashKit.KeyTyped(KeyCode.RKey))
                {
                    kindToAdd = ShapeKind.Rectangle;
                }


                if (SplashKit.KeyTyped(KeyCode.CKey))
                {
                    kindToAdd = ShapeKind.Circle;
                }

                if (SplashKit.MouseClicked(MouseButton.LeftButton))
                {
                    Shape myShape = new MyLine();


                    if (kindToAdd == ShapeKind.Rectangle)
                    {
```

```
54                      MyRectangle newRectangle = new MyRectangle();
55                      myShape = newRectangle;
56                  }
57                  else if (kindToAdd == ShapeKind.Circle)
58                  {
59                      MyCircle newCircle = new MyCircle();
60                      myShape = newCircle;
61                  }
62                  else if (kindToAdd == ShapeKind.Line)
63                  {
64                      MyLine newLine = new MyLine();
65                      myShape = newLine;
66                  }
67
68
69                  myShape.X = SplashKit.MouseX();
70                  myShape.Y = SplashKit.MouseY();
71
72
73                  myDrawing.AddShapes(myShape);
74              }
75
76
77
78
79              if (SplashKit.MouseClicked(MouseButton.RightButton))
80              {
81                  myDrawing.SelectShapesAt(SplashKit.MousePosition());
82              }
83
84
85
86
87
88              if (SplashKit.KeyTyped(KeyCode.SpaceKey))
89              {
90                  myDrawing.Background = SplashKit.RandomRGBColor(255);
91              }
92
93
94              if (SplashKit.KeyTyped(KeyCode.BackspaceKey) ||
                ↪  (SplashKit.KeyTyped(KeyCode.DeleteKey)))
95              {
96                  foreach (Shape Shape in myDrawing.SelectedShapes)
97                  {
98                      myDrawing.RemoveShapes(Shape);
99                  }
100             }
101
102             myDrawing.Draw();
103
104             //SplashKit.ClearScreen();
105             SplashKit.RefreshScreen();
```

```
106             } while (!SplashKit.WindowCloseRequested("Shape Drawer"));
107         }
108     }
109 }
```

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



//using System.Collections.Generic.List;

namespace Task4._1
{
    public class Drawing
    {
        private readonly List<Shape> _shapes;
        private Color _background;

        public List<Shape> SelectedShapes
        {
            get
            {
                List<Shape> NewShapeSelected = new List<Shape>();

                foreach (Shape shape in _shapes)
                {
                    if (shape.Selected)
                    {
                        NewShapeSelected.Add(shape);
                    }
                }
                return NewShapeSelected;
            }

        }


        public Drawing() : this(Color.Red)
        { }
        //foreach (Shape shape in _shapes)
        //{
        //  shape.Draw();
        //}
        //}



        public Drawing(Color background)
        {
            _shapes = new List<Shape>();
            _background = background;

        }
```

```
54
55
56          public int ShapeCount
57          {
58              get
59              {
60                  return _shapes.Count;
61              }
62          }
63
64
65          public void Draw()
66          {
67              SplashKit.ProcessEvents();
68              //SplashKit.ClearScreen(_background);
69              SplashKit.FillRectangle(_background, 0, 0, SplashKit.ScreenWidth(),
                ↪  SplashKit.ScreenHeight());
70
71              foreach (Shape shape in _shapes) shape.Draw();
72
73          }
74
75
76          public void AddShapes(Shape shape)
77          {
78              _shapes.Add(shape);
79          }
80
81
82          public Color Background
83          {
84              get
85              {
86                  return _background;
87              }
88              set
89              {
90                  _background = value;
91              }
92          }
93
94
95          public void SelectShapesAt(Point2D pt)
96          {
97              foreach (Shape Shapes in _shapes)
98              {
99                  if (Shapes.IsAt(pt))
100                 {
101                     Shapes.Selected = true;
102                 }
103                 else
104                 {
105                     Shapes.Selected = false;
```

```
106                 }
107             }
108         }
109
110         public void RemoveShapes(Shape shape)
111         {
112             _shapes.Remove(shape);
113         }
114
115
116     }
117
118
119
120
121
122
123
124 }
```

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

namespace Task4._1
{
    public abstract class Shape
    {
        private Color _color;
        private float _x, _y;
        //private int _width, _height;
        private bool _selected;

        public Shape() //contructor can only return the reference of the obj
        {
            Color = Color.Yellow;
            //_x = 0;
            //_y = 0;
            //_width = 230;
            //_height = 150;
        }

        public abstract void Draw();


        public abstract void DrawOutline();


        public float X
        {
            get { return _x; }

            set { _x = value; }
        }


        public Color Color
        {
            get { return _color; }
            set { _color = value; }

        }


        public float Y
        {
            get { return _y; }

            set { _y = value; }
        }
```

```
54
55
56          public bool Selected
57          {
58              get { return _selected; }
59              set { _selected = value; }
60          }
61
62          public abstract bool IsAt(Point2D point);
63
64
65
66
67      }
68      }
```

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

namespace Task4._1
{
    public class MyRectangle : Shape
    {
        private int height, width;

        public MyRectangle(Color clr, float x, float y, int width, int height)
        {
            Color = clr;
            X = x;
            Y = y;
            Height = height;
            Width = width;

        }

        public MyRectangle(): this(Color.Green,0,0,100,100)
        {
            //Color = Color.Green;
            //Height = 100;
            //Width = 100;
            //X = 0;
            //Y = 0;
        }

        public int Height
        {
            get { return height; }
            set { height = value; }
        }


        public int Width
        {
            get { return width; }
            set { width = value; }
        }

        public override bool IsAt (Point2D point)
        {
            if (point.X >= X && point.X <= X + width && point.Y >= Y && +point.Y <=
            ↪  Y + height)
            {
                return true;
            }
            return false;
```

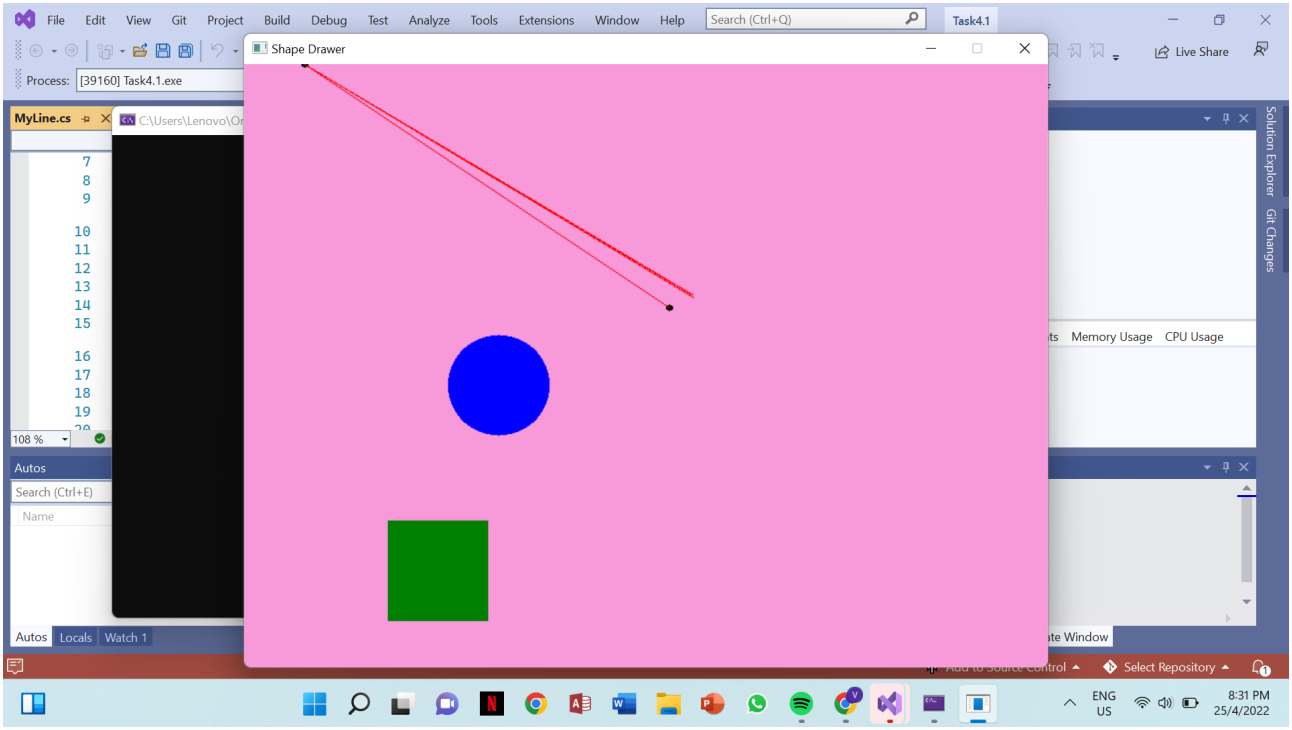
```
53          }
54
55          public override void Draw()
56          {
57              if (Selected)
58              {
59                  DrawOutline();
60              }
61
62              SplashKit.FillRectangle(Color, X, Y, width, height);
63          }
64
65          public override void DrawOutline()
66          {
67              SplashKit.FillRectangle(Color.Black, X - 2, Y - 2, width + 4, height +
                ↪ 4);
68          }
69      }
70  }
```

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

namespace Task4._1
{
    public class MyCircle : Shape
    {
        private int radius;

        public int Radius
        {
            get
            {
                return radius;
            }

            set
            {
                radius = value;
            }
        }

        public MyCircle(Color clr, int radius)
        {
            Radius = radius;
            Color = clr;
        }


        public MyCircle(): this(Color.Blue,50)
        {
            //Color = Color.Blue;
            //Radius = 50;
        }

        public override bool IsAt(Point2D point)
        {
            if (point.X >= X && point.X <= X + Radius && point.Y >= Y && +point.Y
            ↪  <= Y + Radius)
            {
                return true;
            }
            return false;
        }


        public override void Draw()
        {
            if (Selected)
```

```
53              {
54                  DrawOutline();
55              }
56
57              SplashKit.FillCircle(Color, X, Y, radius);
58          }
59
60
61          public override void DrawOutline()
62          {
63              SplashKit.FillCircle(Color.Black, X, Y, + Radius + 2);
64          }
65      }
66  }
```

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

namespace Task4._1
{
    public class MyLine : Shape
    {

        private float _endX, _endY;


        public MyLine() : this(Color.Red,0,0, 60,0)
        {
            //Color = Color.Red;
        }


        public MyLine(Color clr, int x, int y, float xEnd, float yEnd)
        {
            Color = clr;
            X = x;
            Y = y;
            EndX = xEnd;
            EndY = yEnd;
        }


        public float EndX
        {
            get
            {
                return _endX;
            }
            set
            {
                _endX = value;
            }
        }

        public float EndY
        {
            get
            {
                return _endY;
            }
            set
            {
                _endY = value;
            }
```

```
54          }
55
56
57          public override bool IsAt(Point2D point)
58          {
59              if (point.X >= X && point.X <= X + EndX && point.Y >= Y && +point.Y <=
                ↪  Y + EndY)
60              {
61                  return true;
62              }
63              return false;
64          }
65
66
67          public override void Draw()
68          {
69              if (Selected)
70              {
71                  DrawOutline();
72              }
73
74              SplashKit.DrawLine(Color, X, Y, EndX, EndY);
75          }
76
77
78          public override void DrawOutline()
79          {
80              SplashKit.FillCircle(Color.Black, X, Y, 3); //hard code the values for
                ↪  the end point
81              SplashKit.FillCircle(Color.Black, EndX, EndY, 3); //hard code the
                ↪  values for the end point
82          }
83      }
84  }
85
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Task4.1
Live Share
Process: [39160] Task4.1.exe
Shape Drawer
MyLine.cs
Memory Usage
CPU Usage
Autos
Search (Ctrl+E)
Name
Autos
Locals
Watch 1
Select Repository
ENG
US
8:31 PM
25/4/2022

16 Task 4.2P: Case Study Iteration 2: Player Class and Inventory

Object oriented programming really shines with larger programs. In this task we'll continue with our case study, implementing some additional features which benefit from the application of polymorphism and inheritance.

Date	Author	Comment
2022/05/04 11:33	Jai Cornes	demoed

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 4.2P: Case Study Iteration 2: Player Class and Inventory

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/04/23 15:49

Tutor:
Jai Cornes

April 23, 2022

```
using System;
using System.Collections.Generic;


namespace Task_4._2
{
    public class IdentifiableObject
    {
        private List<string> _identifiers = new List<string>();



        public IdentifiableObject(string[] idents)
        {
            foreach (string id in idents)
            {
                _identifiers.Add(id.ToLower());
            }
        }

        //private List<string> _identifiers= new List<string>();




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

            foreach(string idAY in _identifiers)
            {
                if (id.ToLower() == idAY)
                {
                    return true;
                }

                //return false;
            }

            return false;
        }


        public string FirstID
        {
            get
            {
                if(_identifiers.Count > 0)
                {
                    return _identifiers[0];
                }

                return "";
```

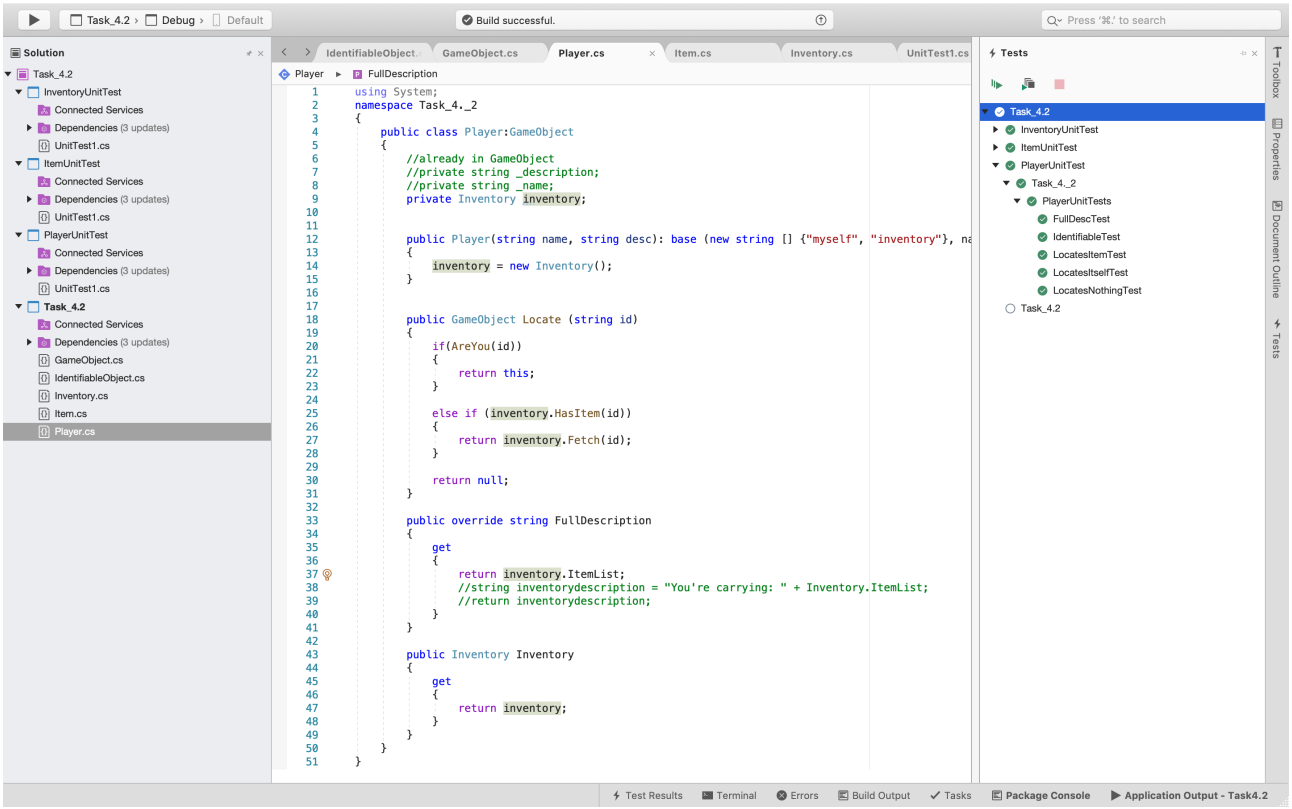
```
54              }
55          }
56
57          public void AddIdentifier(string id)
58          {
59              //id = id.ToLower();
60              _identifiers.Add(id.ToLower());
61
62              return;
63          }
64
65
66
67      }
68
69  }
```

```
using System;

namespace Task_4._2
{
    public abstract class GameObject : IdentifiableObject
    {
        private string _description;
        private string _name;




        public string Name
        {
            get
            {
                return _name;
            }

        }

        public string ShortDescription
        {
            get
            {
                return _name + " (" + FirstID + ")";
            }

        }


        public virtual string FullDescription
        {
            get
            {
                return _description;
            }

        }


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


    }
}
```

```
using System;
namespace Task_4._2
{
    public class Player:GameObject
    {
        //already in GameObject
        //private string _description;
        //private string _name;
        private Inventory inventory;


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


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

            else if (inventory.HasItem(id))
            {
                return inventory.Fetch(id);
            }

            return null;
        }

        public override string FullDescription
        {
            get
            {
                return inventory.ItemList;
                //string inventorydescription = "You're carrying: " +
                ↪  Inventory.ItemList;
                //return inventorydescription;
            }
        }

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

```
using NUnit.Framework;

namespace Task_4._2
{
    public class PlayerUnitTests
    {
        public Player _player;
        public Item _mirror;
        public Inventory _inventory;


        [SetUp]
        public void Setup()
        {
            _inventory = new Inventory();
            _player = new Player("Fred", "You're carrying: ");
            _mirror = new Item(new string[] { "mirror", "comb" }, "a mirror", "This
            ↪  is a room item....");
        }


        [Test]
        public void IdentifiableTest()
        {
            Assert.IsTrue(_player.AreYou("myself"));
            Assert.IsTrue(_player.AreYou("inventory"));
        }


        [Test]
        public void LocatesNothingTest()
        {
            Assert.IsNull(_player.Locate("handbag"));

        }


        [Test]
        public void LocatesItemTest()
        {
            _inventory.Put(_mirror);
            _player.Locate("mirror");
            Assert.AreEqual("\ta mirror (mirror)\n", _inventory.ItemList);
            Assert.IsTrue(_inventory.HasItem("mirror"));
        }


        [Test]
        public void FullDescTest()
        {
            _inventory.Put(_mirror);
            StringAssert.Contains("You're carrying: " + _player.FullDescription,
            ↪  "You're carrying: " + _inventory.ItemList);
```

```
52
53          }
54
55          [Test]
56          public void LocatesItselfTest()
57          {
58              Assert.AreEqual(_player, _player.Locate("myself"));
59              Assert.AreEqual(_player, _player.Locate("inventory"));
60          }
61      }
62
63  }
```


Task_4.2 › Debug › Default
Build successful.
Press '⌘.' to search
Solution
Task_4.2
InventoryUnitTest
Connected Services
Dependencies (3 updates)
UnitTest1.cs
ItemUnitTest
Connected Services
Dependencies (3 updates)
UnitTest1.cs
PlayerUnitTest
Connected Services
Dependencies (3 updates)
UnitTest1.cs
Task_4.2
Connected Services
Dependencies (3 updates)
GameObject.cs
IdentifiableObject.cs
Inventory.cs
Item.cs
Player.cs
IdentifiableObject
GameObject.cs
Player.cs
Item.cs
Inventory.cs
UnitTest1.cs
Player ▸ FullDescription
using System;
namespace Task_4._2
{
public class Player:GameObject
{
//already in GameObject
//private string _description;
//private string _name;
private Inventory inventory;
public Player(string name, string desc): base (new string [] {"myself", "inventory"}, na
{
inventory = new Inventory();
}
public GameObject Locate (string id)
{
if(AreYou(id))
{
return this;
}
else if (inventory.HasItem(id))
{
return inventory.Fetch(id);
}
return null;
}
public override string FullDescription
{
get
{
return inventory.ItemList;
//string inventorydescription = "You're carrying: " + Inventory.ItemList;
//return inventorydescription;
}
}
public Inventory Inventory
{
get
{
return inventory;
}
}
}
}
Tests
Task_4.2
InventoryUnitTest
ItemUnitTest
PlayerUnitTest
Task_4._2
PlayerUnitTests
FullDescTest
IdentifiableTest
LocatesItemTest
LocatesItselfTest
LocatesNothingTest
Task_4.2
Toolbox
Properties
Document Outline
Tests
Test Results
Terminal
Errors
Build Output
Tasks
Package Console
Application Output - Task4.2

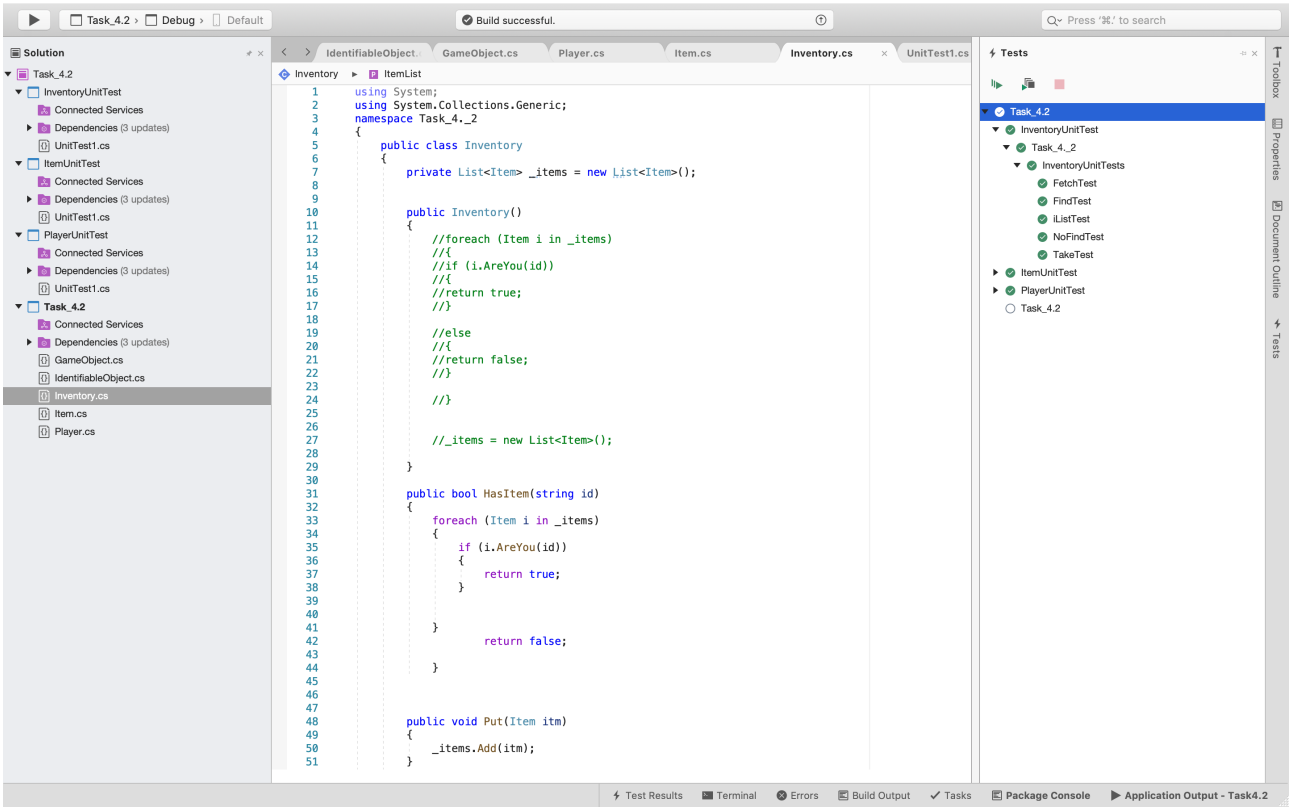
```
using System;
using System.Collections.Generic;
namespace Task_4._2
{
    public class Inventory
    {
        private List<Item> _items = new List<Item>();


        public Inventory()
        {
            //foreach (Item i in _items)
            //{
            //if (i.AreYou(id))
            //{
            //return true;
            //}

            //else
            //{
            //return false;
            //}

            //}


            //_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);
        }


```

```
54          public Item Fetch(string id)
55          {
56              foreach (Item i in _items)
57              {
58                  if (i.AreYou(id))
59                  {
60                      Item itemToFetch = i;
61
62                      //return true;
63                      return itemToFetch;
64                  }
65
66                  //return null;
67
68              }
69
70              return null;
71          }
72
73
74          public Item Take(string id)
75          {
76              Item i = Fetch(id);
77              if (i != null)
78              {
79                  _items.Remove(i);
80                  return i;
81              }
82
83              return null;
84          }
85
86
87          public string ItemList
88          {
89              get
90              {
91                  string iList = "";
92                  foreach (Item i in _items)
93                  {
94                      iList += "\t" + i.ShortDescription + "\n";
95                  }
96
97                  if (iList == null)
98                  {
99                      return "Item not found!";
100                 }
101
102                 return iList;
103             }
104
105         }
106     }
```

```
107 }
108
109
110
111
```

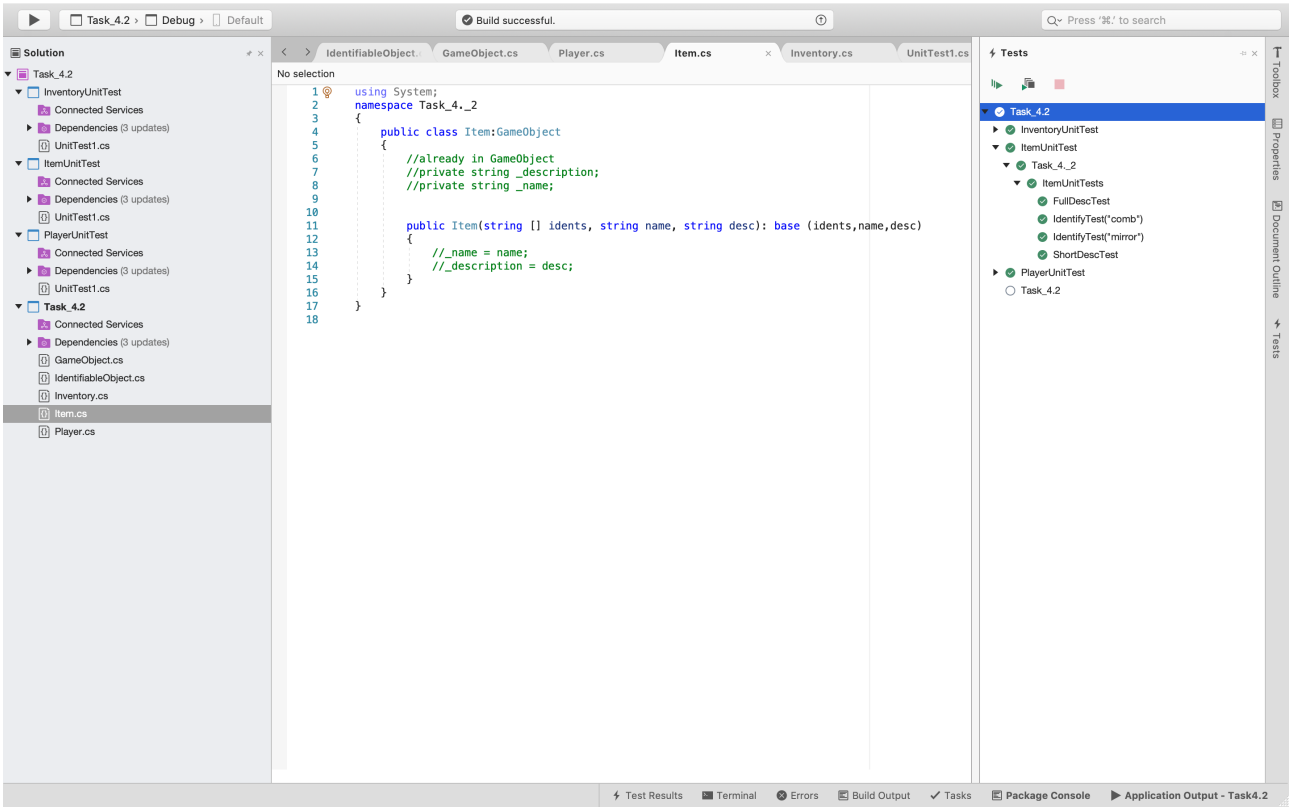
```
using NUnit.Framework;

namespace Task_4._2
{
    public class InventoryUnitTests
    {
        public Inventory _item;
        public Item comb;

        [SetUp]
        public void Setup()
        {
            _item = new Inventory();
            //_item.Put(comb);
            comb = new Item(new string[] { "comb" }, "a green comb", "This is a
            ↪  hair item....");
            _item.Put(comb);
        }


        [Test]
        public void NoFindTest()
        {
            Assert.IsFalse(_item.HasItem("handbag"));
        }


        [Test]
        public void iListTest()
        {
            //_item.Put(comb);
            //Assert.AreEqual("\t" + "a green comb (comb)" + "\n", _item.ItemList);

            string ItemLists = _item.ItemList;
            Assert.AreEqual("\t" + "a green comb (comb)" + "\n", ItemLists);
        }


        [Test]
        public void TakeTest()
        {
            _item.Take("comb");
            Item Comb = _item.Fetch("mirror");
            Assert.IsNull(Comb);

        }


        [Test]
        public void FindTest()
        {
            _item.Put(comb);
            Assert.IsTrue(_item.HasItem("comb"));
```

```
53          }
54
55
56          [Test]
57          public void FetchTest()
58          {
59              _item.Put(comb);
60              Assert.AreEqual(comb, _item.Fetch("comb"));
61              Assert.IsTrue(_item.HasItem("comb"));
62          }
63      }
64  }
```


Task_4.2 › Debug › Default
Build successful.
Press '⌘.' to search
Solution
Task_4.2
InventoryUnitTest
Connected Services
Dependencies (3 updates)
UnitTest1.cs
ItemUnitTest
Connected Services
Dependencies (3 updates)
UnitTest1.cs
PlayerUnitTest
Connected Services
Dependencies (3 updates)
UnitTest1.cs
Task_4.2
Connected Services
Dependencies (3 updates)
GameObject.cs
IdentifiableObject.cs
Inventory.cs
Item.cs
Player.cs
IdentifiableObject
GameObject.cs
Player.cs
Item.cs
Inventory.cs
UnitTest1.cs
Inventory ▸ ItemList
using System;
using System.Collections.Generic;
namespace Task_4._2
{
public class Inventory
{
private List<Item> _items = new List<Item>();
public Inventory()
{
//foreach (Item i in _items)
//{
//if (i.AreYou(id))
//{
//return true;
//}
//else
//{
//return false;
//}
//}
//_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);
}
Tests
Task_4.2
InventoryUnitTest
Task_4._2
InventoryUnitTests
FetchTest
FindTest
IListTest
NoFindTest
TakeTest
ItemUnitTest
PlayerUnitTest
Task_4.2
Toolbox
Properties
Document Outline
Tests
Test Results
Terminal
Errors
Build Output
Tasks
Package Console
Application Output - Task4.2

```
using System;
namespace Task_4._2
{
    public class Item:GameObject
    {
        //already in GameObject
        //private string _description;
        //private string _name;


        public Item(string [] idents, string name, string desc): base
        ↪  (idents,name,desc)
        {
            //_name = name;
            //_description = desc;
        }
    }
}
```

```
using NUnit.Framework;

namespace Task_4._2
{
    public class ItemUnitTests
    {
        public Item _items;


        [SetUp]
        public void Setup()
        {
            _items = new Item(new string[] { "mirror", "comb" }, "a mirror", "This might be fine....");
        }

        [Test]
        public void ShortDescTest()
        {
            Assert.AreEqual("a mirror (mirror)", _items.ShortDescription);
        }


        [Test]
        public void FullDescTest()
        {
            Assert.AreEqual(_items.FullDescription, "This might be fine....");
        }


        [Test]
        [TestCase("mirror")]
        [TestCase("comb")]

        public void IdentifyTest(string id)
        {
            Assert.IsTrue(_items.AreYou(id));
        }
    }
}
```


Task_4.2 › Debug › Default
Build successful.
Press '⌘.' to search
Solution
Task_4.2
InventoryUnitTest
Connected Services
Dependencies (3 updates)
UnitTest1.cs
ItemUnitTest
Connected Services
Dependencies (3 updates)
UnitTest1.cs
PlayerUnitTest
Connected Services
Dependencies (3 updates)
UnitTest1.cs
Task_4.2
Connected Services
Dependencies (3 updates)
GameObject.cs
IdentifiableObject.cs
Inventory.cs
Item.cs
Player.cs
IdentifiableObject
GameObject.cs
Player.cs
Item.cs
Inventory.cs
UnitTest1.cs
No selection
using System;
namespace Task_4._2
{
public class Item:GameObject
{
//already in GameObject
//private string _description;
//private string _name;
public Item(string [] idents, string name, string desc): base (idents,name,desc)
{
//_name = name;
//_description = desc;
}
}
}
Tests
Task_4.2
InventoryUnitTest
ItemUnitTest
Task_4._2
ItemUnitTests
FullDescTest
IdentifyTest("comb")
IdentifyTest("mirror")
ShortDescTest
PlayerUnitTest
Task_4.2
Toolbox
Properties
Document Outline
Tests
Test Results
Terminal
Errors
Build Output
Tasks
Package Console
Application Output - Task4.2

17 Task 6.1P: Case Study Iteration 5: Tying it Together

Object oriented programming really shines with larger programs. Our case study has quite a lot of stuff now. The last step is to put it all together into a functional program!

Date	Author	Comment
2022/05/23 17:19	Jai Cornes	I'm not a fan of 'while(true)' - might be better to use the input - 'while(input !- "quit")'...
2022/05/25 17:40	Jai Cornes	Add "look at me"
2022/05/25 20:42	Jai Cornes	demoed

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 6.1P: Case Study Iteration 5: Tying it Together

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/05/25 18:56

Tutor:
Jai Cornes

May 25, 2022

```
using System;
using System.Collections.Generic;
namespace Task_6._1
{
    public class IdentifiableObject
    {
        private List<string> _identifiers = new List<string>();



        public IdentifiableObject(string[] idents)
        {
            foreach (string id in idents)
            {
                _identifiers.Add(id.ToLower());
            }
        }

        //private List<string> _identifiers= new List<string>();




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

            foreach (string idAY in _identifiers)
            {
                if (id.ToLower() == idAY)
                {
                    return true;
                }

                //return false;
            }

            return false;
        }


        public string FirstID
        {
            get
            {
                if (_identifiers.Count > 0)
                {
                    return _identifiers[0];
                }

                return "";
            }
        }
```

```
54
55          public void AddIdentifier(string id)
56          {
57              //id = id.ToLower();
58              _identifiers.Add(id.ToLower());
59
60              return;
61          }
62      }
63  }
```

```
using System;
namespace Task_6._1
{
    public abstract class GameObject : IdentifiableObject
    {
        private string _description;
        private string _name;




        public string Name
        {
            get
            {
                return _name;
            }

        }

        public string ShortDescription
        {
            get
            {
                return _name + " (" + FirstID + ")";
            }

        }


        public virtual string FullDescription
        {
            get
            {
                return _description;
            }

        }


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


    }
}
```

```
using System;
namespace Task_6._1
{
    public class Player : GameObject, IHaveInventory
    {
        //already in GameObject
        //private string _description;
        //private string _name;
        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))
            {
                return this;
            }

            else if (inventory.HasItem(id))
            {
                return inventory.Fetch(id);
            }

            return null;
        }

        public override string FullDescription
        {
            get
            {
                //return inventory.ItemList;
                return "You are carrying: " + Name + inventory.ItemList;

                //return inventorydescription;
            }
        }

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

```
using System;
using System.Collections.Generic;
namespace Task_6._1
{
    public class Inventory
    {
        private List<Item> _items = new List<Item>();


        public Inventory()
        {
            //foreach (Item i in _items)
            //{
            //if (i.AreYou(id))
            //{
            //return true;
            //}

            //else
            //{
            //return false;
            //}

            //}


            //_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);
        }


```

```
54          public Item Fetch(string id)
55          {
56              foreach (Item i in _items)
57              {
58                  if (i.AreYou(id))
59                  {
60                      Item itemToFetch = i;
61
62                      //return true;
63                      return itemToFetch;
64                  }
65
66                  //return null;
67
68              }
69
70              return null;
71          }
72
73
74          public Item Take(string id)
75          {
76              Item i = Fetch(id);
77              if (i != null)
78              {
79                  _items.Remove(i);
80                  return i;
81              }
82
83              return null;
84          }
85
86
87          public string ItemList
88          {
89              get
90              {
91                  string iList = "";
92                  foreach (Item i in _items)
93                  {
94                      iList += "\t" + i.ShortDescription + "\n";
95                  }
96
97                  if (iList == null)
98                  {
99                      return "Item not found!";
100                 }
101
102                 return iList;
103             }
104
105         }
106     }
```

```
107 }
```

```
using System;
namespace Task_6._1
{
    public class Item : GameObject
    {
        //already in GameObject
        //private string _description;
        //private string _name;


        public Item(string[] idents, string name, string desc) : base(idents, name,
        ↪ desc)
        {
            //_name = name;
            //_description = desc;
        }
    }
}
```

```
using System;
namespace Task_6._1
{
    public class Bag : Item, IHaveInventory
    {
        private Inventory _inventory;

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

        public Inventory Inventory
        {
            get
            {
                return _inventory;
            }
        }

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

        }

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

            else if (_inventory.HasItem(id))
            {
                return _inventory.Fetch(id);
            }

            return null;
        }

    }
}
```

```
using System;
namespace Task_6._1
{
    public interface IHaveInventory
    {
        GameObject Locate(string id);

        public string Name
        {
            get;
        }


    }
}
```

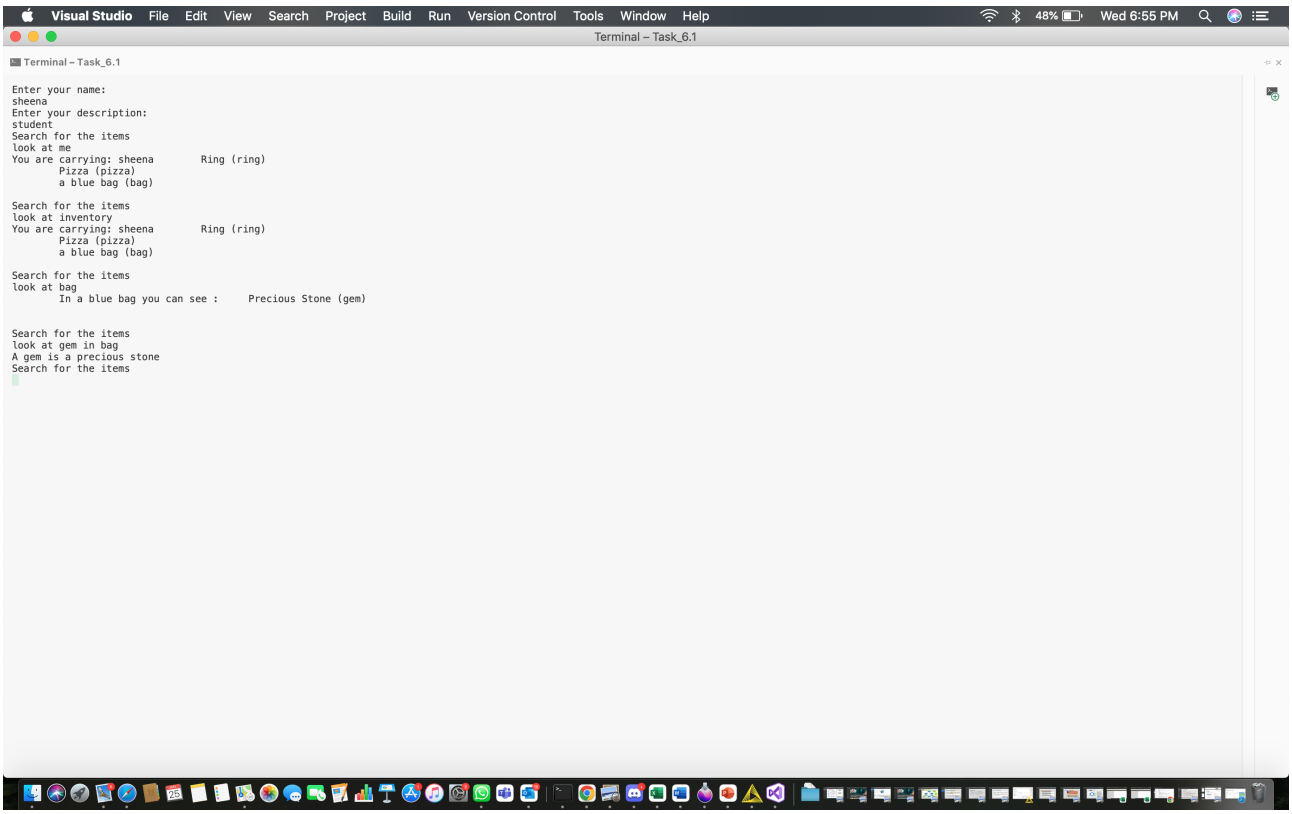
```
using System;
namespace Task_6._1
{
    public abstract class Command : IdentifiableObject
    {
        public Command(string[] ids) : base(ids)
        {
        }

        public abstract string Execute(Player p, string[] text);
    }
}
```

```
using System;
namespace Task_6._1
{
    public class LookCommand : Command
    {
        public LookCommand() : base(new string[] { "look" })
        {
        }

        public override string Execute(Player p, string[] text)
        {
            if (text.Length == 3 || text.Length == 5)
            {
                if (text[0] == "look")
                {
                    if (text[1] == "at")
                    {
                        if (text.Length == 3)
                        {
                            return LookAtIn(text[2], p);
                        }

                        else
                        {
                            if (text[3] == "in")
                            {
                                if (FetchContainer(p, text[4]) == null)
                                {
                                    return "I can't find " + text[2] + " in " +
                                    ↪  text[4];
                                }

                                else
                                {
                                    return LookAtIn(text[2], FetchContainer(p,
                                    ↪  text[4]));
                                }
                            }

                            else
                            {
                                return "What do you want to look in?";
                            }
                        }
                    }

                    else
                    {
                        return "What do you want to look at?";
                    }
                }

                else
```

```
52                  {
53                      return "Error look input";
54                  }
55              }
56
57              else
58              {
59                  return "I don't know how to look like that";
60              }
61          }
62
63
64          private string LookAtIn(string ThingID, IHaveInventory container)
65          {
66              if (container.Locate(ThingID) == null)
67              {
68                  return "I can't find " + ThingID;
69              }
70
71              else
72              {
73                  return container.Locate(ThingID).FullDescription;
74              }
75          }
76
77          private IHaveInventory FetchContainer(Player p, string ContainerID)
78          {
79              return p.Locate(ContainerID) as IHaveInventory;
80          }
81      }
82  }
```

```
using System;

namespace Task_6._1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter your name: ");
            string _namePly = Console.ReadLine();

            Console.WriteLine("Enter your description: ");
            string _descriptPly = Console.ReadLine();

            Player _ply = new Player(_namePly, _descriptPly);

            Item _ring = new Item(new string[] { "ring", "jewellery" }, "Ring", "A
            ↪  shiny jewelley for the fingers");

            _ply.Inventory.Put(_ring);

            Item _pizza = new Item(new string[] { "pizza", "food" }, "Pizza", "A
            ↪  delicious fast food");

            _ply.Inventory.Put(_pizza);


            Bag _bag = new Bag(new string[] { "bag" , "blueBag" }, "a blue bag", "A
            ↪  bag which is blue in colour has...");
            _ply.Inventory.Put(_bag);


            Item _gem = new Item(new string[] { "gem", "a stone" }, "Precious
            ↪  Stone", "A gem is a precious stone");
            _bag.Inventory.Put(_gem);

            LookCommand lkCommand = new LookCommand();
            while (true)
            {
                Console.Write("Search for the items\n");

                string strCommand = Console.ReadLine();

                if(strCommand == "exit")
                {
                    break;
                }

                else
                {
                    Console.WriteLine(lkCommand.Execute(_ply, strCommand.Split()));
                }
            }
```

```
50
51
52              }
53      }
54  }
```


Visual Studio File Edit View Search Project Build Run Version Control Tools Window Help
48% Wed 6:55 PM
Terminal – Task_6.1
Terminal – Task_6.1
Enter your name:
sheena
Enter your description:
student
Search for the items
look at me
You are carrying: sheena Ring (ring)
Pizza (pizza)
a blue bag (bag)
Search for the items
look at inventory
You are carrying: sheena Ring (ring)
Pizza (pizza)
a blue bag (bag)
Search for the items
look at bag
In a blue bag you can see : Precious Stone (gem)
Search for the items
look at gem in bag
A gem is a precious stone
Search for the items

18 Task 8.1P: Semester TEST (Time-limited Evaluation of Skills Task)

While we barely have deadlines in this unit, we do need some markers. This ”test” (Time-limited Evaluation of Skills Task) is here to help us understand where you’re at with your OO programming skills. At this point in the semester you should be able to complete this task, but if not, it will give you an indication of what gaps you need to fill in your understanding.

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 8.1P: Semester TEST (Time-limited Evaluation of Skills Task)

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/04/29 15:54

Tutor:
Jai Cornes

April 29, 2022

```
using System;
using System.Collections.Generic;

namespace OOP_TEST
{
    class Program
    {
        public static void Main(string[] args)
        {
            //Create a DataAnalyser object with a list of 10 numbers and the minmax summary strategy
            DataAnalyser _dataAnalyobj = new DataAnalyser (new List<int> { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 }, new MinMaxSummary());

            //Call the Summarise method
            MinMaxSummary _minMaxobj = new MinMaxSummary();
            _dataAnalyobj.Summarise();

            //Add three more numbers to the data analyser
            _dataAnalyobj.AddNumber(24);
            _dataAnalyobj.AddNumber(15);
            _dataAnalyobj.AddNumber(10);

            //Set the summary strategy to the average strategy
            AverageSummary _avgObj = new AverageSummary();

            //Call the Summarise method
            _dataAnalyobj.summaryStrategy = _avgObj;
            _dataAnalyobj.Summarise();


        }
    }
}
```

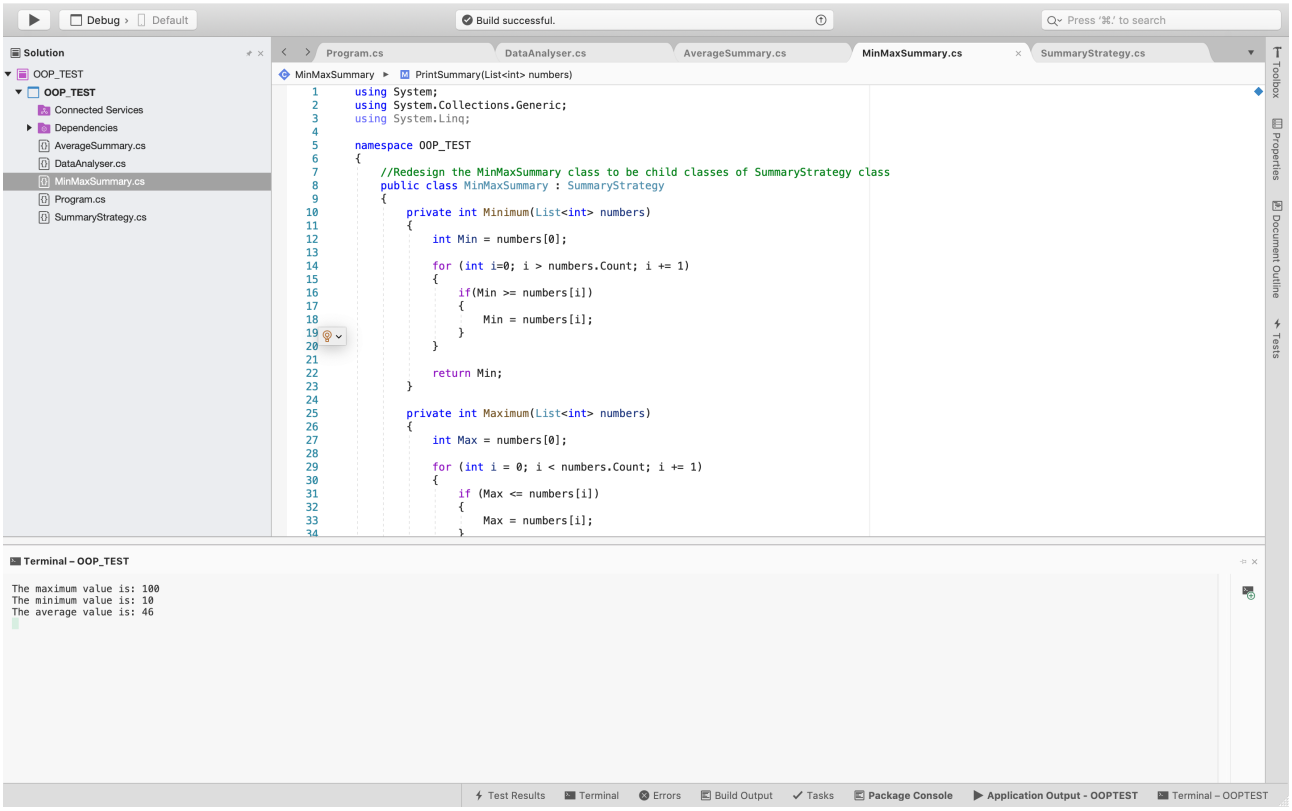
```
using System;
using System.Collections.Generic;
using System.Linq;

namespace OOP_TEST
{
    public class DataAnalyser
    {
        private List<int> _numbers;

        private AverageSummary _avgSummariser;

        private MinMaxSummary _minMaxSummariser;

        //Modify DataAnalyser to have a private variable, "_strategy", that is of
        ↪  the type Summary Strategy
        private SummaryStrategy _strategy;

        //by default, set the strategy to the average strategy
        public DataAnalyser() :this (new List<int>(), new AverageSummary())
        {}

        //allow the strategy to be set through a parameter
        public DataAnalyser(List<int> numbers, SummaryStrategy strategy)
        {
            _numbers = numbers ;
            _strategy = strategy;
            _avgSummariser = new AverageSummary();
            _minMaxSummariser = new MinMaxSummary();

        }


        public void AddNumber(int num)
        {
            _numbers.Add(num);
        }


        //Modify DataAnalyser's Summarise method to use the currently stored
        ↪  strategy instead of relying on a string parameter
        public void Summarise()
        {

            _strategy.PrintSummary(_numbers);
        }


        //Add a public property for this new private variable
        public SummaryStrategy summaryStrategy
        {
            get
            {
```

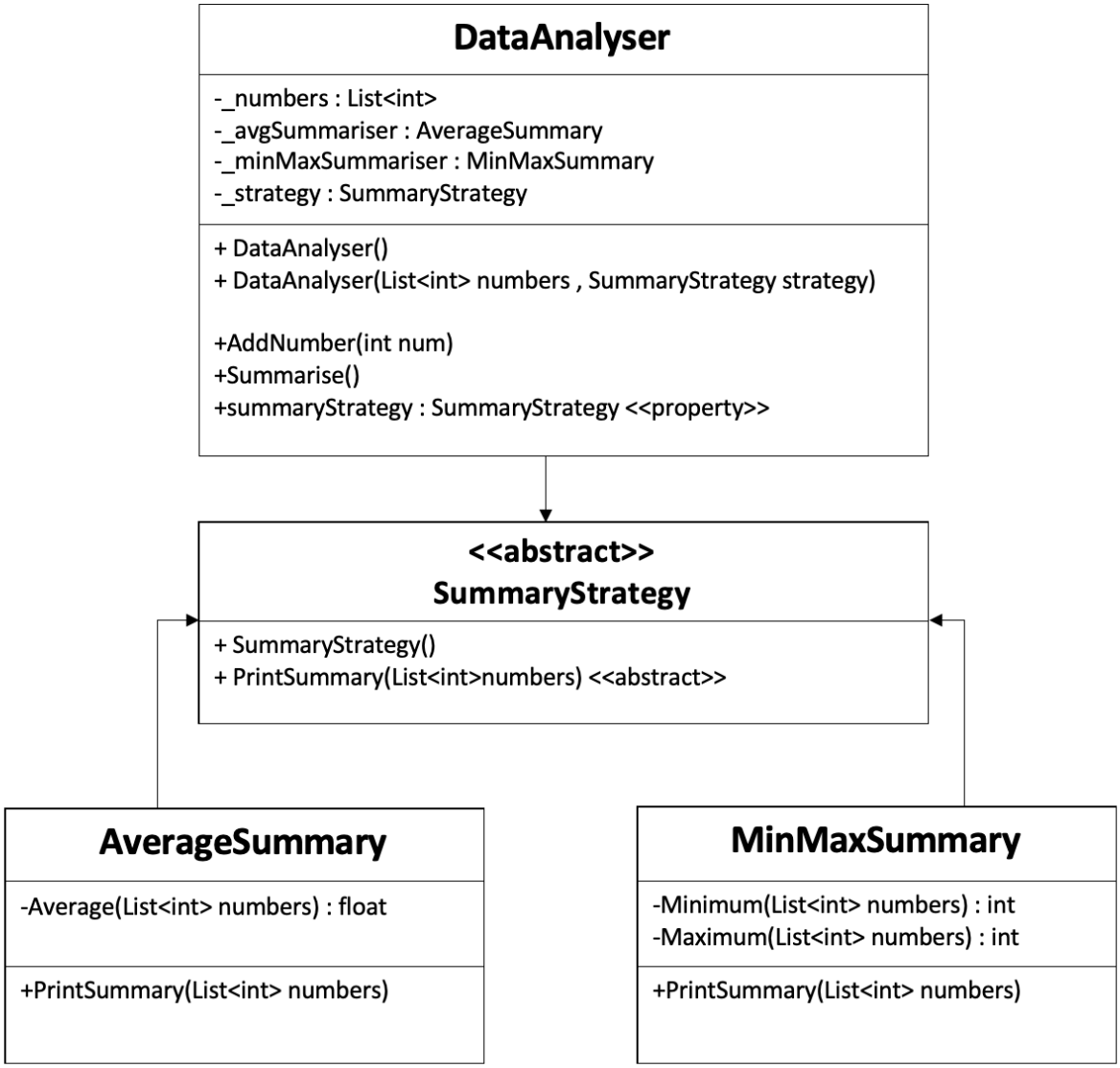
```
52                  return _strategy;
53              }
54
55              set
56              {
57
58                  _strategy = value;
59              }
60          }
61
62      }
63  }
```

```
using System;
using System.Collections.Generic;
namespace OOP_TEST
{
    //Implement the SummaryStrategy abstract class according to the design given
    public abstract class SummaryStrategy
    {
        public SummaryStrategy()
        {

        }

        public abstract void PrintSummary(List<int> numbers);


    }
}
```

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

namespace OOP_TEST
{
    //Redesign the AverageSummary class to be child class of SummaryStrategy class
    public class AverageSummary: SummaryStrategy
    {
        public AverageSummary()
        {

        }


        public override void PrintSummary(List<int> numbers)
        {
            int total = 0;
            int numbercount = numbers.Count;


            for (int i = 0; i < numbers.Count; i += 1)
            {
                total = total + numbers[i];
            }


            int Average = total / numbercount;

            Console.WriteLine("The average value is: " + Average);
        }



    }
}
```

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

namespace OOP_TEST
{
    //Redesign the MinMaxSummary class to be child classes of SummaryStrategy class
    public class MinMaxSummary : SummaryStrategy
    {
        private int Minimum(List<int> numbers)
        {
            int Min = numbers[0];

            for (int i=0; i > numbers.Count; i += 1)
            {
                if(Min >= numbers[i])
                {
                    Min = numbers[i];
                }
            }

            return Min;
        }

        private int Maximum(List<int> numbers)
        {
            int Max = numbers[0];

            for (int i = 0; i < numbers.Count; i += 1)
            {
                if (Max <= numbers[i])
                {
                    Max = numbers[i];
                }
            }

            return Max;
        }

        public override void PrintSummary(List<int> numbers)
        {
            Console.WriteLine("The maximum value is: " + Maximum(numbers));
            Console.WriteLine("The minimum value is: " + Minimum(numbers));
        }



    }
}
```


Debug
Default
Build successful.
Press '⌘.' to search
Solution
OOP_TEST
OOP_TEST
Connected Services
Dependencies
AverageSummary.cs
DataAnalyser.cs
MinMaxSummary.cs
Program.cs
SummaryStrategy.cs
Program.cs
DataAnalyser.cs
AverageSummary.cs
MinMaxSummary.cs
SummaryStrategy.cs
MinMaxSummary
PrintSummary(List<int> numbers)
using System;
using System.Collections.Generic;
using System.Linq;
namespace OOP_TEST
{
//Redesign the MinMaxSummary class to be child classes of SummaryStrategy class
public class MinMaxSummary : SummaryStrategy
{
private int Minimum(List<int> numbers)
{
int Min = numbers[0];
for (int i=0; i > numbers.Count; i += 1)
{
if(Min >= numbers[i])
{
Min = numbers[i];
}
}
return Min;
}
private int Maximum(List<int> numbers)
{
int Max = numbers[0];
for (int i = 0; i < numbers.Count; i += 1)
{
if (Max <= numbers[i])
{
Max = numbers[i];
}
Toolbox
Properties
Document Outline
Tests
Terminal – OOP_TEST
The maximum value is: 100
The minimum value is: 10
The average value is: 46
Test Results
Terminal
Errors
Build Output
Tasks
Package Console
Application Output - OOPTEST
Terminal – OOPTEST


DataAnalyser
-_numbers : List<int>
-_avgSummariser : AverageSummary
-_minMaxSummariser : MinMaxSummary
-_strategy : SummaryStrategy
+ DataAnalyser()
+ DataAnalyser(List<int> numbers , SummaryStrategy strategy)
+AddNumber(int num)
+Summarise()
+summaryStrategy : SummaryStrategy <<property>>
<<abstract>>
SummaryStrategy
+ SummaryStrategy()
+ PrintSummary(List<int>numbers) <<abstract>>
AverageSummary
-Average(List<int> numbers) : float
+PrintSummary(List<int> numbers)
MinMaxSummary
-Minimum(List<int> numbers) : int
-Maximum(List<int> numbers) : int
+PrintSummary(List<int> numbers)

OOP Test 8.1

1. The principle of polymorphism refers to the idea of being able to access objects of many types using the same interface. This interface can be implemented in a variety of ways by each type. The Average Summary and MinMax Summary is inherited from the Summary Strategy. A subtype polymorphism is used in Task 1 namely the Average Summary and MinMax Summary which are different forms of Summary Strategy. Thus, polymorphism is occurring.

2. Abstraction is the process of hiding some details from the user and only displaying them what they need to know. For example, in the Summarise method in Data Analyser class, Print Summary is used and we don't need to know how the Print Summary works and all we need to do is call them.

```
//Modify DataAnalyser's Summarise method to use the currently stored strategy instead of relying on a string parameter
public void Summarise()
{

    _strategy.PrintSummary(_numbers);
}
```

3. The issue with the original design in Task 1 is that in the Summarise method, it was relying on a string parameter, thus 'if' functions had to be done for each summary approach (if typeOfString is "Average"/ if typeOfString is "MinMax"). In this case, an 'if' function has to be done for both the average and minmax. If there were 50 different summary approaches to choose from instead of just 2, it would be time consuming and a very inefficient way of coding as there would be 50 different 'if' statements. To avoid this from happening, we could get the same function done in a single line of code and this would be more efficient. As an example from my code, in the newly implemented design of the Summarise method, it currently stores strategy. It also calls the Print Summary function to display the summary of the '_numbers' called.

19 Task 7.1P: Key Object Oriented Concepts

By this point you should have a fairly solid understanding of the core principles of OOP. In this task we will ask you to demonstrate your understanding by explaining these concepts and visualising how they relate to one another.

Swinburne University of Technology

Object Oriented Programming (2022 S1)

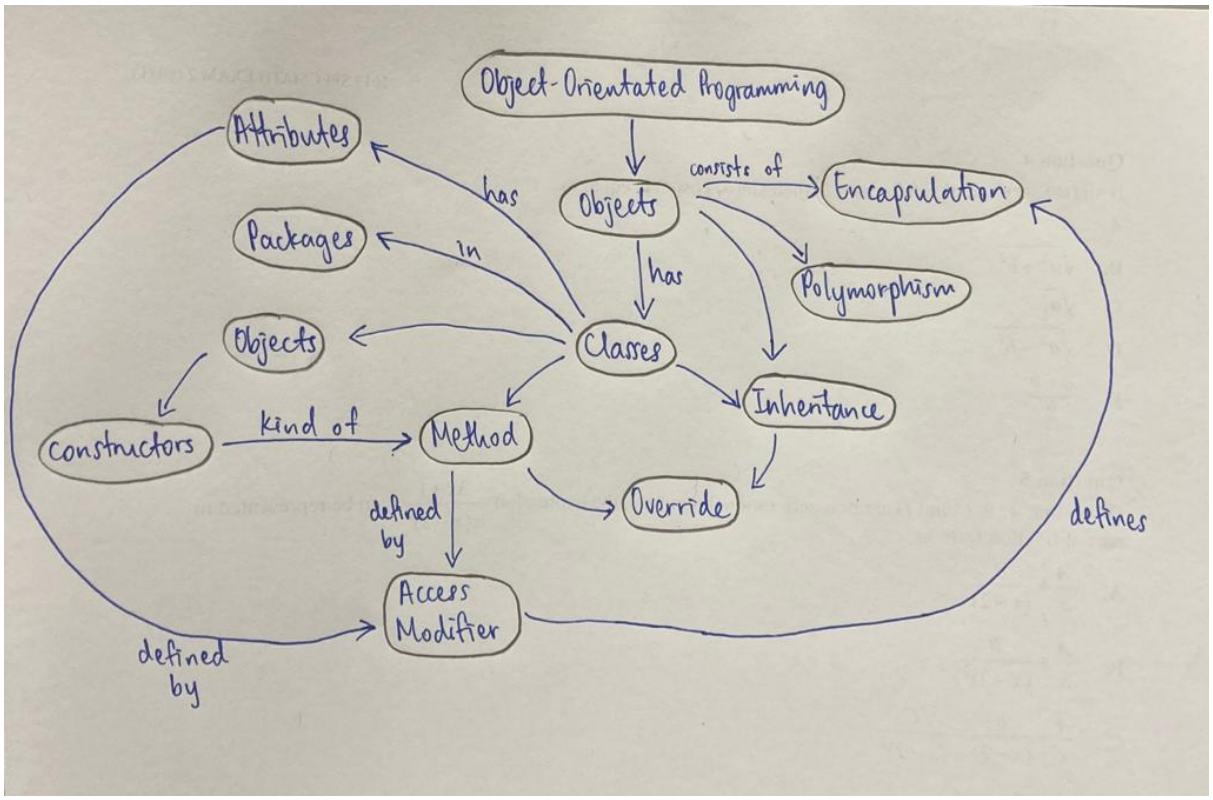
Doubtfire Submission

Task 7.1P: Key Object Oriented Concepts

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/05/23 00:23

Tutor:
Jai Cornes

May 23, 2022



Object-oriented programming (OOP) is a programming language that uses classes and objects. It's used to break down a programme into reusable chunks of code (called classes) that may be utilised to generate individual instances of objects. There are some crucial concepts, artefacts, actions and terminologies of OOP.

Firstly, the three main concepts are encapsulation, polymorphism and inheritance. To illustrate, encapsulation is the way of keeping classes private so they cannot be changed externally, polymorphism allows codes to have different functions and inheritance ensures that the codes are reused in different classes. Besides, abstraction is the process of hiding some details from the user and only displaying them what they need to know. There are also roles, responsibilities and collaborations in OOP. A role is a collection of replaceable responsibilities, responsibilities are obligations to complete a

task or possess information and the interaction of items or/and roles is known as collaboration. Moreover, coupling involves how classes interact with each other, whereas cohesion is focused on how a single class is constructed.

Secondly, some of the artefacts of OOP are class, object, interface, method and fields. The difference between class and object is classes are a blueprint from which objects are created while objects are an instance of a class. Furthermore, interfaces can be used to describe which methods a class should implement and a method is a function that belongs to a class. Fields are a piece of data contained within a class or an object.

Thirdly, there are two main actions in OOP, namely method call and new. Method call is calling a function associated with a class or object while new means creating an instance of a user-defined object or an object that has a constructor.

Last but not least, there are a few common yet important terminologies in OOP. Some of them includes value type, reference type, abstract class, abstract methods, private, public, protected, overload, override and virtual. To explain it further, a value type stores data in the own memory allocation while a reference type stores data in another memory address. An abstract method is a method that is defined but not implemented in the code while an abstract class is a class with at least one abstract method. Besides, private means that only the current class can have access to the field or method, public means that any class can refer to the field or call the method and protected means that only the current class and subclasses of this class will have access to the field or method. Overload means if a function is overloaded, it will be redefined with the same function name. In a base class, the keyword 'virtual' follows the overridde function, which is redefined without the keyword in a derived class.

20 Task 11.1P: Clock in Another Language

Object oriented concepts apply across all object oriented languages. However, when learning a new language it's best to start small so you can learn about and get used to a new set of syntax. In this task you'll do just that by recreating a simple program in a new language.

Date	Author	Comment
2022/05/04 11:35	Jai Cornes	Rewrite this to use the same UML as the original
2022/05/26 09:58	Jai Cornes	We prefer "getName" and "setName" over the S and G, which don't mean much

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 11.1P: Clock in Another Language

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/05/25 18:52

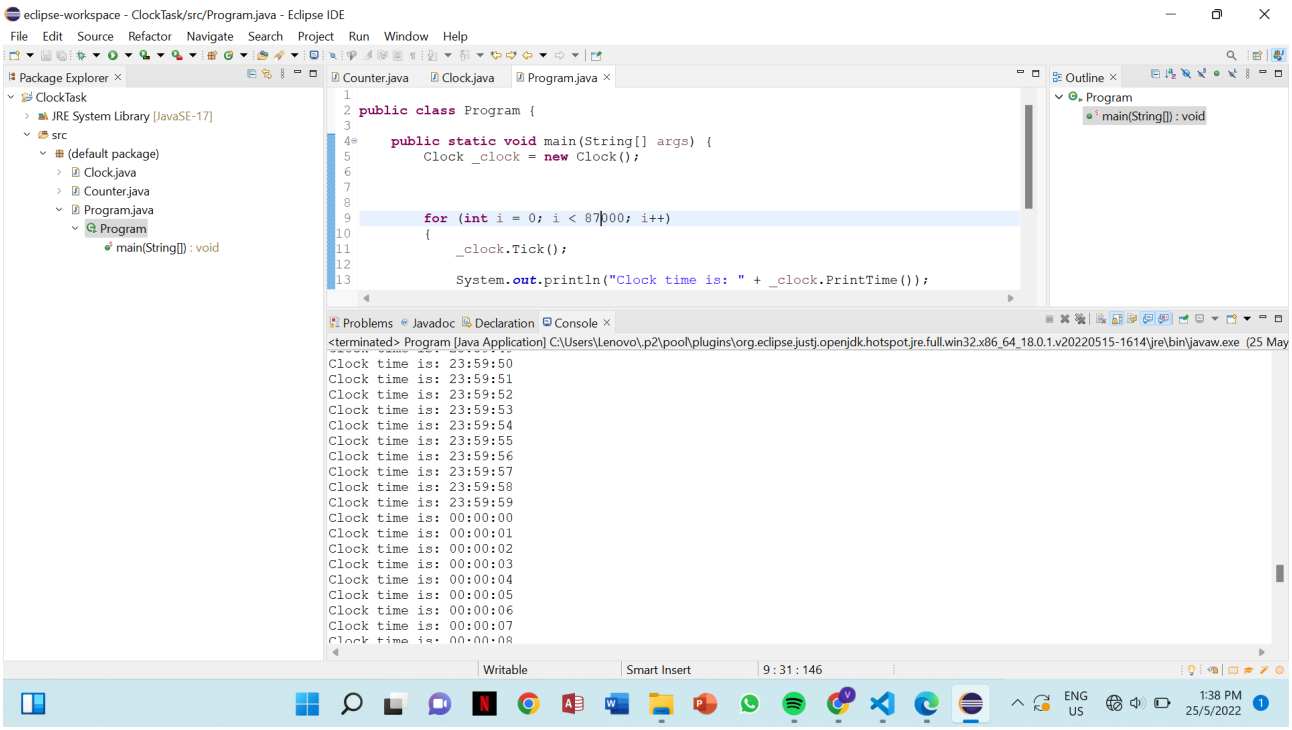
Tutor:
Jai Cornes

May 25, 2022

```
//Program.java
public class Program {

    public static void main(String[] args) {
        Clock _clock = new Clock();



        for (int i = 0; i < 87000; i++)
        {
            _clock.Tick();

            System.out.println("Clock time is: " + _clock.PrintTime());

        }


    }

}

//Counter.java
public class Counter {

    private int _count;
    private String _name;

    public String NameG()
    {
            return _name;
    }

    public void NameS(String _newsetName)

    {
        this._name = _newsetName;
    }


    public Counter(String name)
    {
        _name = name;
        _count = 0;
    }

    public void Increment()
    {
        _count++;
    }

    public void Reset()
    {
        _count = 0;
```

```
54
55      }
56
57
58      public int Ticks()
59      {
60
61              return _count;
62      }
63
64
65
66
67  }
68
69  //Clock.java
70  public class Clock {
71
72      private Counter _hours;
73      private Counter _minutes;
74      private Counter _seconds;
75
76
77      public Clock()
78      {
79          _hours = new Counter("hours");
80          _minutes = new Counter("minutes");
81          _seconds = new Counter("seconds");
82      }
83
84      public void Tick()
85      {
86          _seconds.Increment();
87          if (_seconds.Ticks() == 60)
88          {
89              _seconds.Reset();
90              _minutes.Increment();
91          }
92
93          if (_minutes.Ticks()== 60)
94          {
95              _minutes.Reset();
96              _hours.Increment();
97          }
98
99           if (_hours.Ticks() == 24)
100          {
101             _hours.Reset();
102          }
103
104     }
105
106
```

```
107
108     public void Reset()
109     {
110         _hours.Reset();
111         _minutes.Reset();
112         _seconds.Reset();
113     }
114
115
116     public String PrintTime()
117     {
118         return String.format("%02d:%02d:%02d",_hours.Ticks(),_minutes.Ticks(),
            ↪  _seconds.Ticks());
119
120     }
121
122
123
124 }
```


eclipse-workspace - ClockTask/src/Program.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
ClockTask
JRE System Library [JavaSE-17]
src
(default package)
Clock.java
Counter.java
Program.java
Program
main(String[]) : void
Counter.java
Clock.java
Program.java
public class Program {
public static void main(String[] args) {
Clock _clock = new Clock();
for (int i = 0; i < 87000; i++)
{
_clock.Tick();
System.out.println("Clock time is: " + _clock.PrintTime());
Outline
Program
main(String[]) : void
Problems Javadoc Declaration Console
<terminated> Program [Java Application] C:\Users\Lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_18.0.1.v20220515-1614\jre\bin\javaw.exe (25 May
Clock time is: 23:59:50
Clock time is: 23:59:51
Clock time is: 23:59:52
Clock time is: 23:59:53
Clock time is: 23:59:54
Clock time is: 23:59:55
Clock time is: 23:59:56
Clock time is: 23:59:57
Clock time is: 23:59:58
Clock time is: 23:59:59
Clock time is: 00:00:00
Clock time is: 00:00:01
Clock time is: 00:00:02
Clock time is: 00:00:03
Clock time is: 00:00:04
Clock time is: 00:00:05
Clock time is: 00:00:06
Clock time is: 00:00:07
Writable
Smart Insert
9 : 31 : 146
ENG
US
1:38 PM
25/5/2022

21 Task 11.2P: Learning Summary Report Draft

Reflection is an important part of learning. In this task you will start your learning summary report. This report gives you an opportunity to reflect on and explain how the work you have completed this semester demonstrates that you have met all of the unit's learning outcomes.

Date	Author	Comment
2022/05/31 17:36	Charlotte Pierce	Your definition of "abstraction" in your reflection is wrong.
2022/06/02 19:16	Vaissheenavi Prabakaran	Okay, I have uploaded the corrected document already, Charlotte. Thanks.

Swinburne University of Technology

Object Oriented Programming (2022 S1)

Doubtfire Submission

Task 11.2P: Learning Summary Report Draft

Submitted By:
Vaissheenavi Prabakaran
103508183
2022/06/02 19:17

Tutor:
Jai Cornes

June 2, 2022

COS20007

Object-Orientated Programming

Learning Summary Report

VAISSHEENAVI PRABAKARAN
103508183

Self-Assessment Details

The following checklists provide an overview of my self-assessment for this unit.

	Pass (D)	Credit (C)	Distinction (B)	High Distinction (A)
Self-Assessment	/			

Self-Assessment Statement

	Included
Learning Summary Report	/
Test is Complete in Doubtfire	/
C# programs that demonstrate coverage of core concepts	/
Explanation of OO principles	/
All Pass Tasks are Complete on Doubtfire	/

Minimum Pass Checklist

	Included
All Credit Tasks are Complete on Doubtfire	

Minimum Credit Checklist (in addition to Pass Checklist)

	Included
Distinction tasks (other than Custom Program) are Complete	
Custom program meets Distinction criteria & Interview booked	
Design report has UML diagrams and screenshots of program	

Minimum Distinction Checklist (in addition to Credit Checklist)

	Included
HD Project included	
Custom project meets HD requirements	

Minimum High Distinction Checklist (in addition to Distinction Checklist)

Declaration

I declare that this portfolio is my individual work. I have not copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part of this submission been written for me by another person.

Signature: **P.Vaissheenavi**

Portfolio Overview

This portfolio includes work that demonstrates that I have achieve all Unit Learning Outcomes for COS20007 Object-Orientated Programming to a **Pass** level.

[Provide a justification for why you should receive this grade… Write this for the assessment panel – tell them why you should get this grade.

For Pass: you need to indicate how you have demonstrated all Unit Learning Outcomes to a minimal level.
For Credit: you need to indicate how you have demonstrated all Unit Learning Outcomes to a good level.
For Distinction: you need to indicate how you have been able to apply all of the Unit Learning Outcomes in achieving the distinction tasks.
For High Distinction: you need to indicate how you have been able to extend beyond the material presented in the unit.

In this section, refer to the tasks you have completed. These will be attached by Doubtfire after this summary. Do not try to demonstrate the outcomes here, this is just a summary.

Think of this like a cover letter to a job application – here it is a cover letter to your grade application.]

Justification

I have demonstrated all the Unit Learning Outcomes by having a solid understanding on the core principles of the object orientated programming paradigm which are encapsulation, abstraction, polymorphism, and inheritance. Some of the other contents that I learned in this unit includes the use of C# programming language, class libraries to develop OOP programs, framework classes to reuse the class to save time and doing NUnit tests to ensure there are no mistakes in the codes. Moreover, for some Pass tasks and the Week 8 Test, I learned how to properly construct a good UML diagram and textual descriptions that represents the program developed for the tasks accordingly. Along the way of this unit, I have designed, developed, tested, and debugged the tasks along with my tutor. My tutor gave me meaningful feedbacks which then really helped me to consolidate my understanding on some OOP good coding practices and concepts. Constantly working on the OOP tasks for 12 weeks strengthen my knowledge on the concepts and ‘when’/ ’how’ to use them efficiently. I like how this unit was designed as there were tasks created based on the core concepts. This helped me to apply the concepts that I learned over time from the online class/ tutorial session into the tasks assigned. On the side note, I have also referred to the unit online to demonstrate the learning outcomes. Now, regarding the grade, I should get a Pass because I have completed all the Pass tasks required. However, I am still trying my best on working with the credit tasks. I will upload what I have done for the credit tasks to Doubtfire before the deadline (5th June 2022) as you could assess my progress in the credit tasks too. Overall, this is an interesting unit and I can’t wait to implement what I have

learned from this unit in the future.

Reflection

The most important things I learnt:

The most important things I learnt is that encapsulation, abstraction, polymorphism, and inheritance. Encapsulation is the ability of an object to hide its data when it is not necessary to the users. Inheritance gives us an ability to create a new class from an existing class. Abstraction is used to hide the implementation information and only showing the users the required function. This is to hide unwanted information from the users and only shows the relevant information. Polymorphism refers to the idea of being able to access objects of many types using the same interface. This interface can be implements in a variety of ways by each type.

Referencing: https://www.programiz.com/csharp-programming/inheritance#:~:text=In%20C%23%2C%20inheritance%20allows%20us,derived%20class%20(child%20or%20subclass)

https://www.techopedia.com/definition/3787/encapsulation-c#:~:text=Encapsulation%2C%20in%20the%20context%20of,a%20single%20unit%20or%20object.

The things that helped me most were:

The things that helped me the most were the helpdesk sessions and my tutor. For example, when I had trouble understanding the tasks, I would often go to the helpdesks to ask for further clarifications from different tutors. All the tutors were so helpful, and they also ensured that I have completely understood the tasks before doing. Besides, the lecture recordings and materials were also helpful along the way.

I found the following topics particularly challenging:

At first, I found polymorphism was challenging for me grasp the concept of it. I managed to overcome it by doing the tasks. Besides, the NUnit Tests were also challenging for me but with the help of my tutor and helpdesk sessions, I managed to learn a lot from them.

I found the following topics particularly interesting:

I found inheritance interesting as we can reuse the codes for different classes by just inheriting the properties.

I feel I learnt these topics, concepts, and/or tools really well:

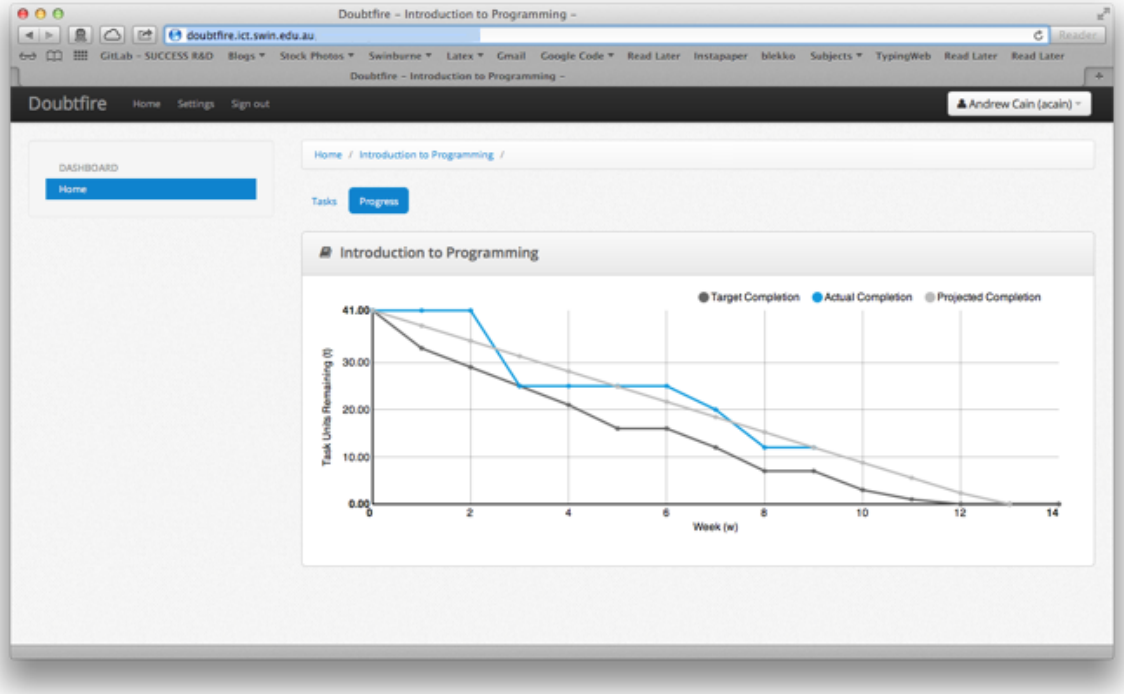
I feel I learned encapsulation and inheritance well. I perfectly understand how both encapsulation and inheritance. Moreover, I am able to confidently code the parts where this is required because I can understand it from the requirements given.

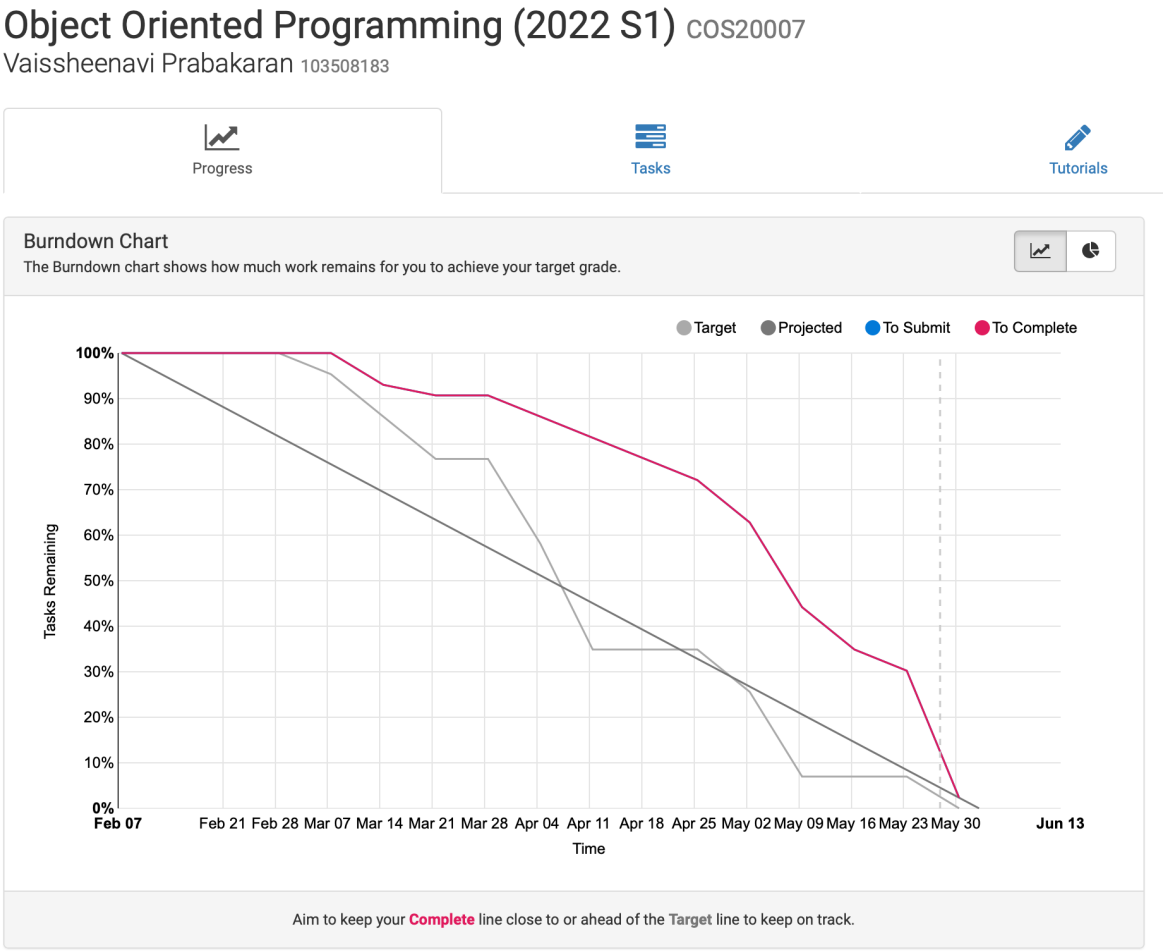
I still need to work on the following areas:

I still need to work on polymorphism as I still need time to wrap my head around this concept. I know how it works but I would like to be more confident while applying it.

My progress in this unit was …:

[Include a screenshot of your **progress graph** from **Doubtfire**, and comment on what happened from your perspective… what does the graph say about how you approached the unit? (Login to Doubtfire to get your graph https://doubtfire.ict.swin.edu.au)]





This unit will help me in the future:

The importance of encapsulation, abstraction, polymorphism, inheritance and all the topics that I have learnt in this unit, I believe that it will be valuable in the future. This is because this unit has successfully provided me with strong basics of object orientated programming and I am sure that I will be able to put all the knowledge I have learnt from this unit to a good use in the future.

If I did this unit again I would do the following things differently:

If I did this unit again, I would learn C# prior to learning this unit as most of the tasks (all Pass tasks) is in C#.

Other…:

If I have learnt C# early through the Internet, it would have been an easier process of understanding the tasks assigned. Fortunately, as the times goes, I got the hang of it and hopefully, I can do better next time.