SIT771

Object-oriented Development

Learning Summary Report

Yiyang Hou
220570074

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	✓
Pass tasks complete	✓

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	

Minimum Distinction Checklist (in addition to Credit Checklist)

	Included
Something Awesome 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: **Yiyang Hou**

Portfolio Overview

This portfolio includes work that demonstrates that I have achieve all Unit Learning Outcomes for SIT771 Object-oriented Development to a **Credit** level.

[After progressing through the unit, I have learnt the principals of object-oriented programming and their application, how to design and build my own program as well as how to evaluate a piece of given code.

In credit task 4.2C, I was able to completed the task by evaluating and refactoring the target code. That demonstrates that I have achieved the unit learning outcome of Evaluate Code.

The series of robot dodge programs gave me the opportunity to practice my coding skills and have demonstrated my achievement of the unit learning outcome of Build Programs. In addition to that, they also helped with my understanding of how to read, evaluate and communicate solution structures, it demonstrates my achievement of the unit learning outcome of Design.

The concept visualisation tasks helped me with consolidating my understanding of the key principals of object-oriented programming and have demonstrated my achievement of the unit learning outcome of Principals.

Finally, I have completed all the tasks by providing the relevant evidence such as the code, screenshots and diagrams, that demonstrates that I have achieved the unit learning outcome of Justify.

I was able to complete all the tasks up to the level of credit with relative ease, in conjunction with the forementioned justifications, I believe that I have demonstrated that I have reached a credit level.]

Reflection

The most important things I learnt:

[The most important thing that I have learnt must be the key principals of object-oriented programming. Those notions were difficult to understand at first, but became clearer with proper practice. Once I got grasp on those principals all the tasks become easier for me, even learning another language becomes easy.]

The things that helped me most were:

[The thing that helped me the most was the syntax guides. A lot of the times, I know how to approach the objective of a task, but stuck with getting the syntax wrong, the syntax guides have always been very clear and helpful.]

I found the following topics particularly challenging:

[The concept of object and class was very challenging for me, as this is the foundation to object-oriented programming, so I struggled with the first several tasks because of it. As I do some extra studies, I have finally understood the concept.]

I found the following topics particularly interesting:

[I found every topic interesting, because everything I have learnt in this unit is very useful and practical. I can visualise that there is a place for every topic covered in this unit in an actual program.]

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

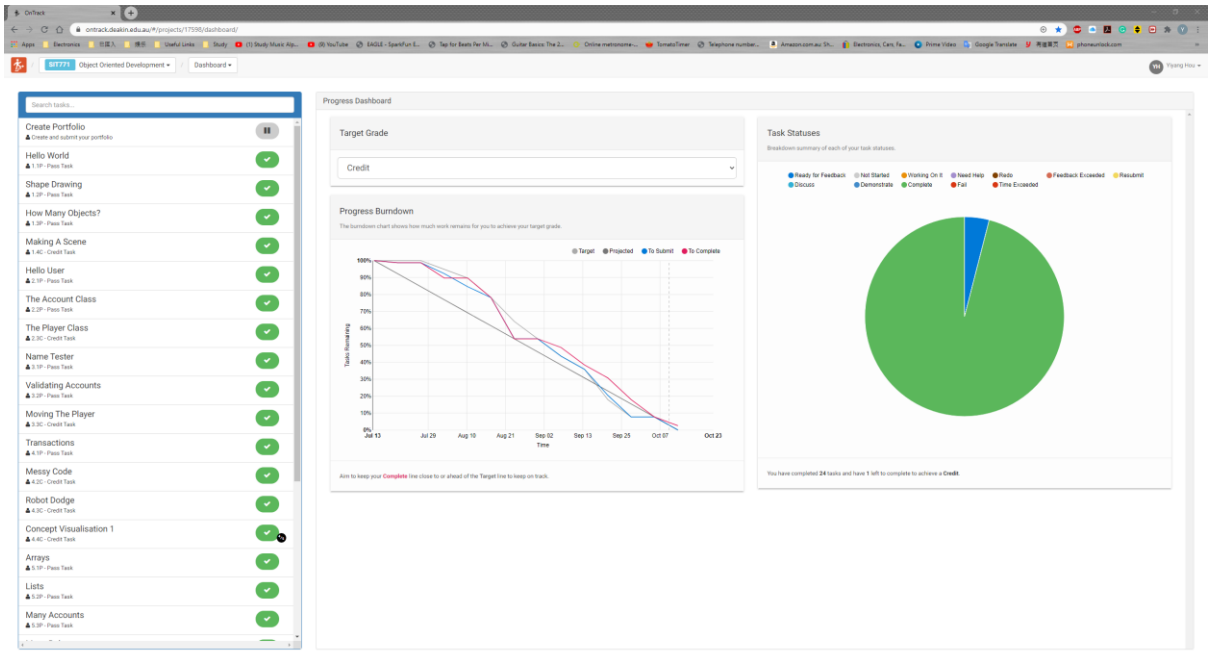
[I learnt the key principles of object-oriented programming really well, because every time I stuck in a task, I go back to these principles, they can always help me to solve the problems I had at the time.]

I still need to work on the following areas:

[I still need to work on my design skills. In my attempt of completing the distinction tasks, I found that designing my own solution is my weakness. I will need to work on that aspect going forward.]

My progress in this unit was …:

[According to the graph, I was a little behind the schedule at a point, but I quickly picked up the pace and managed to stay on the track.



This unit will help me in the future:

[I believe that everything I learnt from this unit will be valuable for me whether for my studies or my career in the industry. Because this unit introduces the world of programming to me and be able to program and understanding the related concepts are the building blocks for the software industry. So I will be applying the knowledge and skills that I have learnt from this unit in my future studies as well as my career.]

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

[I tried to do my best not to procrastinate and leave thing to the last moment, but because the online learning mode is very new to me, I could have done better in terms of progress management and eliminating distractions. With the experience of this trimester, I will confidently apply the learning skills that I have learnt this trimester to the future studies, whether online or on campus.]

Deakin University

Object Oriented Development

Yiyang Hou

Portfolio Submission

Submitted By:
Yiyang Hou
houyiy

Tutor:
Chetan Arora

October 9, 2020

Contents

2 Overall Task Status

Task	Status	Times Assessed
Hello World	Complete	1
Shape Drawing	Complete	1
How Many Objects?	Complete	1
Making A Scene	Complete	1
Help Others	Not Started	
Hello User	Complete	1
The Account Class	Complete	1
The Player Class	Complete	2
Name Tester	Complete	1
Validating Accounts	Complete	2
Moving The Player	Complete	1
Transactions	Complete	1
Messy Code	Complete	1
Robot Dodge	Complete	1
Concept Visualisation 1	Complete	1
Arrays	Complete	1
Lists	Complete	1
Many Accounts	Complete	1
Many Robots	Complete	1
Document Design	Complete	1
Robot Dodge Changes	Not Started	0
Custom Program Pitch	Not Started	0
Abstract Transactions	Complete	1
Different Robots	Complete	1
Custom Program Code	Not Started	
Something Awesome	Not Started	
Interview - Code Explanation	Complete	2
Concept Visualisation 2	Complete	1
Another Language	Complete	1
Draft Learning Summary Report	Ready to Mark	0

3 Learning Outcomes

3.1 Evaluate Code

Evaluate simple program code for correct use of coding conventions, and use code tracing and debugging techniques to identify and correct issues.

Task	Rating	Status	Times Assessed
How Many Objects?	♦♦◇◇◇	Complete	1
The Player Class	♦◇◇◇◇	Complete	2
Messy Code	♦♦♦◇◇	Complete	1
Document Design	♦♦◇◇◇	Complete	1
Interview - Code Explanation	♦♦♦♦♦	Complete	2
Draft Learning Summary Report	♦◇◇◇◇	Ready to Mark	0

3.2 Principles

Apply and explain the principles of object oriented programming including abstraction, encapsulation, inheritance and polymorphism.

Task	Rating	Status	Times Assessed
How Many Objects?	♦♦◇◇◇	Complete	1
The Account Class	♦♦◇◇◇	Complete	1
Concept Visualisation 1	♦♦♦◇◇	Complete	1
Interview - Code Explanation	♦♦♦♦♦	Complete	2
Concept Visualisation 2	♦♦♦◇◇	Complete	1
Draft Learning Summary Report	♦◇◇◇◇	Ready to Mark	0

3.3 Build Programs

Implement, and test small object oriented programs that conform to planned system structures and requirements

Task	Rating	Status	Times Assessed
Hello World	♦♦◇◇◇	Complete	1
Shape Drawing	♦♦◇◇◇	Complete	1
Making A Scene	♦♦♦◇◇	Complete	1
Hello User	♦♦◇◇◇	Complete	1
The Account Class	♦♦◇◇◇	Complete	1
Name Tester	♦♦◇◇◇	Complete	1
Validating Accounts	♦♦◇◇◇	Complete	2
Moving The Player	♦♦◇◇◇	Complete	1
Transactions	♦♦♦◇◇	Complete	1
Messy Code	♦◇◇◇◇	Complete	1
Robot Dodge	♦♦♦◇◇	Complete	1
Arrays	♦♦◇◇◇	Complete	1
Lists	♦♦◇◇◇	Complete	1
Many Accounts	♦♦♦◇◇	Complete	1
Many Robots	♦♦♦◇◇	Complete	1
Abstract Transactions	♦♦♦◇◇	Complete	1
Different Robots	♦♦♦◇◇	Complete	1
Another Language	♦♦♦◇◇	Complete	1
Draft Learning Summary Report	♦◇◇◇◇	Ready to Mark	0

3.4 Design

Design, communicate, and evaluate solution structures using appropriate diagrams and textual descriptions.

Task	Rating	Status	Times Assessed
The Account Class	♦◊◊◊	Complete	1
The Player Class	♦◊◊◊	Complete	2
Validating Accounts	♦◊◊◊	Complete	2
Moving The Player	♦◊◊◊	Complete	1
Transactions	♦◊◊◊	Complete	1
Robot Dodge	♦◊◊◊	Complete	1
Many Accounts	♦◊◊◊	Complete	1
Many Robots	♦◊◊◊	Complete	1
Document Design	♦♦♦♦◊	Complete	1
Abstract Transactions	♦♦◊◊◊	Complete	1
Different Robots	♦♦◊◊◊	Complete	1
Draft Learning Summary Report	♦◊◊◊	Ready to Mark	0

3.5 Justify

Justify meeting specified outcomes through providing relevant evidence and critiquing the quality of that evidence against given criteria.

Task	Rating	Status	Times Assessed
Hello World	♦◊◊◊	Complete	1
Shape Drawing	♦◊◊◊	Complete	1
How Many Objects?	♦◊◊◊	Complete	1
Making A Scene	♦◊◊◊	Complete	1
Hello User	♦◊◊◊	Complete	1
The Account Class	♦◊◊◊	Complete	1
The Player Class	♦◊◊◊	Complete	2
Name Tester	♦◊◊◊	Complete	1
Validating Accounts	♦◊◊◊	Complete	2
Moving The Player	♦◊◊◊	Complete	1
Transactions	♦◊◊◊	Complete	1
Messy Code	♦◊◊◊	Complete	1
Robot Dodge	♦◊◊◊	Complete	1
Concept Visualisation 1	♦◊◊◊	Complete	1
Arrays	♦◊◊◊	Complete	1
Lists	♦◊◊◊	Complete	1
Many Accounts	♦◊◊◊	Complete	1
Many Robots	♦◊◊◊	Complete	1
Document Design	♦◊◊◊	Complete	1
Abstract Transactions	♦◊◊◊	Complete	1
Different Robots	♦◊◊◊	Complete	1
Interview - Code Explanation	♦♦♦♦♦	Complete	2
Concept Visualisation 2	♦◊◊◊	Complete	1
Another Language	♦◊◊◊	Complete	1
Draft Learning Summary Report	♦♦♦♦♦	Ready to Mark	0

4 Hello World

Get started by testing the tools you have installed.

Outcome	Weight
Build Programs	♦♦◊◊◊

This task helped me to set up the tools I need for the upcoming tasks and gave me a taste of building an actual program.

Outcome	Weight
Justify	♦◊◊◊◊

This task helped me to set up the tools I need for the upcoming tasks and gave me a taste of building an actual program.

Date	Author	Comment
2020/07/16 20:55	Yiyang Hou	Working On It
2020/07/17 11:16	Yiyang Hou	Ready to Mark
2020/07/17 14:07	Chetan Arora	Well done.
2020/07/17 14:07	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Hello World

Submitted By:
Yiyang Hou
houyiy
2020/07/17 11:16

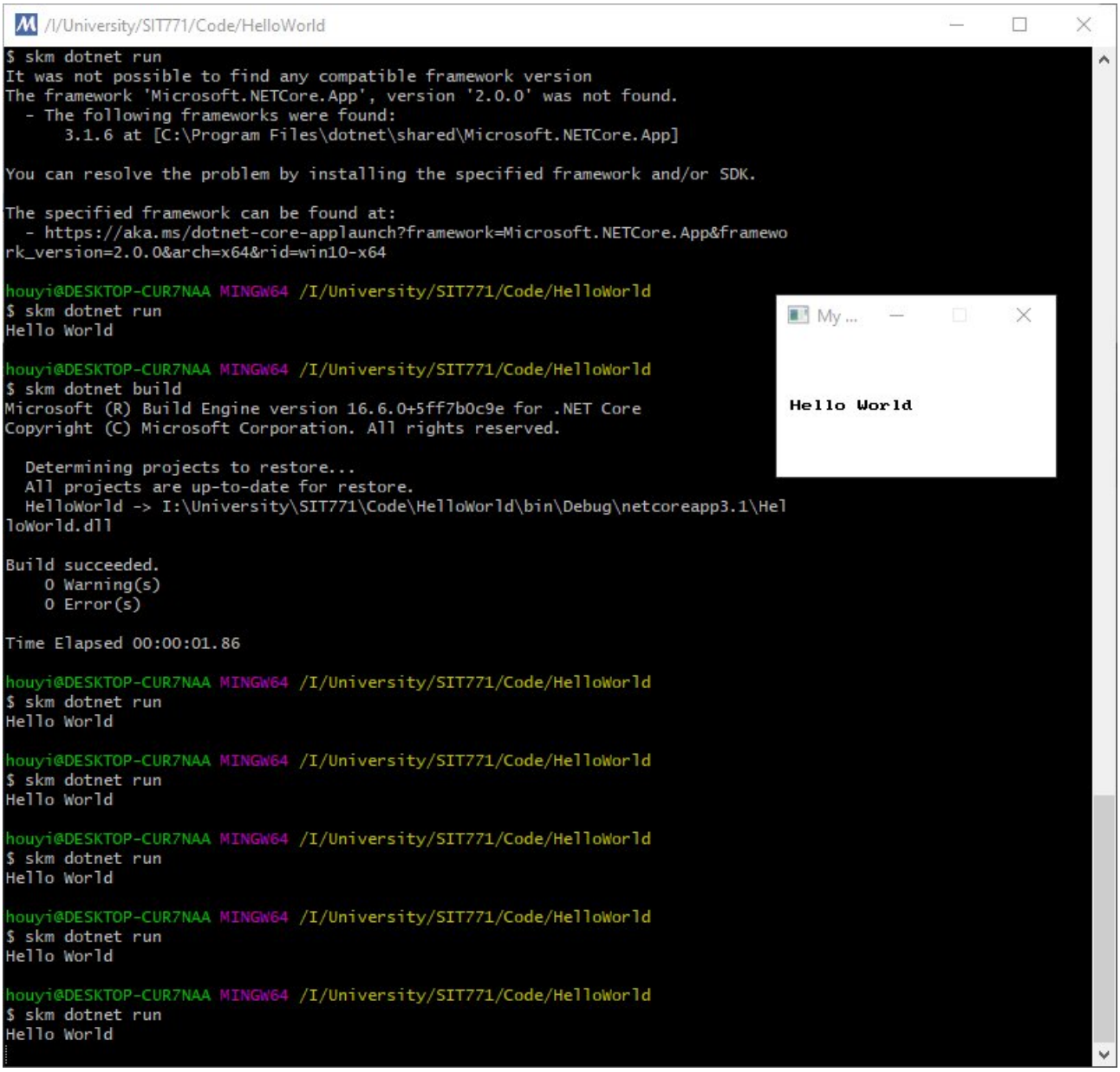
Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦◊◊◊
Justify	♦◊◊◊◊

This task helped me to set up the tools I need for the upcoming tasks and gave me a taste of building an actual program.

July 17, 2020

```
using System;
using SplashKitSDK;

public class Program
{
    public static void Main()
    {
        Console.WriteLine("Hello World");

        Window w = new Window("My First Program", 200, 100);
        w.DrawText("Hello World", Color.Black, 10, 45);
        w.Refresh(60);
        SplashKit.Delay(5000);
    }
}
```


/I/University/SIT771/Code/HelloWorld
$ skm dotnet run
It was not possible to find any compatible framework version
The framework 'Microsoft.NETCore.App', version '2.0.0' was not found.
- The following frameworks were found:
3.1.6 at [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
You can resolve the problem by installing the specified framework and/or SDK.
The specified framework can be found at:
- https://aka.ms/dotnet-core-applaunch?framework=Microsoft.NETCore.App&framework_version=2.0.0&arch=x64&rid=win10-x64
houyi@DESKTOP-CUR7NAA MINGW64 /I/University/SIT771/Code/HelloWorld
$ skm dotnet run
Hello World
houyi@DESKTOP-CUR7NAA MINGW64 /I/University/SIT771/Code/HelloWorld
$ skm dotnet build
Microsoft (R) Build Engine version 16.6.0+5ff7b0c9e for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.
Determining projects to restore...
All projects are up-to-date for restore.
HelloWorld -> I:\University\SIT771\Code\HelloWorld\bin\Debug\netcoreapp3.1\HelloWorld.dll
Build succeeded.
0 Warning(s)
0 Error(s)
Time Elapsed 00:00:01.86
houyi@DESKTOP-CUR7NAA MINGW64 /I/University/SIT771/Code/HelloWorld
$ skm dotnet run
Hello World
houyi@DESKTOP-CUR7NAA MINGW64 /I/University/SIT771/Code/HelloWorld
$ skm dotnet run
Hello World
houyi@DESKTOP-CUR7NAA MINGW64 /I/University/SIT771/Code/HelloWorld
$ skm dotnet run
Hello World
houyi@DESKTOP-CUR7NAA MINGW64 /I/University/SIT771/Code/HelloWorld
$ skm dotnet run
Hello World
houyi@DESKTOP-CUR7NAA MINGW64 /I/University/SIT771/Code/HelloWorld
$ skm dotnet run
Hello World
My ...
Hello World

5 Shape Drawing

Create and draw shapes to the screen using SplashKit!

Outcome	Weight
Build Programs	♦♦◊◊◊

This task demonstrates the implementation and testing of a small object-oriented program and the program meets the desired outcomes proven by the evidence.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the implementation and testing of a small object-oriented program and the program meets the desired outcomes proven by the evidence.

Date	Author	Comment
2020/07/17 11:35	Yiyang Hou	Working On It
2020/07/29 23:42	Yiyang Hou	Ready to Mark
2020/07/30 10:41	Chetan Arora	Very nice drawing
2020/07/30 10:41	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Shape Drawing

Submitted By:
Yiyang Hou
houyiy
2020/07/29 23:42

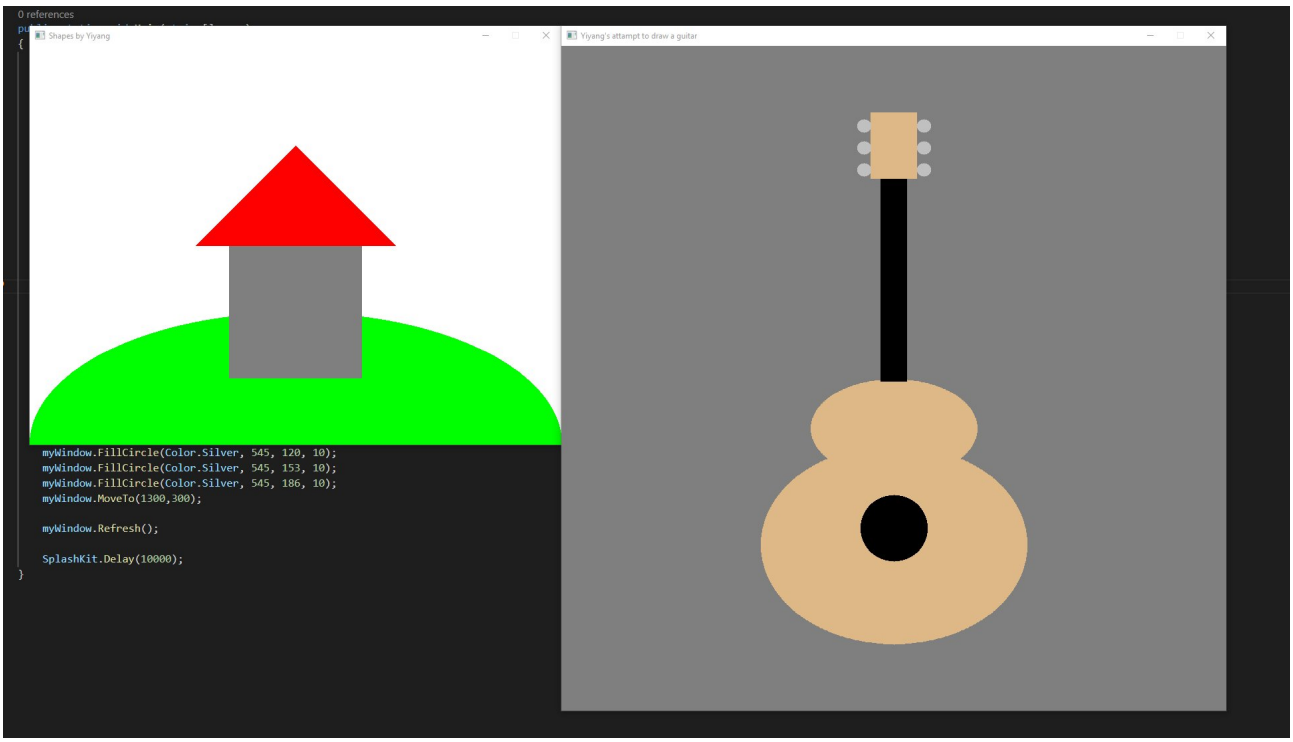
Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦◊◊◊
Justify	♦◊◊◊◊

This task demonstrates the implementation and testing of a small object-oriented program and the program meets the desired outcomes proven by the evidence.

July 29, 2020

```
using System;
using SplashKitSDK;

public class Program
{
    public static void Main(string[] args)
    {
        // House drawing.

        Window shapesWindow = new Window("Shapes by Yiyang", 800, 600);

        shapesWindow.Clear(Color.White);
        shapesWindow.FillEllipse(Color.BrightGreen, 0, 400, 800, 400);
        shapesWindow.FillRectangle(Color.Gray, 300, 300, 200, 200);
        shapesWindow.FillTriangle(Color.Red, 250, 300, 400, 150, 550, 300);
        shapesWindow.MoveTo(500,300);
        shapesWindow.Refresh();

        // My own shapes.

        Window myWindow = new Window("Yiyang's attampt to draw a guitar", 1000,
        ↪  1000);

        myWindow.Clear(Color.Gray);
        myWindow.FillEllipse(Color.BurlyWood, 300, 600, 400, 300);
        ↪          // Lower part of the body.
        myWindow.FillEllipse(Color.BurlyWood, 375, 500, 250, 150);
        ↪          // Higher part of the body.
        myWindow.FillCircle(Color.Black, 500, 725, 50);
        ↪          // Sound Hole.
        myWindow.FillRectangle(Color.Black, 480, 155, 40, 350);
        ↪          // FretBoard.
        myWindow.FillRectangle(Color.BurlyWood, 465, 100, 70, 100);
        ↪          // Head stock.
        // Left side tuning pegs.
        myWindow.FillCircle(Color.Silver, 455, 120, 10);
        myWindow.FillCircle(Color.Silver, 455, 153, 10);
        myWindow.FillCircle(Color.Silver, 455, 186, 10);
        // Right side tuning pegs.
        myWindow.FillCircle(Color.Silver, 545, 120, 10);
        myWindow.FillCircle(Color.Silver, 545, 153, 10);
        myWindow.FillCircle(Color.Silver, 545, 186, 10);
        myWindow.MoveTo(1300,300);

        myWindow.Refresh();

        SplashKit.Delay(10000);
    }
}
```


0 references
Shapes by Yiyang
Yiyang's attampt to draw a guitar
myWindow.FillCircle(Color.Silver, 545, 120, 10);
myWindow.FillCircle(Color.Silver, 545, 153, 10);
myWindow.FillCircle(Color.Silver, 545, 186, 10);
myWindow.MoveTo(1300,300);
myWindow.Refresh();
SplashKit.Delay(10000);
}

6 How Many Objects?

Read the supplied code samples and let us know how many objects are created in each.

Outcome	Weight
Evaluate Code	♦♦◊◊◊

This task demonstrates code evaluation and explaining some of the principles of object-oriented programming and the relevant evidence is provided.

Outcome	Weight
Principles	♦♦◊◊◊

This task demonstrates code evaluation and explaining some of the principles of object-oriented programming and the relevant evidence is provided.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates code evaluation and explaining some of the principles of object-oriented programming and the relevant evidence is provided.

Date	Author	Comment
2020/07/30 00:28	Yiyang Hou	Ready to Mark
2020/07/30 10:43	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

How Many Objects?

Submitted By:
Yiyang Hou
houyiy
2020/07/30 00:28

Tutor:
Chetan Arora

Outcome	**Weight**
Evaluate Code	♦♦◊◊◊
Principles	♦♦◊◊◊
Justify	♦◊◊◊◊

This task demonstrates code evaluation and explaining some of the principles of object-oriented programming and the relevant evidence is provided.

July 30, 2020

Answers for 1.3P How Many Objects?

Student Name: Yiyang Hou
Student ID: 220570074

Question 1: How many of each kind of objects are created in this code?

Class	Number of Objects
Window	2
Bitmap	1
Sound Effect	1
Font	0

Question 2: What are the details of the different windows? Complete the following table. For Color Shown, indicate the color that the window was cleared to.

Window Title	Width	Height	Color Shown
“Hello World”	800	600	Blue
“Another Window”	300	300	Green

Question 3: How are the variables and Window objects connected? Which variables refer to which objects?

Window Title	Number of Variables that Refer to this Object?	Variable Names (comma separate if multiple)
“Hello World”	2	helloWindow, yetAnotherWindow
“Another Window”	1	anotherWindow

Question 4: How many times is the Window object with the title “Hello World” told to do something? Copy in the lines of code that get this Window object to do something.

1. helloWindow.MoveTo(0, 0);
2. yetAnotherWindow.Clear(Color.Blue);
3. yetAnotherWindow.Refresh(60);
4. helloWindow.DrawBitmap(pegasi, 10, 50);
5. helloWindow.Refresh(60);
   This Window object is told to do things for 5 times in this program.

Question 5: How could you create another Bitmap object? One that loads a “Hello.png” image?

Bitmap hello = new Bitmap(“Hello”, “Hello.png”);

Question 6: How could you create another variable that will also refer to the “Hello.png” image you loaded in Question 5?

Bitmap anotherHello = hello;

7 Making A Scene

Using all you have learnt so far, create a scene/comic/story using shape drawing in SplashKit.

Outcome	Weight
Build Programs	♦♦♦◊◊

This task demonstrates the ability to build a program with evidence provided to justify.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the ability to build a program with evidence provided to justify.

Date	Author	Comment
2020/08/09 19:17	Yiyang Hou	Working On It
2020/08/09 19:57	Yiyang Hou	Personal difficulties
2020/08/10 00:06	Yiyang Hou	Ready to Mark
2020/08/11 16:34	Chetan Arora	Can you share the link to your scene? Upload it to youtube or Deakin Air
2020/08/11 16:34	Chetan Arora	Demonstrate
2020/08/11 17:37	Yiyang Hou	Sure I can, still trying to work out how to do the recording though, I'll share the link once I get it working. Thanks
2020/08/11 17:42	Chetan Arora	:+1:
2020/08/11 19:05	Yiyang Hou	https://video.deakin.edu.au/media/t/0_mhyr4p65
2020/08/11 19:06	Yiyang Hou	This is the link, sorry about the delay, took me a while to finally work it out how to upload the vid.
2020/08/14 16:57	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Making A Scene

Submitted By:
Yiyang Hou
houyiy
2020/08/10 00:06

Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦♦◊◊
Justify	♦◊◊◊◊

This task demonstrates the ability to build a program with evidence provided to justify.

August 10, 2020

```
using System;
using SplashKitSDK;

public class Program
{
    public static void Main()
    {
        Window myScene = new Window("Dragon Ball", 500, 360);
        myScene.Clear(Color.White);
        myScene.Refresh(60);

        Bitmap fight = new Bitmap("fight", "fight.jpg");

        myScene.DrawBitmap(fight, 0, 0);
        myScene.Refresh(60);
        SoundEffect fightSound = new SoundEffect("fight5s","fight5s.mp3");
        fightSound.Play();
        SplashKit.Delay(5000);


        myScene.Clear(Color.White);
        myScene.Refresh(60);
        Bitmap fight2 = new Bitmap("fight2", "fight2.jpg");
        myScene.Resize(500, 281);
        myScene.DrawBitmap(fight2, 0, 0);
        myScene.Refresh(60);
        SoundEffect fightSound2 = new SoundEffect("fight3s","fight3s.mp3");
        fightSound2.Play();
        SplashKit.Delay(5000);

        myScene.Clear(Color.White);
        myScene.Refresh(60);
        Bitmap superSaiyan = new Bitmap("SuperSaiyan", "Su.jpg");
        myScene.MoveTo(950,300);
        myScene.Resize(640,940);
        myScene.DrawBitmap(superSaiyan, 0, 0);
        myScene.Refresh(60);
        SoundEffect superSaiyanSound = new
        ↪  SoundEffect("supersaiyan","supersaiyan.mp3");
        superSaiyanSound.Play();
        SplashKit.Delay(5000);

        myScene.Clear(Color.White);
        myScene.Refresh(60);
        Bitmap ha = new Bitmap("Ha", "Ha.jpg");
        myScene.MoveTo(950,300);
        myScene.Resize(500,635);
        myScene.DrawBitmap(ha, 0, 0);
        myScene.Refresh(60);
        SoundEffect haSound = new SoundEffect("ha","ha.mp3");
        haSound.Play();
        SplashKit.Delay(5000);

```

```
53
54
55      }
56  }
```


Dragon Ball

8 Hello User

Create your own version of the Hello User program with some additional requirements.

Outcome	Weight
Build Programs	♦♦◊◊◊

This task demonstrates the development of a small program and relevant evidence is provided.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the development of a small program and relevant evidence is provided.

Date	Author	Comment
2020/08/02 20:48	Yiyang Hou	Ready to Mark
2020/08/03 02:50	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Hello User

Submitted By:
Yiyang Hou
houyiy
2020/08/02 20:48

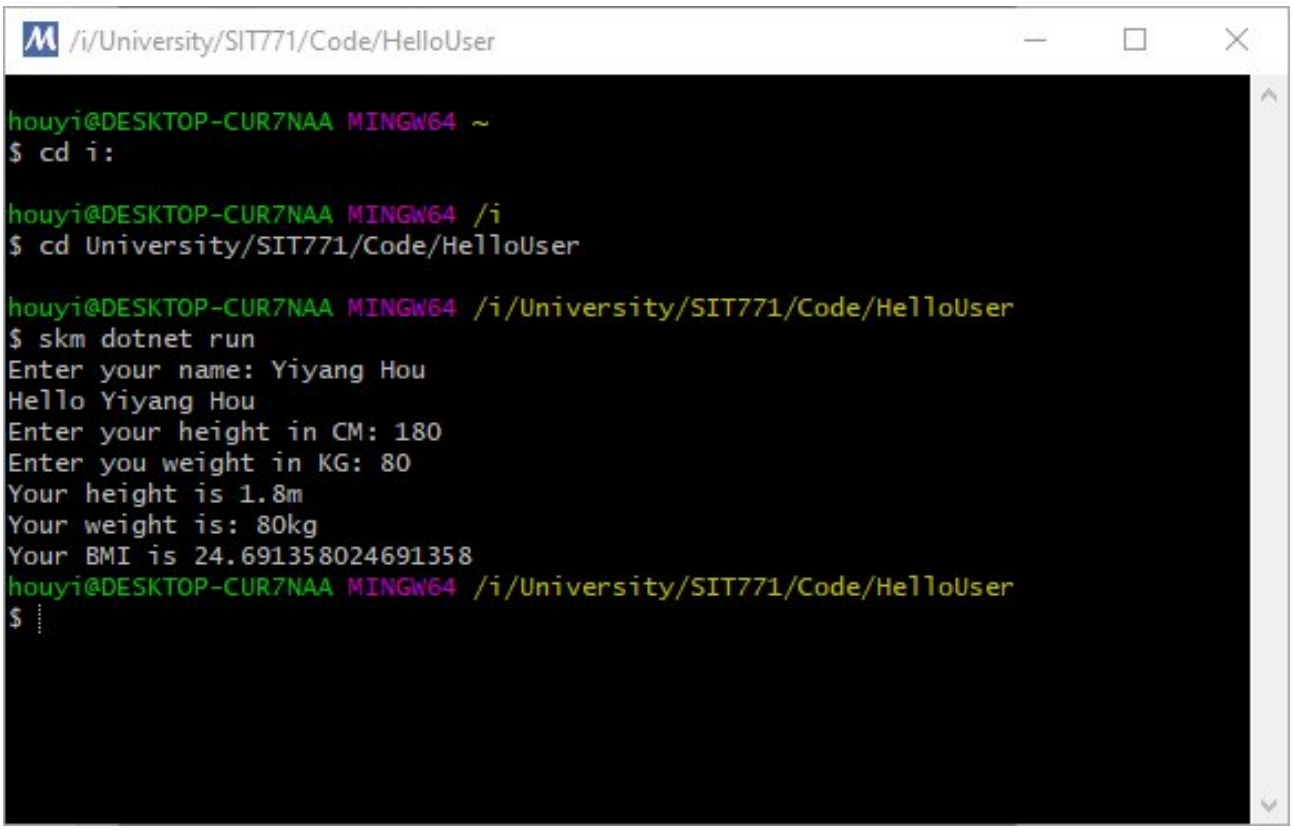
Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦◊◊◊
Justify	♦◊◊◊◊

This task demonstrates the development of a small program and relevant evidence is provided.

August 2, 2020

```
using System;
using SplashKitSDK;

public class Program
{
    public static void Main()
    { // Variables declaration
        string name, inputText;
        int heightInCM;
        double weightInKG ,heightInMeters, bmi;

        //Read in the user's name
        Console.Write("Enter your name: ");
        name = Console.ReadLine();

        Console.WriteLine($"Hello {name}");

        //Read their height and weight
        Console.Write("Enter your height in CM: ");
        inputText = Console.ReadLine();
        heightInCM = Convert.ToInt32(inputText);
        heightInMeters = heightInCM/100.0;
        Console.Write("Enter you weight in KG: ");
        inputText = Console.ReadLine();
        weightInKG = Convert.ToDouble(inputText);

        Console.WriteLine($"Your height is {heightInMeters}m");
        Console.WriteLine($"Your weight is: {weightInKG}kg");

        // Calculate the BMI
        bmi = weightInKG / Math.Pow(heightInMeters,2);  //BMI = kg/m^2
        Console.Write($"Your BMI is {bmi}");

    }
}
```


/i/University/SIT771/Code/HelloUser
houyi@DESKTOP-CUR7NAA MINGW64 ~
$ cd i:
houyi@DESKTOP-CUR7NAA MINGW64 /i
$ cd University/SIT771/Code/HelloUser
houyi@DESKTOP-CUR7NAA MINGW64 /i/University/SIT771/Code/HelloUser
$ skm dotnet run
Enter your name: Yiyang Hou
Hello Yiyang Hou
Enter your height in CM: 180
Enter you weight in KG: 80
Your height is 1.8m
Your weight is: 80kg
Your BMI is 24.691358024691358
houyi@DESKTOP-CUR7NAA MINGW64 /i/University/SIT771/Code/HelloUser
$

9 The Account Class

Create a custom class to represent a product in an account management system.

Outcome	Weight
Principles	♦♦◊◊◊

This task demonstrates the principle and the approach to build a program. This task also demonstrates the evaluation of the solution structure and the justification of the outcome.

Outcome	Weight
Build Programs	♦♦◊◊◊

This task demonstrates the principle and the approach to build a program. This task also demonstrates the evaluation of the solution structure and the justification of the outcome.

Outcome	Weight
Design	♦◊◊◊◊

This task demonstrates the principle and the approach to build a program. This task also demonstrates the evaluation of the solution structure and the justification of the outcome.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the principle and the approach to build a program. This task also demonstrates the evaluation of the solution structure and the justification of the outcome.

Date	Author	Comment
2020/08/03 17:58	Yiyang Hou	Ready to Mark
2020/08/03 23:26	Chetan Arora	Can you please answer all the discussion questions in the task sheet?
2020/08/03 23:26	Chetan Arora	Discuss
2020/08/05 00:09	Yiyang Hou	Yes I can do that, do I just answer the questions in the comment here?
2020/08/05 00:09	Yiyang Hou	1.Classes create objects by declaring constructors. Properties can provide objects the access to the fields. 2.When creating a class, the required elements are the declarations of fields, methods, constructors and properties. The static fields and methods are on the class itself, the constructors are used to initialize new objects, the instance fields are the knowledge for objects created from the class and properties are used to control the accessibility of these knowledge and the instance methods can be called by the objects to perform tasks.3.The fields can give knowledge to objects through getting accessed by properties.4.The variable that contains the reference to a specific object created from the class can call the methods through dot operator, by providing all required parameters, the task that the method is specified to do will perform.
2020/08/05 11:36	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

The Account Class

Submitted By:
Yiyang Hou
houyiy
2020/08/03 17:58

Tutor:
Chetan Arora

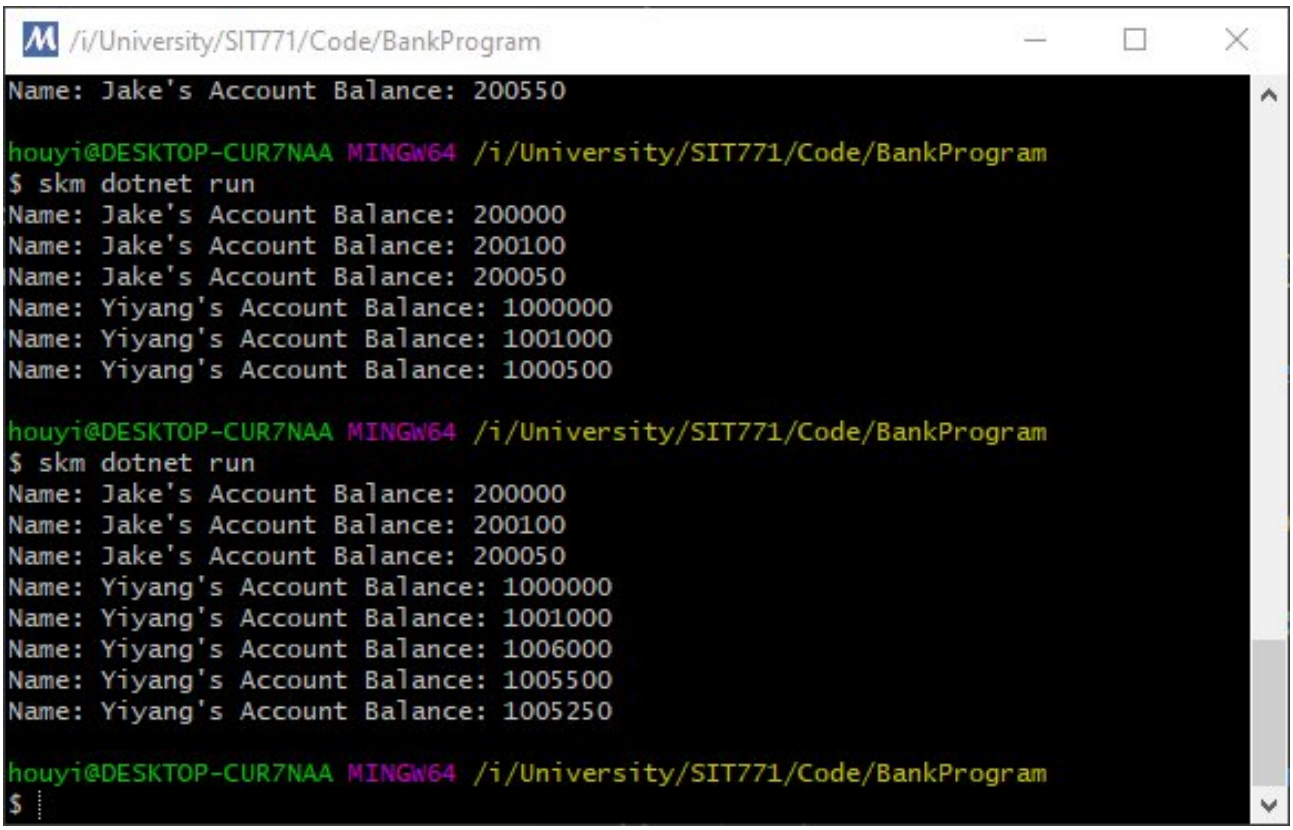
Outcome	**Weight**
Principles	♦♦◊◊◊
Build Programs	♦♦◊◊◊
Design	♦◊◊◊◊
Justify	♦◊◊◊◊

This task demonstrates the principle and the approach to build a program. This task also demonstrates the evaluation of the solution structure and the justification of the outcome.

August 3, 2020

```
using System;
using SplashKitSDK;

public class Program
{
    public static void Main()
    {
        Account account = new Account("Jake's Account", 200000);

        account.Print();
        account.Deposit(100);
        account.Print();
        account.Wiithdraw(50);
        account.Print();

        Account account1 = new Account("Yiyang's Account", 1000000);

        account1.Print();
        account1.Deposit(1000);
        account1.Print();
        account1.Deposit(5000);
        account1.Print();
        account1.Wiithdraw(500);
        account1.Print();
        account1.Wiithdraw(250);
        account1.Print();
    }
}
```

```cs
using System;


public class Account
{
    private decimal _balance;
    private string _name;

    //Constructor
    public Account(string name, decimal startingBalance)
    {
        _name = name;
        _balance = startingBalance;

    }

    //Deposit method
    public void Deposit(decimal amountToAdd)
    {
        _balance += amountToAdd;
    }

    //Withdraw mathod
    public void Wiithdraw(decimal amountToSubtract)
    {
        _balance -= amountToSubtract;
    }

    //Read name property
    public string Name
    {
        get { return _name; }
    }

    //Print method
    public void Print()
    {
        Console.WriteLine($"Name: {_name} Balance: {_balance}");
    }
}
```


/i/University/SIT771/Code/BankProgram
Name: Jake's Account Balance: 200550
houyi@DESKTOP-CUR7NAA MINGW64 /i/University/SIT771/Code/BankProgram
$ skm dotnet run
Name: Jake's Account Balance: 200000
Name: Jake's Account Balance: 200100
Name: Jake's Account Balance: 200050
Name: Yiyang's Account Balance: 1000000
Name: Yiyang's Account Balance: 1001000
Name: Yiyang's Account Balance: 1000500
houyi@DESKTOP-CUR7NAA MINGW64 /i/University/SIT771/Code/BankProgram
$ skm dotnet run
Name: Jake's Account Balance: 200000
Name: Jake's Account Balance: 200100
Name: Jake's Account Balance: 200050
Name: Yiyang's Account Balance: 1000000
Name: Yiyang's Account Balance: 1001000
Name: Yiyang's Account Balance: 1006000
Name: Yiyang's Account Balance: 1005500
Name: Yiyang's Account Balance: 1005250
houyi@DESKTOP-CUR7NAA MINGW64 /i/University/SIT771/Code/BankProgram
$

10 The Player Class

Create a player class

Outcome	Weight
Evaluate Code	♦◊◊◊◊

This task demonstrates the evaluation of codes, design of a program with evidence to justify.

Outcome	Weight
Design	♦◊◊◊◊

This task demonstrates the evaluation of codes, design of a program with evidence to justify.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the evaluation of codes, design of a program with evidence to justify.

Date	Author	Comment
2020/08/09 19:57	Yiyang Hou	Personal difficulties
2020/08/12 16:15	Yiyang Hou	Ready to Mark
2020/08/15 22:20	Chetan Arora	The class diagram in the task sheet doesn't show any input parameter for Draw() method
2020/08/15 22:20	Chetan Arora	- Think about how to code it with window parameter in Draw()
2020/08/15 22:20	Chetan Arora	Discuss
2020/08/16 19:59	Yiyang Hou	Thank you for the feedback Chetan! In fact, I have noticed that there is no parameter for Draw() method in the class diagram. But I can't figure out how to make a Draw() method work with Drawbitmap() method without passing in a Window object as a parameter, because Drawbitmap() requires a window destination, this is the only way I could find to locate the window.
2020/08/16 20:00	Yiyang Hou	Or is there someting else that I could use within Draw() method other than Drawbitmap() method. Or any other approach at all? Thank you!
2020/08/16 20:48	Yiyang Hou	I finally realized that I can just use Splashkit as the target for Drawbitmap() method. Now I don't need to have the parameter for Draw(). New code and screenshot are uploaded.
2020/08/26 10:55	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

The Player Class

Submitted By:
Yiyang Hou
houyiy
2020/08/16 20:48

Tutor:
Chetan Arora

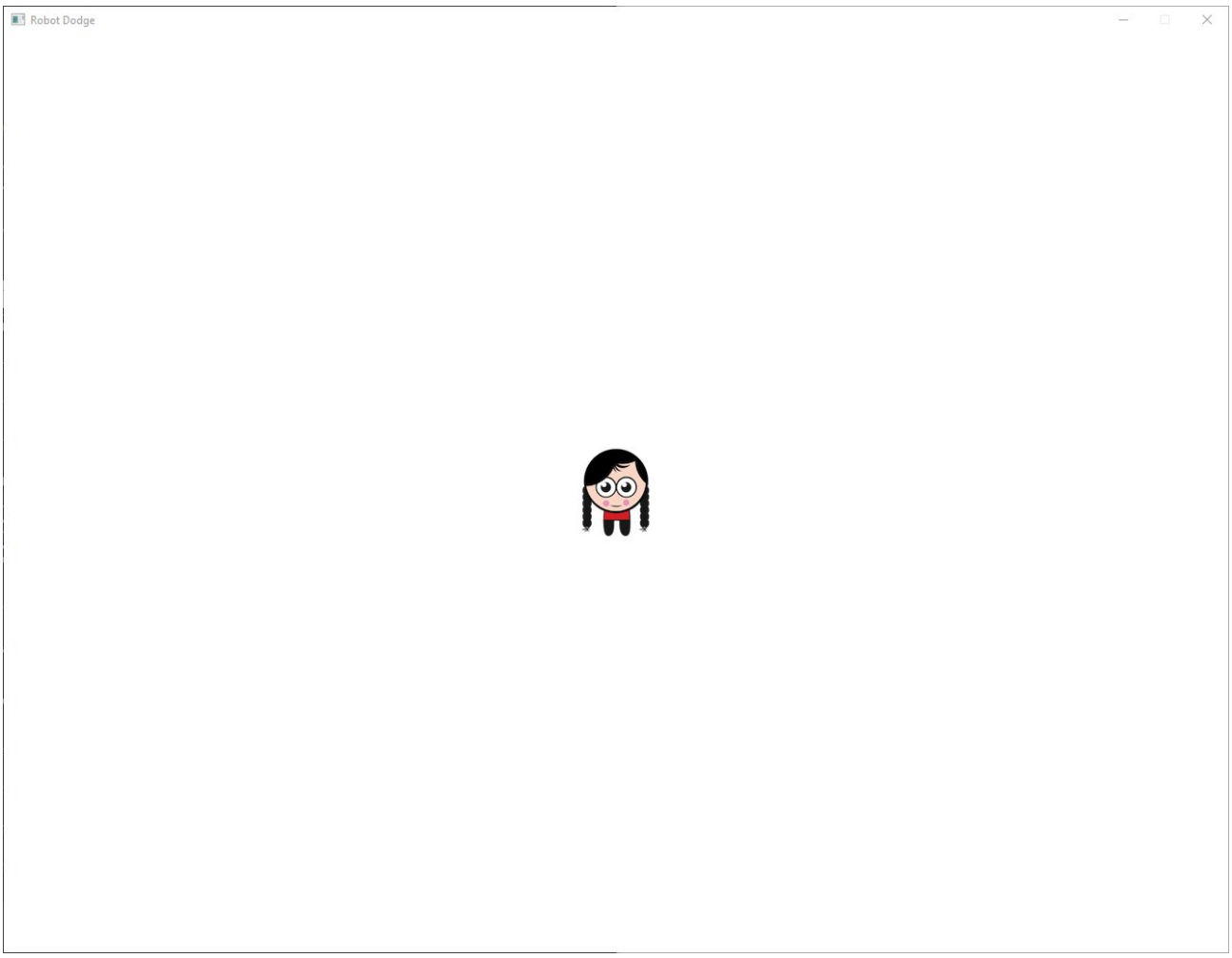
Outcome	**Weight**
Evaluate Code	♦◇◇◇◇
Design	♦◇◇◇◇
Justify	♦◇◇◇◇

This task demonstrates the evaluation of codes, design of a program with evidence to justify.

August 17, 2020

```
using System;
using SplashKitSDK;

public class Program
{
    public static void Main()
    {
        Window gameWindow = new Window("Robot Dodge", 1333, 999);
        Player plr = new Player(gameWindow);
        gameWindow.Clear(Color.White);
        gameWindow.Refresh(60);
        plr.Draw();
        gameWindow.Refresh(60);
        SplashKit.Delay(5000);



    }
}
```

```
using SplashKitSDK;


public class Player
{
    private Bitmap _PlayerBitmap;

    double X {get;set;}
    double Y {get;set;}

    public int Width
    {
        get
        {
            return _PlayerBitmap.Width;
        }
    }

     public int Height
    {
        get
        {
            return _PlayerBitmap.Height;
        }
    }

    public Player(Window windowObject)
    {
        _PlayerBitmap = new Bitmap("Player", "Player.png");

        X = (windowObject.Width - Width) / 2;
        Y = (windowObject.Height - Height) / 2;


    }

    public void Draw()
    {
        SplashKit.DrawBitmap(_PlayerBitmap, X, Y);
    }

}
```


Robot Dodge

11 Name Tester

Create a simple name testing program.

Outcome	Weight
Build Programs	♦♦◊◊◊

This task demonstrates how to build a program and is backed up by relevant evidence.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates how to build a program and is backed up by relevant evidence.

Date	Author	Comment
2020/08/09 19:56	Yiyang Hou	Personal difficulties
2020/08/14 20:28	Yiyang Hou	Ready to Mark
2020/08/15 22:24	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Name Tester

Submitted By:
Yiyang Hou
houyiy
2020/08/14 20:28

Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦◊◊◊
Justify	♦◊◊◊◊

This task demonstrates how to build a program and is backed up by relevant evidence.

August 14, 2020

```
using System;
using SplashKitSDK;

public enum MenuOption
{
    TestName, GuessThatNumber, Quit
}
public class Program
{
    public static void Main()
    {
        MenuOption userSelection;

        do{
            userSelection = ReadUserOption();

            switch(userSelection)
            {
                case MenuOption.TestName:
                    TestName();
                    break;

                case MenuOption.GuessThatNumber:
                    RunGuessThatNumber();
                    break;

                case MenuOption.Quit:
                    Console.WriteLine("QUIT!");
                    break;
            }
        } while (userSelection != MenuOption.Quit);
    }
    private static MenuOption ReadUserOption()  //User input
    {
        int option = 0;
        string userInput;


        Console.WriteLine("**--Menu--**");
        Console.WriteLine("--1 will run Test Name--");
        Console.WriteLine("--2 will play Guess That Number--");
        Console.WriteLine("--3 will Quit--");

        do
        {
            Console.Write("What is you choice? Give it here:");
            userInput = Console.ReadLine();
            try
            {
                option = Convert.ToInt32(userInput);

                if(option < 1 || option > 3)
                {
```

```
                    Console.WriteLine("Invalid number! Try again:");
                }
            }
            catch(Exception e)
            {
                Console.WriteLine(e.Message);
                Console.WriteLine("Because " + userInput + " is not a number! Try again:");
                option = -1;
            }
        }
        while( option < 1 || option > 3 );

        return (MenuOption)(option - 1);

    }

    public static void TestName()  //Name testing
    {
        string userName;
        Console.WriteLine("Please enter you name:");
        userName = Console.ReadLine();
        Console.WriteLine($"Hello {userName}!");
        if( userName.ToLower() == "yiyang")
        {
            Console.WriteLine("Welcome, my master!");
        }
        else if( userName.ToLower() == "freddy")
        {
            Console.WriteLine("Welcome, Yiyang's friend No.1!");
        }
        else if( userName.ToLower() == "doris")
        {
            Console.WriteLine("Welcome, Yiyang's friend No.2!");
        }
        else
        {
            Console.WriteLine("You are not a friend of Yiyang!");
        }

    }

    public static void RunGuessThatNumber()
    {
        int guess = 0, target, lowGuess = 1, highGuess = 100;

        target = new Random().Next(100) + 1;

        Console.WriteLine($"Guess a number between 1 and 100.");
        //Console.WriteLine($"(Whisper: It is {target})");
        while( guess != target )
        {
            guess = ReadGuess(lowGuess, highGuess);
```

```
106
107             if( guess < target)
108             {
109                 Console.WriteLine("Your guess is too low, Try again:");
110                 lowGuess = guess;
111             }
112             else if( guess > target)
113             {
114                 Console.WriteLine("Your guess is too high, Try again:");
115                 highGuess = guess;
116             }
117             else
118             {
119                 Console.WriteLine("Bingo!");
120             }
121         }
122
123     }
124
125     private static int ReadGuess(int min, int max)
126     {
127         int userGuess = 0;
128         string userInput;
129
130
131         Console.WriteLine( $"Enter your guess between {min} and {max}");
132         do
133         {
134             userInput = Console.ReadLine();
135
136             try
137             {
138                 userGuess = Convert.ToInt32(userInput);
139                 if(userGuess < min || userGuess > max)
140                 {
141                     Console.WriteLine("Input out of range! Try again:");
142                 }
143             }
144             catch(Exception e)
145             {
146                 Console.WriteLine(e.Message);
147                 Console.WriteLine("Because " + userInput + " is not a number! Try
                    ↪  again:");
148                 userGuess = -1;
149             }
150         }
151         while( userGuess < min || userGuess > max );
152
153         return userGuess;
154     }
155
156 }
```


/i/University/SIT771/Code/NameTester
hoyu@DESKTOP-CU87NAA MINGW64 ~
$ /i/University/SIT771/Code/NameTester
-bash: /i/University/SIT771/Code/NameTester: Is a directory
hoyu@DESKTOP-CU87NAA MINGW64 ~
$ cd /i/University/SIT771/Code/NameTester
hoyu@DESKTOP-CU87NAA MINGW64 /i/University/SIT771/Code/NameTester
$ sbm dotnet run
--Menu--
--1 will run Test Name--
--2 will play Guess That Number--
--3 will Quit--
What is you choice? Give it here: 1
Please enter you name:
jack
Hello jack!
You are not a friend of Yiyang!
--Menu--
--1 will run Test Name--
--2 will play Guess That Number--
--3 will Quit--
What is you choice? Give it here:1
Please enter you name:
Yiyang
Hello Yiyang!
Welcome, my master!
--Menu--
--1 will run Test Name--
--2 will play Guess That Number--
--3 will Quit--
What is you choice? Give it here:1
Please enter you name:
Freddy
Hello Freddy!
Welcome, Yiyang's friend No.1!
--Menu--
--1 will run Test Name--
--2 will play Guess That Number--
--3 will Quit--
What is you choice? Give it herex
Input string was not in a correct format.
Because x is not a number! Try again:
What is you choice? Give it here:0
Invalid number! Try again:
What is you choice? Give it here:2
Guess a number between 1 and 100.
(Whisper: It is 11)
Enter your guess between 1 and 100
b
Input string was not in a correct format.
Because b is not a number! Try again:
0
Input out of range! Try again:
5
Your guess is too low, Try again:
Enter your guess between 5 and 100
50
Your guess is too high, Try again:
Enter your guess between 5 and 50
11
Bingo!
--Menu--
--1 will run Test Name--
--2 will play Guess That Number--
--3 will Quit--
What is you choice? Give it here:3
QUIT!
hoyu@DESKTOP-CU87NAA MINGW64 /i/University/SIT771/Code/NameTester
$

12 Validating Accounts

Add validations to the program to make sure things are used correctly.

Outcome	Weight
Build Programs	♦♦◊◊◊

This task demonstrates the build of a program by learning with appropriate diagrams and texts. Evidence is also provided to justify.

Outcome	Weight
Design	♦◊◊◊◊

This task demonstrates the build of a program by learning with appropriate diagrams and texts. Evidence is also provided to justify.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the build of a program by learning with appropriate diagrams and texts. Evidence is also provided to justify.

Date	Author	Comment
2020/08/09 19:56	Yiyang Hou	Personal difficulties
2020/08/16 18:17	Yiyang Hou	Ready to Mark
2020/08/16 18:17	Yiyang Hou	Task Discussion:1.Accounts can be created by initializing different Account objects and can be selected by passing into methods as their parameter. For menu options, the selection is carried out by selecting a particular integer that refers to the index of an option within the MenuOption enumeration. Then the corresponding option can be returned and stored in a variable for later process.2.The advantage of using an enumeration for the menu is that all the options are created and stored in one place, as well as indexed in order. It makes the options easy to refer to. The alternative to using enumeration is that we can just treat these menu options as strings, store they in different variables, call one of the variables when the program needs a particular option.3.The repetitions are carried out by using do while loop in this program. This allows us to run a block of code for at least once before it checks for any conditions.4.The control flow statements can help the objects to do repetitive things, do selections and check for conditions with minimal code and simple logic.
2020/08/18 14:32	Chetan Arora	- Can you add warning messages for user, if they try to deposit a negative amount or try to withdraw more than they have.
2020/08/18 14:32	Chetan Arora	- Well done
2020/08/18 14:32	Chetan Arora	Fix and Resubmit
2020/08/18 14:33	Chetan Arora	Discuss
2020/08/18 18:51	Yiyang Hou	I have uploaded an updated version of code for the warning messages. On top of the general failed transaction message, the program will now print out the specific reasons for the failures. I put the warning message printing methods into the validation code of Deposit() and Withdraw() in Accounts.cs.
2020/08/19 12:00	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Validating Accounts

Submitted By:
Yiyang Hou
houyiy
2020/08/18 18:51

Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦◊◊◊
Design	♦◊◊◊◊
Justify	♦◊◊◊◊

This task demonstrates the build of a program by learning with appropriate diagrams and texts. Evidence is also provided to justify.

August 18, 2020

```
using System;
using SplashKitSDK;

public enum MenuOption
{
    Withdraw,
    Deposit,
    Print,
    Quit
}
public class Program
{
    public static void Main()
    {
        Account YiyangsAcount = new Account("Yiyang Hou", 10000);
        MenuOption userSelection;
       do
       {
            userSelection = ReadUserOption();
            switch(userSelection)
            {
                case MenuOption.Withdraw:
                DoWithdraw(YiyangsAcount);
                break;

                case MenuOption.Deposit:
                DoDeposit(YiyangsAcount);
                break;

                case MenuOption.Print:
                DoPrint(YiyangsAcount);
                break;

                case MenuOption.Quit:
                Console.WriteLine("QUIT!");
                break;
            }
       }
       while( userSelection != MenuOption.Quit);

    }

    public static MenuOption ReadUserOption()
    {
        int usersOption = 0;

        do
        {
            Console.WriteLine("*****Menu***** \n--------------------\n1.
            ↪  ---Withdraw---\n2. ---Deposit---\n3. ---Print---\n4.
            ↪  ---Quit---\n--------------------");
            Console.Write("What would you like to do today: ");
            string userInput = Console.ReadLine();
```

```
52              try{
53                  usersOption = Convert.ToInt32(userInput);
54                  if (usersOption < 1 || usersOption > 4)
55                  {
56                      Console.WriteLine("Invalid Input, please try again!\n");
57                  }
58              }
59              catch(Exception e)
60              {
61                  Console.WriteLine(e.Message);
62                  Console.WriteLine($"{userInput} is not a number, Please enter a
                    ↪  number!\n");
63                  usersOption = -1;
64              }
65          }
66          while(usersOption < 1 || usersOption > 4);
67          return (MenuOption)(usersOption - 1);
68      }
69
70
71      public static void DoWithdraw(Account account)
72      {
73          decimal withdrawAmount = 0;
74          Console.Write("How much would you like to WITHDRAW? Please enter: $");
75          string userWithdraw = Console.ReadLine();
76          try
77          {
78              withdrawAmount = Convert.ToDecimal(userWithdraw);
79              bool successfulness = account.Withdraw(withdrawAmount);
80              if(successfulness == true)
81              {
82                  Console.WriteLine("Yay! Withdrawal SUCCEEDED!\n");
83              }
84              else
85              {
86                  Console.WriteLine("Boo-boo! Withdrawal FAILED!\n");
87              }
88          }
89          catch(Exception e)
90          {
91              Console.WriteLine(e.Message);
92              Console.WriteLine($"{userWithdraw} is not decimal number!\n");
93          }
94      }
95
96      public static void DoDeposit(Account account)
97      {
98          decimal depositAmount = 0;
99          Console.Write("How much would you like to DEPOSIT? Please enter: $");
100         string userDeposit = Console.ReadLine();
101         try
102         {
103             depositAmount = Convert.ToDecimal(userDeposit);
```

```
104             bool successfulness = account.Deposit(depositAmount);
105             if(successfulness == true)
106             {
107                 Console.WriteLine("Yay! Deposit SUCCEEDED!\n");
108             }
109             else
110             {
111                 Console.WriteLine("Boo-boo! Deposit FAILED!\n");
112             }
113         }
114         catch(Exception e)
115         {
116             Console.WriteLine(e.Message);
117             Console.WriteLine( $ "{userDeposit} is not decimal number!\n");
118         }
119     }
120 
121     public static void DoPrint(Account account)
122     {
123         account.Print();
124     }
125 }
```

```cs
using System;


public class Account
{
    private decimal _balance;
    private string _name;

    //Constructor
    public Account(string name, decimal startingBalance)
    {
        _name = name;
        _balance = startingBalance;

    }

    //Deposit method
    public bool Deposit(decimal amountToDeposit)
    {
        if (amountToDeposit > 0)
        {
            _balance += amountToDeposit;
            return true;
        }
        else
        {
            Console.WriteLine("**WARNING**\nThe deposit amount must be a positive number!");
            return false;
        }
    }

    //Withdraw mathod
    public bool Withdraw(decimal amountToWithdraw)
    {
        if(amountToWithdraw > 0 && amountToWithdraw <= _balance )
        {
            _balance -= amountToWithdraw;
            return true;
        }
        else if (amountToWithdraw <= 0)
        {
            Console.WriteLine("**WARNING**\nThe withdraw amount must be a positive number!");
            return false;
        }
        else
        {
            Console.WriteLine("**WARNING**\nThe withdraw amount must not exceed your exiting funds!");
            return false;
        }

```

```
51      }
52
53      //Read name property
54      public string Name
55      {
56          get { return _name; }
57      }
58
59      //Print method
60      public void Print()
61      {
62          Console.WriteLine( $"Account Name: {Name}\nBalance: {_balance}\n");
63      }
64  }
```


AS\University\SIT771\Code\ValidatingAccounts
$ skm dotnet run
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Quit---
----
what would you like to do today: 3
Account Name: Yiyang Hou
Balance: 10000
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Quit---
----
what would you like to do today: 1
How much would you like to WITHDRAW? Please enter: $3000
Yay! Withdrawal SUCCEEDED!
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Quit---
----
what would you like to do today: 3
Account Name: Yiyang Hou
Balance: 7000
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Quit---
----
what would you like to do today: 2
How much would you like to DEPOSIT? Please enter: $5000
Yay! Deposit SUCCEEDED!
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Quit---
----
what would you like to do today: 3
Account Name: Yiyang Hou
Balance: 12000
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Quit---
----
what would you like to do today: 1
How much would you like to WITHDRAW? Please enter: $15000
WARNING
The withdraw amount must not exceed your exiting funds!
Boo-boo! Withdrawal FAILED!
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Quit---
----
what would you like to do today: 2
How much would you like to DEPOSIT? Please enter: $-1
WARNING
The deposit amount must be a positive number!
Boo-boo! Deposit FAILED!
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Quit---
----
what would you like to do today: 1
How much would you like to WITHDRAW? Please enter: $-10
WARNING
The withdraw amount must be a positive number!
Boo-boo! Withdrawal FAILED!
*****Menu*****
----

13 Moving The Player

Get your robot character moving

Outcome	Weight
Build Programs	♦♦◊◊◊

This task demonstrates how a program is designed and built and is backed up with relevant evidence.

Outcome	Weight
Design	♦◊◊◊◊

This task demonstrates how a program is designed and built and is backed up with relevant evidence.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates how a program is designed and built and is backed up with relevant evidence.

Date	Author	Comment
2020/08/19 22:38	Yiyang Hou	Ready to Mark
2020/08/20 14:14	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Moving The Player

Submitted By:
Yiyang Hou
houyiy
2020/08/19 22:38

Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦◊◊◊
Design	♦◊◊◊◊
Justify	♦◊◊◊◊

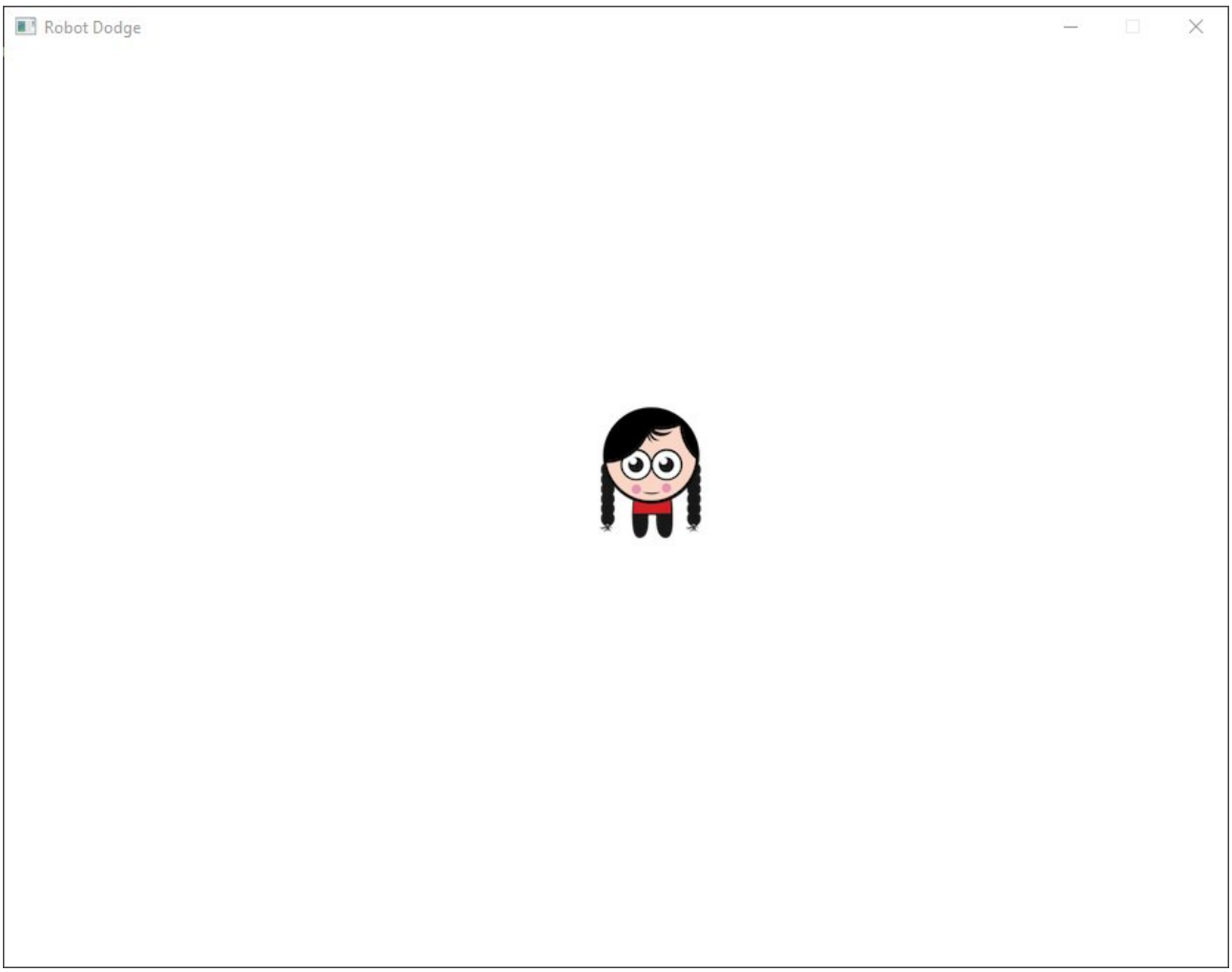
This task demonstrates how a program is designed and built and is backed up with relevant evidence.

August 19, 2020

```
using System;
using SplashKitSDK;

public class Program
{
    public static void Main()
    {
        Window gameWindow = new Window("Robot Dodge", 888, 666);
        Player plr = new Player(gameWindow);

        while( !(plr.Quit || SplashKit.WindowCloseRequested(gameWindow)))
        {
            plr.HandleInput();
            plr.StayOnWindow(gameWindow);
            gameWindow.Clear(Color.White);
            plr.Draw();
            gameWindow.Refresh(60);
        }

        gameWindow.Close();
        gameWindow = null;

    }
}
```

```
using SplashKitSDK;


public class Player
{
    private Bitmap _PlayerBitmap;

    double X {get;set;}
    double Y {get;set;}

    public bool Quit {get; private set;}


    public int Width
    {
        get
        {
            return _PlayerBitmap.Width;
        }
    }

     public int Height
    {
        get
        {
            return _PlayerBitmap.Height;
        }
    }

    public Player(Window windowObject)
    {
        _PlayerBitmap = new Bitmap("Player", "Player.png");
        Quit = false;

        X = (windowObject.Width - Width) / 2;
        Y = (windowObject.Height - Height) / 2;
    }

    public void Draw()
    {
        _PlayerBitmap.Draw(X, Y);
    }

    public void HandleInput()
    {
        int speed = 5;

        SplashKit.ProcessEvents();

        if(SplashKit.KeyReleased(KeyCode.EscapeKey)) Quit = true;

        if(SplashKit.KeyDown(KeyCode.LeftShiftKey) ||
        ↪  SplashKit.KeyDown(KeyCode.RightShiftKey))
```

```
53          {
54              speed = 15;
55          }
56
57          if(SplashKit.KeyDown(KeyCode.WKey) || SplashKit.KeyDown(KeyCode.UpKey))
58          {
59              Y -= speed;
60          }
61
62          if(SplashKit.KeyDown(KeyCode.SKey) || SplashKit.KeyDown(KeyCode.DownKey))
63          {
64              Y += speed;
65          }
66
67          if(SplashKit.KeyDown(KeyCode.AKey) || SplashKit.KeyDown(KeyCode.LeftKey))
68          {
69              X -= speed;
70          }
71
72          if(SplashKit.KeyDown(KeyCode.DKey) || SplashKit.KeyDown(KeyCode.RightKey))
73          {
74              X += speed;
75          }
76
77      }
78
79      public void StayOnWindow(Window limit)
80      {
81          const int GAP = 10;
82          if( X < GAP)
83          {
84              X = GAP;
85          }
86          if( X + Width > limit.Width - GAP)
87          {
88              X = limit.Width - GAP - Width;
89          }
90
91          if( Y < GAP)
92          {
93              Y = GAP;
94          }
95
96          if( Y + Height > limit.Height - GAP)
97          {
98              Y = limit.Height - GAP - Height;
99          }
100     }
101
102 }
```


Robot Dodge

14 Transactions

Add transaction classes to your program

Outcome	Weight
Build Programs	♦♦♦◇◇

This task demonstrates the design and the build of a program and is justified by relevant evidence.

Outcome	Weight
Design	♦◇◇◇◇

This task demonstrates the design and the build of a program and is justified by relevant evidence.

Outcome	Weight
Justify	♦◇◇◇◇

This task demonstrates the design and the build of a program and is justified by relevant evidence.

Date	Author	Comment
2020/08/21 01:02	Yiyang Hou	Ready to Mark
2020/08/21 01:02	Yiyang Hou	Task Discussion:1.The objects are created and used based on the information that they retain and the actions that the objects can perform, which are defined by the class. These factors determine what are the roles of the objects in the program. The core principle of abstraction is that we should have a clear idea of what do we need of an object and create a class that defines such an object that fits to that role.2.Within the TransferTransaction class, a DepositTransaction object and a WithdrawTransaction object are created and conduct a series of actions based on its own needs. Those two objects have their own identity and information and they only share their public information to the external access. From an external perspective, we would not know that it uses other objects internally, because the use of those objects are the information it retains in encapsulation and hidden from the external.
2020/08/23 12:43	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Transactions

Submitted By:
Yiyang Hou
houyiy
2020/08/21 01:02

Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦♦◊◊
Design	♦◊◊◊◊
Justify	♦◊◊◊◊

This task demonstrates the design and the build of a program and is justified by relevant evidence.

August 21, 2020

```
using System;
using SplashKitSDK;

public enum MenuOption
{
    Withdraw,
    Deposit,
    Print,
    Transfer,
    Quit
}
public class Program
{
    public static void Main()
    {
        Account yiyangsAccount = new Account("Yiyang", 10000);
        Account jakesAccount = new Account("Jake", 50000);
        MenuOption userSelection;
       do
       {
            userSelection = ReadUserOption();
            switch(userSelection)
            {
                case MenuOption.Withdraw:
                DoWithdraw(yiyangsAccount);
                break;

                case MenuOption.Deposit:
                DoDeposit(yiyangsAccount);
                break;

                case MenuOption.Print:
                DoPrint(yiyangsAccount, jakesAccount);
                break;

                case MenuOption.Transfer:
                DoTransfer(jakesAccount, yiyangsAccount);
                break;

                case MenuOption.Quit:
                Console.WriteLine("QUIT!");
                break;
            }
       }
       while( userSelection != MenuOption.Quit);

    }

    public static MenuOption ReadUserOption()
    {
        int usersOption = 0;

        do
```

```
54          {
55              Console.WriteLine("*****Menu***** \n--------------------\n1.
                ↪  ---Withdraw---\n2. ---Deposit---\n3. ---Print---\n4.
                ↪  ---Transfer---\n5. ---Quit---\n--------------------");
56              Console.Write("What would you like to do today: ");
57              string userInput = Console.ReadLine();
58              try{
59                  usersOption = Convert.ToInt32(userInput);
60                  if (usersOption < 1 || usersOption > 5)
61                  {
62                      Console.WriteLine("Invalid Input, please try again!\n");
63                  }
64              }
65              catch(Exception e)
66              {
67                  Console.WriteLine($"{e.Message}\n");
68                  usersOption = -1;
69              }
70          }
71          while(usersOption < 1 || usersOption > 5);
72          return (MenuOption)(usersOption - 1);
73      }
74
75
76      public static void DoWithdraw(Account account)
77      {
78          decimal withdrawAmount = 0;
79          Console.Write("How much would you like to WITHDRAW? Please enter: $");
80          string userWithdraw = Console.ReadLine();
81          try
82          {
83              withdrawAmount = Convert.ToDecimal(userWithdraw);
84              WithdrawTransaction withdrawTransaction = new
                ↪  WithdrawTransaction(account, withdrawAmount);
85              withdrawTransaction.Execute();
86              withdrawTransaction.Print();
87          }
88          catch(Exception e)
89          {
90              Console.WriteLine($"{e.Message}\n");
91          }
92      }
93
94      public static void DoDeposit(Account account)
95      {
96          decimal depositAmount = 0;
97          Console.Write("How much would you like to DEPOSIT? Please enter: $");
98          string userDeposit = Console.ReadLine();
99          try
100         {
101             depositAmount = Convert.ToDecimal(userDeposit);
102             DepositTransaction depositTransaction = new DepositTransaction(account,
                ↪  depositAmount);
```

```
103             depositTransaction.Execute();
104             depositTransaction.Print();
105         }
106         catch(Exception e)
107         {
108             Console.WriteLine($"{e.Message}\n");
109         }
110     }
111 
112     public static void DoPrint(Account account, Account account1)
113     {
114         account.Print();
115         account1.Print();
116     }
117 
118     public static void DoTransfer(Account fromAccount, Account toAccount)
119     {
120         decimal transferAmount = 0;
121         Console.Write($"How much would you like to TRANSFER from
            ↪ {fromAccount.Name}'s account to {toAccount.Name}'s account? Please
            ↪ enter: $");
122         string userTransfer = Console.ReadLine();
123         try
124         {
125             transferAmount = Convert.ToDecimal(userTransfer);
126             TransferTransaction transferTransaction = new
                ↪ TransferTransaction(fromAccount, toAccount, transferAmount);
127             transferTransaction.Execute();
128             transferTransaction.Print();
129         }
130         catch(Exception e)
131         {
132             Console.WriteLine($"{e.Message}\n");
133         }
134     }
135 }
```

```cs
using System;


public class Account
{
    private decimal _balance;
    private string _name;

    //Constructor
    public Account(string name, decimal startingBalance)
    {
        _name = name;
        _balance = startingBalance;

    }

    //Deposit method
    public bool Deposit(decimal amountToDeposit)
    {
        if (amountToDeposit > 0)
        {
            _balance += amountToDeposit;
            return true;
        }
        else
        {
            Console.WriteLine("**WARNING**\nThe deposit amount must be a positive number!");
            return false;
        }
    }

    //Withdraw mathod
    public bool Withdraw(decimal amountToWithdraw)
    {
        if(amountToWithdraw > 0 && amountToWithdraw <= _balance )
        {
            _balance -= amountToWithdraw;
            return true;
        }
        else if (amountToWithdraw <= 0)
        {
            Console.WriteLine("**WARNING**\nThe withdraw amount must be a positive number!");
            return false;
        }
        else
        {
            Console.WriteLine("**WARNING**\nThe withdraw amount must not exceed your exiting funds!");
            return false;
        }

```

```
51      }
52
53      //Read name property
54      public string Name
55      {
56          get { return _name; }
57      }
58
59      //Print method
60      public void Print()
61      {
62          Console.WriteLine( $"Account Name: {Name}\nBalance: {_balance}\n");
63      }
64  }
```

```
using System;
using SplashKitSDK;

public class WithdrawTransaction
{
    private Account _account;
    private decimal _amount;
    private bool _executed = false;
    private bool _success = false;
    private bool _reversed = false;

    public bool Executed
    {
        get
        {
            return _executed;
        }
    }
    public bool Success
    {
        get
        {
            return _success;
        }
    }
    public bool Reversed
    {
        get
        {
            return _reversed;
        }
    }

    public WithdrawTransaction(Account account, decimal amount)
    {
        _account = account;
        _amount = amount;
    }

    public void Execute()
    {
        if( _executed )
        {
            throw new Exception("Cannot execute this transaction as it has already
            ↪  been exected.");
        }

        _executed = true;
        _success = _account.Withdraw(_amount);
    }

    public void Rollback()
    {
```

```
53          if( !_executed )
54          {
55              throw new Exception("Cannot rollback this transaction as it has not
                ↪  been executed.");
56          }
57          if( _reversed )
58          {
59              throw new Exception("Cannot rollback this transaction as it has already
                ↪  been reversed.");
60          }
61          _reversed = true;
62          _account.Deposit(_amount);
63      }
64
65      public void Print()
66      {
67          if( _success )
68          {
69              Console.Write($"*${_amount} was SUCCESSFULLY withdrawed from
                ↪  {_account.Name}'s account.");
70          }
71          else
72          {
73              Console.Write("*This Withdraw transaction was FAILED!");
74          }
75
76          if( _reversed )
77          {
78              Console.WriteLine("|The transaction was rolled back.");
79          }
80          else
81          {
82              Console.WriteLine("|The transaction was not rolled back.");
83          }
84
85      }
86
87
88  }
```

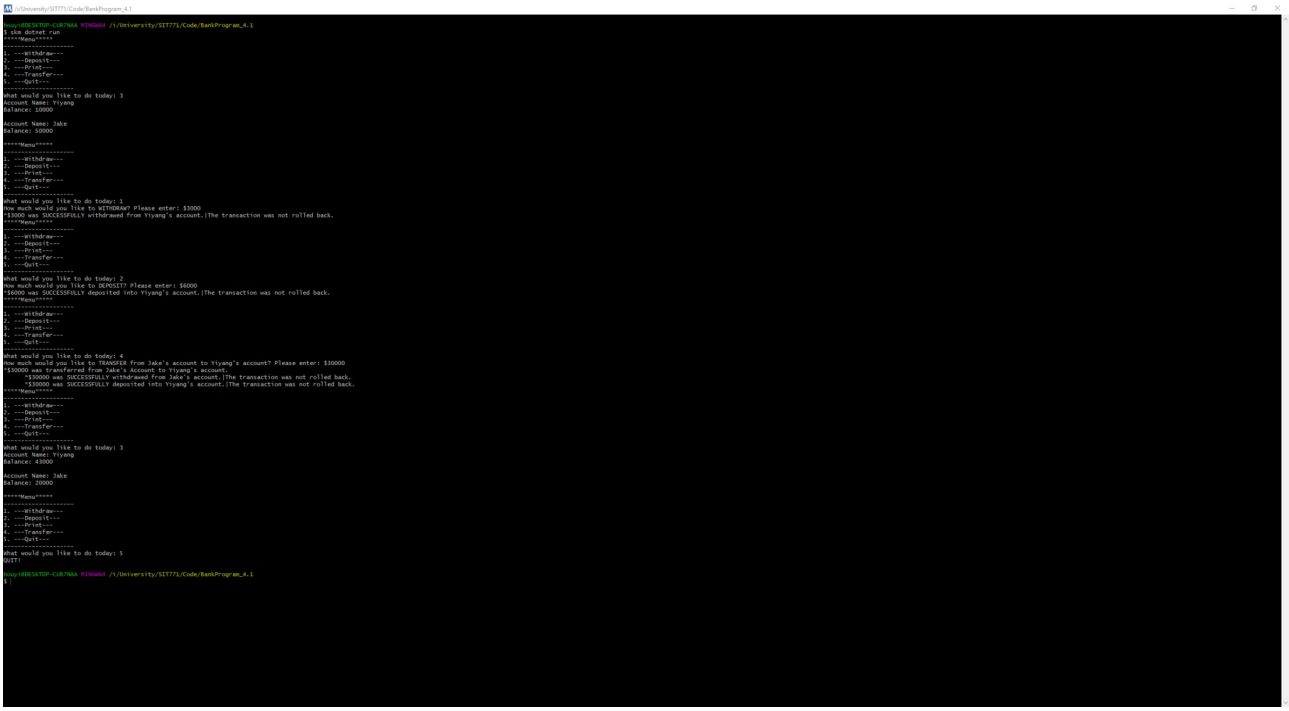
```
using System;
using SplashKitSDK;

public class DepositTransaction
{
    private Account _account;
    private decimal _amount;
    private bool _executed = false;
    private bool _success = false;
    private bool _reversed = false;

    public bool Executed
    {
        get
        {
            return _executed;
        }
    }
    public bool Success
    {
        get
        {
            return _success;
        }
    }
    public bool Reversed
    {
        get
        {
            return _reversed;
        }
    }

    public DepositTransaction(Account account, decimal amount)
    {
        _account = account;
        _amount = amount;
    }

    public void Execute()
    {
        if( _executed )
        {
            throw new Exception("Cannot execute this transaction as it has already
            ↪  been exected.");
        }

        _executed = true;
        _success = _account.Deposit(_amount);
    }

    public void Rollback()
    {
```

```
53          if( !_executed )
54          {
55              throw new Exception("Cannot rollback this transaction as it has not
                ↪  been executed.");
56          }
57          if( _reversed )
58          {
59              throw new Exception("Cannot rollback this transaction as it has already
                ↪  been reversed.");
60          }
61          _reversed = true;
62          _account.Withdraw(_amount);
63      }
64
65      public void Print()
66      {
67          if( _success )
68          {
69              Console.Write($"*${_amount} was SUCCESSFULLY deposited into
                ↪  {_account.Name}'s account.");
70          }
71          else
72          {
73              Console.Write("*This deposit transaction was FAILED!");
74          }
75
76          if( _reversed )
77          {
78              Console.WriteLine("|The transaction was rolled back.");
79          }
80          else
81          {
82              Console.WriteLine("|The transaction was not rolled back.");
83          }
84
85      }
86
87
88  }
```

```
using System;

public class TransferTransaction
{
    private Account _toAccount;
    private Account _fromAccount;
    private decimal _amount;

    private DepositTransaction _theDeposit;
    private WithdrawTransaction _theWithdraw;
    private bool _executed = false;
    private bool _reversed = false;

    public bool Executed
    {
        get
        {
            return _executed;
        }
    }
    public bool Success
    {
        get
        {
            return (_theDeposit.Success && _theWithdraw.Success);
        }
    }
    public bool Reversed
    {
        get
        {
            return _reversed;
        }
    }
    public TransferTransaction(Account fromAccount, Account toAccount, decimal
    ↪ amount)
    {
        _fromAccount = fromAccount;
        _toAccount = toAccount;
        _amount = amount;
        _theWithdraw = new WithdrawTransaction(fromAccount, amount);
        _theDeposit = new DepositTransaction(toAccount, amount);
    }

    public void Execute()
    {
        if(_executed)
        {
            throw new Exception("Cannot execute this transaction as it has already
            ↪ been exected.");
        }
        _theWithdraw.Execute();
        if( _theWithdraw.Success )
```

```
52          {
53              _theDeposit.Execute();
54
55              if( !_theDeposit.Success && _theDeposit.Executed)
56              {
57                  _theWithdraw.Rollback();
58              }
59          }
60          _executed = true;
61      }
62
63      public void Rollback()
64      {
65          if( !_executed )
66          {
67              throw new Exception("Cannot rollback this transaction as it has not
                ↪  been executed.");
68          }
69          if( _reversed )
70          {
71              throw new Exception("Cannot rollback this transaction as it has already
                ↪  been reversed.");
72          }
73          if(_theWithdraw.Success)
74          {
75              _theWithdraw.Rollback();
76          }
77          if(_theDeposit.Success)
78          {
79              _theDeposit.Rollback();
80          }
81
82          _reversed = true;
83      }
84
85      public void Print()
86      {
87          if( Success )
88          {
89              Console.WriteLine($"*${_amount} was transferred from
                ↪  {_fromAccount.Name}'s Account to {_toAccount.Name}'s account.");
90              Console.Write($"      ");
91              _theWithdraw.Print();
92              Console.Write($"      ");
93              _theDeposit.Print();
94          }
95          else
96          {
97              Console.WriteLine("*This transaction was FAILED!");
98              Console.Write($"      ");
99              _theWithdraw.Print();
100             Console.Write($"      ");
101             _theDeposit.Print();
```

```
102         }
103
104         if( _reversed )
105         {
106             Console.WriteLine("|The transaction was rolled back.");
107         }
108
109
110     }
111
112 }
```


hoyu1@DESKTOP-CUR7NAA MINGW64 /I/University/SIT771/Code/BankProgram_4.1
$ dotnet run
*****Menu*****
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---Quit---
what would you like to do today: 3
Account Name: Yiyang
Balance: 10000
Account Name: Jake
Balance: 10000
what would you like to do today: 1
How much would you like to WITHDRAW? Please enter: $3000
$3000 was SUCCESSFULLY withdrawed from Yiyang's account.|The transaction was not rolled back.
what would you like to do today: 2
How much would you like to DEPOSIT? Please enter: $6000
$6000 was SUCCESSFULLY deposited into Yiyang's account.|The transaction was not rolled back.
what would you like to do today: 4
How much would you like to TRANSFER from Jake's account to Yiyang's account? Please enter: $30000
$30000 was transferred from Jake's Account to Yiyang's account.
$30000 was SUCCESSFULLY withdrawed from Jake's account.|The transaction was not rolled back.
$30000 was SUCCESSFULLY deposited into Yiyang's account.|The transaction was not rolled back.
what would you like to do today: 3
Account Name: Yiyang
Balance: 43000
Account Name: Jake
Balance: 20000
what would you like to do today: 5
QUIT!
hoyu1@DESKTOP-CUR7NAA MINGW64 /I/University/SIT771/Code/BankProgram_4.1
$

15 Messy Code

What does this mess do? Lets fix it up!

Outcome	Weight
Evaluate Code	♦♦♦◊◊

This task demonstrates the ability to evaluate and fix some code, as well as building and testing a program. It is also backed up with relevant evidence.

Outcome	Weight
Build Programs	♦◊◊◊◊

This task demonstrates the ability to evaluate and fix some code, as well as building and testing a program. It is also backed up with relevant evidence.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the ability to evaluate and fix some code, as well as building and testing a program. It is also backed up with relevant evidence.

Date	Author	Comment
2020/08/22 00:31	Yiyang Hou	Ready to Mark
2020/08/23 13:29	Chetan Arora	Well done. You also don't need %360 in line 108
2020/08/23 13:29	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Messy Code

Submitted By:
Yiyang Hou
houyiy
2020/08/22 00:31

Tutor:
Chetan Arora

Outcome	**Weight**
Evaluate Code	♦♦♦◇◇
Build Programs	♦◇◇◇◇
Justify	♦◇◇◇◇

This task demonstrates the ability to evaluate and fix some code, as well as building and testing a program. It is also backed up with relevant evidence.

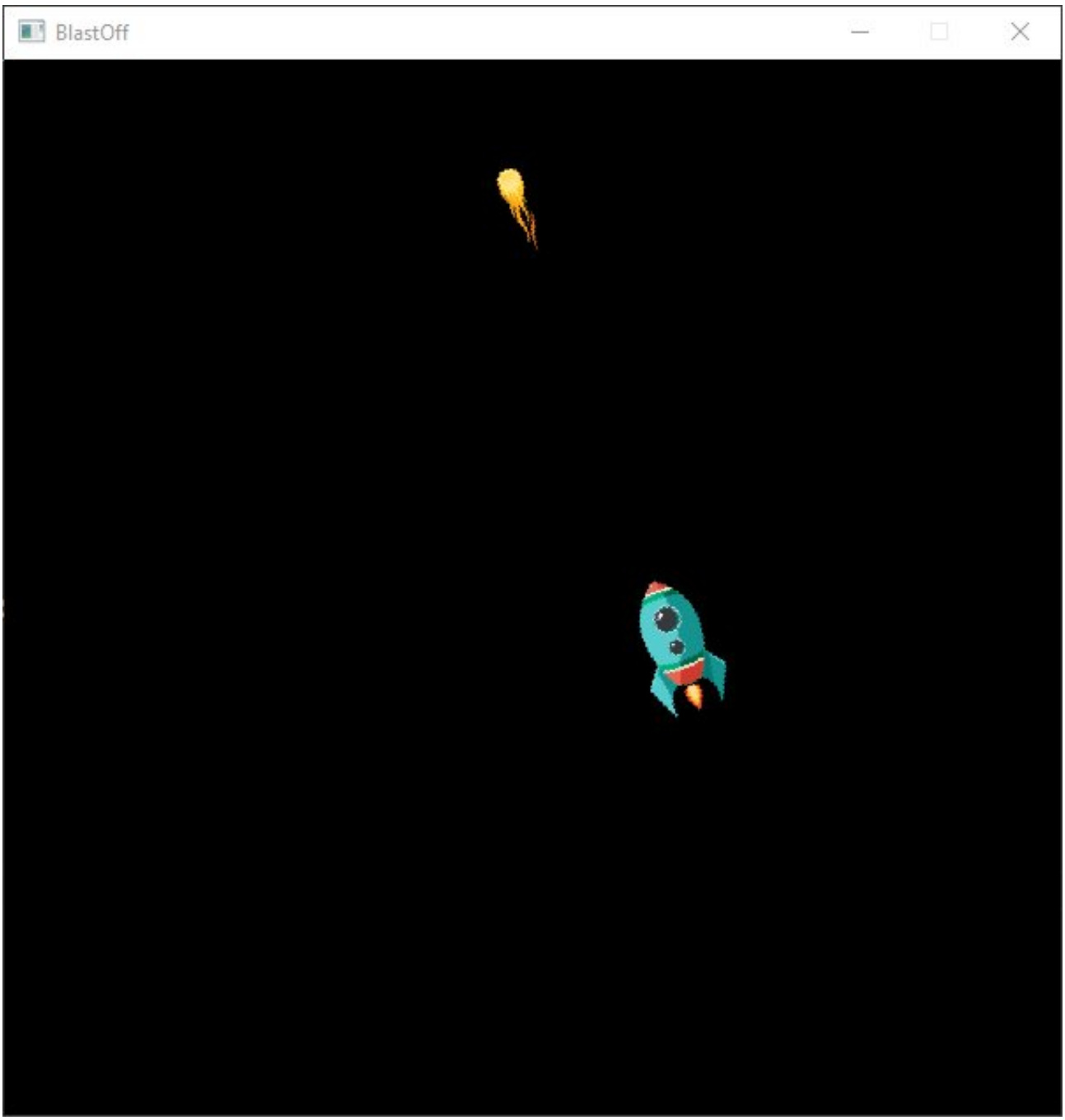
August 22, 2020

```
//TODO: It should be the ship firing the bullet!!! What have I done wrong :(
//TODO: Looks like 90 degrees is being added to the movement of the bullet somewhere...
//but I cant find it :( If only I could actually READ this code.
using System;
using SplashKitSDK;

namespace CharacterDrawing1
{
    public class Program
    {
        public static void Main()
        {
            SpaceGame game = new SpaceGame();
            game.Run();
        }
    }

    public class SpaceGame
    {
        private SpaceShip _spaceShip;
        private Window _gameWindow;
        public SpaceGame()
        {
            BitmapLoading();
            _spaceShip = new SpaceShip { X = 110, Y = 110 };
        }
        private void BitmapLoading()
        {
            SplashKit.LoadBitmap("Aquarii", "Aquarii.png");     // Names in the wrong order
            SplashKit.LoadBitmap("Gliese", "Gliese.png");
            SplashKit.LoadBitmap("Pegasi", "Pegasi.png");
            SplashKit.LoadBitmap("Bullet", "Fire.png");
        }
        public void Run()
        {
            _gameWindow = new Window("BlastOff", 600, 600);
            while (!_gameWindow.CloseRequested)
            {
                SplashKit.ProcessEvents();

                if (SplashKit.KeyDown(KeyCode.UpKey))
                {
                    _spaceShip.Move(4, 0);
                }

                if (SplashKit.KeyDown(KeyCode.DownKey))
                {
                    _spaceShip.Move(-4, 0);
                }
                if (SplashKit.KeyDown(KeyCode.LeftKey))
                {
```

```
52                      _spaceShip.Rotate(-4);
53                  }
54                  if (SplashKit.KeyDown(KeyCode.RightKey))
55                  {
56                      _spaceShip.Rotate(4);
57                  }
58                  if (SplashKit.KeyTyped(KeyCode.SpaceKey))
59                  {
60                      _spaceShip.Shoot();
61                  }
62
63                  _spaceShip.BulletFlying();
64                  Draw();
65              }
66              _gameWindow.Close();
67              _gameWindow = null;
68          }
69          private void Draw()
70          {
71              _gameWindow.Clear(Color.Black);
72              _spaceShip.Draw();
73              _gameWindow.Refresh(60);
74          }
75      }
76      public class SpaceShip
77      {
78          private double _x, _y;
79          private double _angle;
80          private Bitmap _shipBitmap;
81          private Bullet _bullet = new Bullet();
82
83          public SpaceShip()
84          {
85              Angle = 270;
86              _shipBitmap = SplashKit.BitmapNamed("Aquarii");
87          }
88
89          public double X
90          {
91              get { return _x; }
92              set { _x = value; }
93          }
94          public double Y
95          {
96              get { return _y; }
97              set { _y = value; }
98          }
99
100         public double Angle
101         {
102             get { return _angle; }
103             set { _angle = value; }
104         }
```

```
105
106         public void Rotate(double amount)
107         {
108             _angle = (_angle + amount) % 360;
109         }
110
111         public void Draw()
112         {
113             _shipBitmap.Draw(_x, _y, SplashKit.OptionRotateBmp(_angle));
114             _bullet.Draw();
115         }
116
117         public void Shoot()
118         {
119             Matrix2D anchorMatrix =
                ↪ SplashKit.TranslationMatrix(SplashKit.PointAt(_shipBitmap.Width /
                ↪ 2, _shipBitmap.Height / 2));
120
121             // Move centre point of picture to origin
122             Matrix2D result = SplashKit.MatrixMultiply(SplashKit.IdentityMatrix(),
                ↪ SplashKit.MatrixInverse(anchorMatrix));
123             // Rotate around origin
124             result = SplashKit.MatrixMultiply(result,
                ↪ SplashKit.RotationMatrix(_angle));
125             // Move it back...
126             result = SplashKit.MatrixMultiply(result, anchorMatrix);
127
128             // Now move to location on screen...
129             result = SplashKit.MatrixMultiply(result,
                ↪ SplashKit.TranslationMatrix(X, Y));
130
131             // Result can now transform a point to the ship's location
132             // Get right/centre
133             Vector2D vector = new Vector2D();
134             vector.X = _shipBitmap.Width;
135             vector.Y = _shipBitmap.Height / 2;
136             // Transform it...
137             vector = SplashKit.MatrixMultiply(result, vector);
138             _bullet = new Bullet(vector.X, vector.Y, Angle);
139         }
140
141         public void BulletFlying()
142         {
143             _bullet.FlyingBullet();
144         }
145
146         public void Move(double amountForward, double amountStrafe)
147         {
148             Vector2D movement = new Vector2D();
149             Matrix2D rotation = SplashKit.RotationMatrix(_angle);
150             movement.X += amountForward;
151             movement.Y += amountStrafe;
152             movement = SplashKit.MatrixMultiply(rotation, movement);
```

```
153             _x += movement.X;
154             _y += movement.Y;
155         }
156     }
157
158     public class Bullet
159     {
160         private Bitmap _bulletBitmap;
161         private double _x, _y, _angle;
162         private bool _active = false;
163         public Bullet(double x, double y, double angle)
164         {
165             _bulletBitmap = SplashKit.BitmapNamed("Bullet");
166             _x = x - _bulletBitmap.Width / 2;
167             _y = y - _bulletBitmap.Height / 2;
168             _angle = angle;
169             _active = true;
170         }
171
172         public Bullet()
173         {
174             _active = false;
175         }
176
177         public void FlyingBullet()
178         {
179             const int speed = 8;
180             Vector2D movement = new Vector2D();
181             Matrix2D rotation = SplashKit.RotationMatrix(_angle);  //Wrong
                ↪  ballistic angle here.
182             movement.X += speed;
183             movement = SplashKit.MatrixMultiply(rotation, movement);
184             _x += movement.X;
185             _y += movement.Y;
186             if ((_x > SplashKit.ScreenWidth() || _x < 0) || _y >
                ↪  SplashKit.ScreenHeight() || _y < 0)
187             { _active = false; }
188         }
189
190         public void Draw()
191         {
192             if (_active)
193             {
194                 DrawingOptions options = SplashKit.OptionRotateBmp(_angle);
195                 _bulletBitmap.Draw(_x, _y, options);
196             }
197
198         }
199     }
200 }
```


BlastOff

1. Having a code that works is certainly essential, while having a code that follows coding conventions, easy to ready, logical and elegant is also important.
2. The code with messy formatting, counter intuitive naming, wrong use of case and duplications can be hard to read.
3. It would be very important to follow the coding standards in a team project setting. While collaborating with the team members, having a clearly formatted and consistent code that follows the coding standards not only maximizes the efficiency and minimizes the chance of having errors due to misunderstanding and miscommunication.
4. For software development in general, consistently follow coding standards can enhance efficiency, allows others to spent least time possible to understand the code. It also makes bug rectification and maintenance a lot easier.
5. Coding refactoring means transforming a messy code to a clean code without adding in any other functions. In this task, except for the part of fixing the bugs, things like fixing poor naming and indentation are code refactoring, as these actions clean up the code and make it more readable.

16 Robot Dodge

Add dodging functionality to the robot class

Outcome	Weight
Build Programs	♦♦♦◊◊

This task demonstrates the design and the building process of a program and is backed up by relevant evidence.

Outcome	Weight
Design	♦◊◊◊◊

This task demonstrates the design and the building process of a program and is backed up by relevant evidence.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the design and the building process of a program and is backed up by relevant evidence.

Date	Author	Comment
2020/08/22 17:12	Yiyang Hou	Ready to Mark
2020/08/23 13:37	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Robot Dodge

Submitted By:
Yiyang Hou
houyiy
2020/08/22 17:12

Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦♦◊◊
Design	♦◊◊◊◊
Justify	♦◊◊◊◊

This task demonstrates the design and the building process of a program and is backed up by relevant evidence.

August 22, 2020

```
using System;
using SplashKitSDK;

public class Program
{
    public static void Main()
    {
        Window gameWindow = new Window("Robot Dodge", 888, 666);
        RobotDodge game = new RobotDodge(gameWindow);

        while( !(game.Quit || SplashKit.WindowCloseRequested(gameWindow)))
        {
            game.HandleInput();
            game.Update();
            game.Draw();
        }

        gameWindow.Close();
        gameWindow = null;

    }
}
```

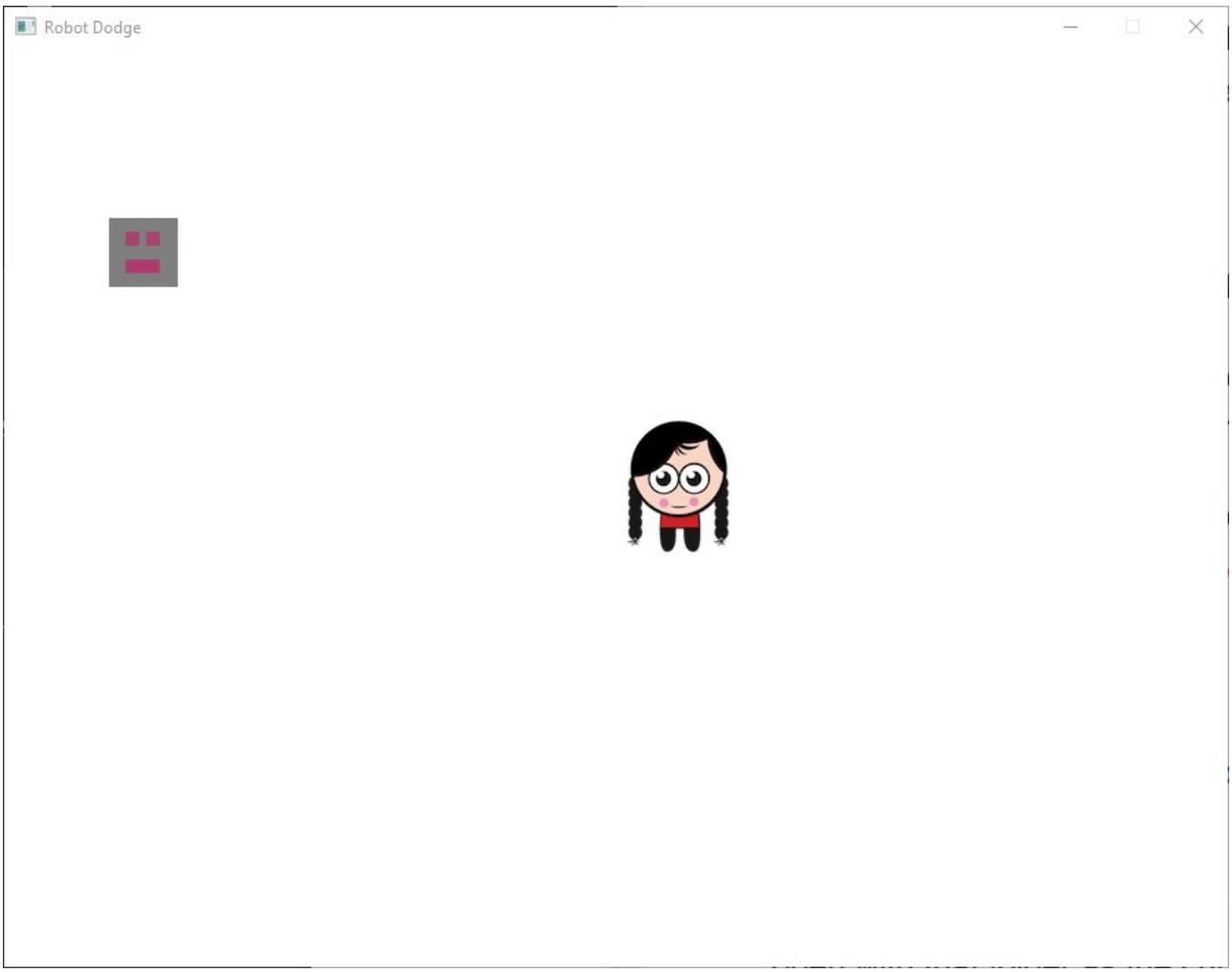
```
using SplashKitSDK;


public class Player
{
    private Bitmap _PlayerBitmap;

    double X {get;set;}
    double Y {get;set;}

    public bool Quit {get; private set;}


    public int Width
    {
        get
        {
            return _PlayerBitmap.Width;
        }
    }

     public int Height
    {
        get
        {
            return _PlayerBitmap.Height;
        }
    }

    public Player(Window windowObject)
    {
        _PlayerBitmap = new Bitmap("Player", "Player.png");
        Quit = false;

        X = (windowObject.Width - Width) / 2;
        Y = (windowObject.Height - Height) / 2;
    }

    public void Draw()
    {
        _PlayerBitmap.Draw(X, Y);
    }

    public void HandleInput()
    {
        int speed = 5;

        SplashKit.ProcessEvents();

        if(SplashKit.KeyReleased(KeyCode.EscapeKey)) Quit = true;

        if(SplashKit.KeyDown(KeyCode.LeftShiftKey) ||
        ↪  SplashKit.KeyDown(KeyCode.RightShiftKey))
```

```
53          {
54              speed = 15;
55          }
56
57          if(SplashKit.KeyDown(KeyCode.WKey) || SplashKit.KeyDown(KeyCode.UpKey))
58          {
59              Y -= speed;
60          }
61
62          if(SplashKit.KeyDown(KeyCode.SKey) || SplashKit.KeyDown(KeyCode.DownKey))
63          {
64              Y += speed;
65          }
66
67          if(SplashKit.KeyDown(KeyCode.AKey) || SplashKit.KeyDown(KeyCode.LeftKey))
68          {
69              X -= speed;
70          }
71
72          if(SplashKit.KeyDown(KeyCode.DKey) || SplashKit.KeyDown(KeyCode.RightKey))
73          {
74              X += speed;
75          }
76
77      }
78
79      public void StayOnWindow(Window limit)
80      {
81          const int GAP = 10;
82          if( X < GAP)
83          {
84              X = GAP;
85          }
86          if( X + Width > limit.Width - GAP)
87          {
88              X = limit.Width - GAP - Width;
89          }
90
91          if( Y < GAP)
92          {
93              Y = GAP;
94          }
95
96          if( Y + Height > limit.Height - GAP)
97          {
98              Y = limit.Height - GAP - Height;
99          }
100     }
101
102     public bool CollidedWith(Robot other)
103     {
104         return _PlayerBitmap.CircleCollision(X,Y, other.CollisionCircle);
105     }
```

```
106
107 }
```

```
using SplashKitSDK;

public class Robot
{
    public double X { get; set; }
    public double Y { get; set; }
    public Color MainColor { get; set; }
    public int Width
    {
        get{ return 50; }
    }
    public int Height
    {
        get{ return 50; }
    }
    public Circle CollisionCircle
    {
        get{ return SplashKit.CircleAt( X + 25, Y + 25, 35.36); }
    }

    public Robot( Window gameWindow )
    {
        X = SplashKit.Rnd( gameWindow.Width - Width );
        Y = SplashKit.Rnd( gameWindow.Height - Height );
        MainColor = Color.RandomRGB(200);
    }
    public void Draw()
    {
        double leftX, rightX;
        double eyeY, mouthY;

        leftX = X + 12;
        rightX = X + 27;
        eyeY = Y + 10;
        mouthY = Y + 30;
        SplashKit.FillRectangle( Color.Gray, X, Y, Width, Height);
        SplashKit.FillRectangle( MainColor, leftX, eyeY, 10, 10);
        SplashKit.FillRectangle( MainColor, rightX, eyeY, 10, 10);
        SplashKit.FillRectangle( MainColor, leftX, mouthY, 25, 10);
        SplashKit.FillRectangle( MainColor, leftX + 2, mouthY + 2, 21, 6);
    }
}
```

```
using SplashKitSDK;

public class RobotDodge
{
    private Player _Player;
    private Window _GameWindow;
    private Robot _TestRobot;

    public bool Quit
    {
        get
        {
            return _Player.Quit;
        }
    }

    public RobotDodge( Window gameWindow )
    {
        _GameWindow = gameWindow;
        _Player = new Player( gameWindow );
        _TestRobot = RandomRobot();
    }

    public void HandleInput()
    {
        _Player.HandleInput();
        _Player.StayOnWindow(_GameWindow);
    }

    public void Draw()
    {
        _GameWindow.Clear(Color.White);
        _TestRobot.Draw();
        _Player.Draw();
        _GameWindow.Refresh(60);
    }

    public void Update()
    {
        if( _Player.CollidedWith(_TestRobot) )
        {
            _TestRobot = RandomRobot();
        }
    }
    public Robot RandomRobot()
    {
        return new Robot(_GameWindow);
    }
}
```


Robot Dodge

17 Concept Visualisation 1

Visualise the programming concepts we've covered so far

Outcome	Weight
Principles	♦♦♦◊◊

This task demonstrates the understanding of principles and concepts of object-oriented programming.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the understanding of principles and concepts of object-oriented programming.

Date	Author	Comment
2020/08/23 18:05	Yiyang Hou	Ready to Mark
2020/08/23 20:35	Chetan Arora	Well done.
2020/08/23 20:35	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Concept Visualisation 1

Submitted By:
Yiyang Hou
houyiy
2020/08/23 18:05

Tutor:
Chetan Arora

Outcome	Weight
Principles	♦♦♦◊◊
Justify	♦◊◊◊◊

This task demonstrates the understanding of principles and concepts of object-oriented programming.

August 23, 2020

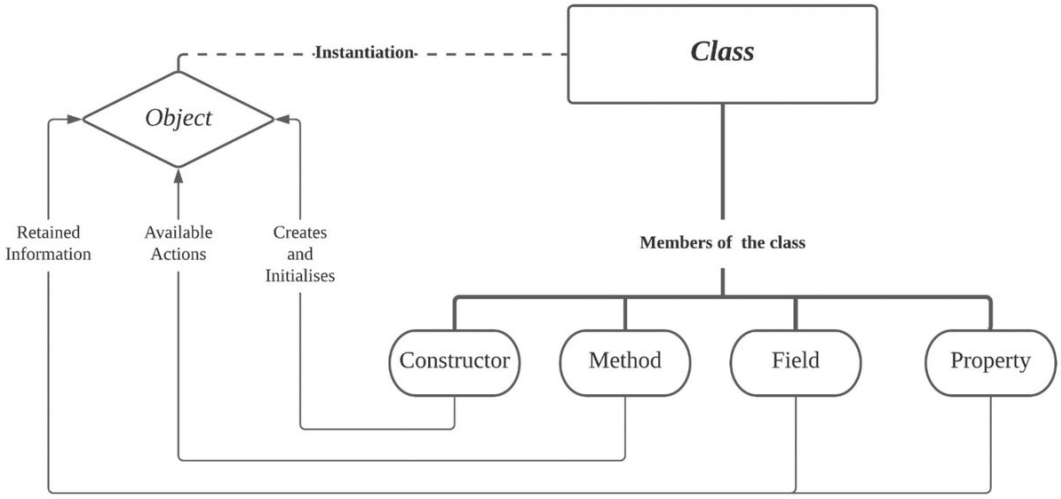


Figure 1. The Visualisation of The Concept of Class and Object

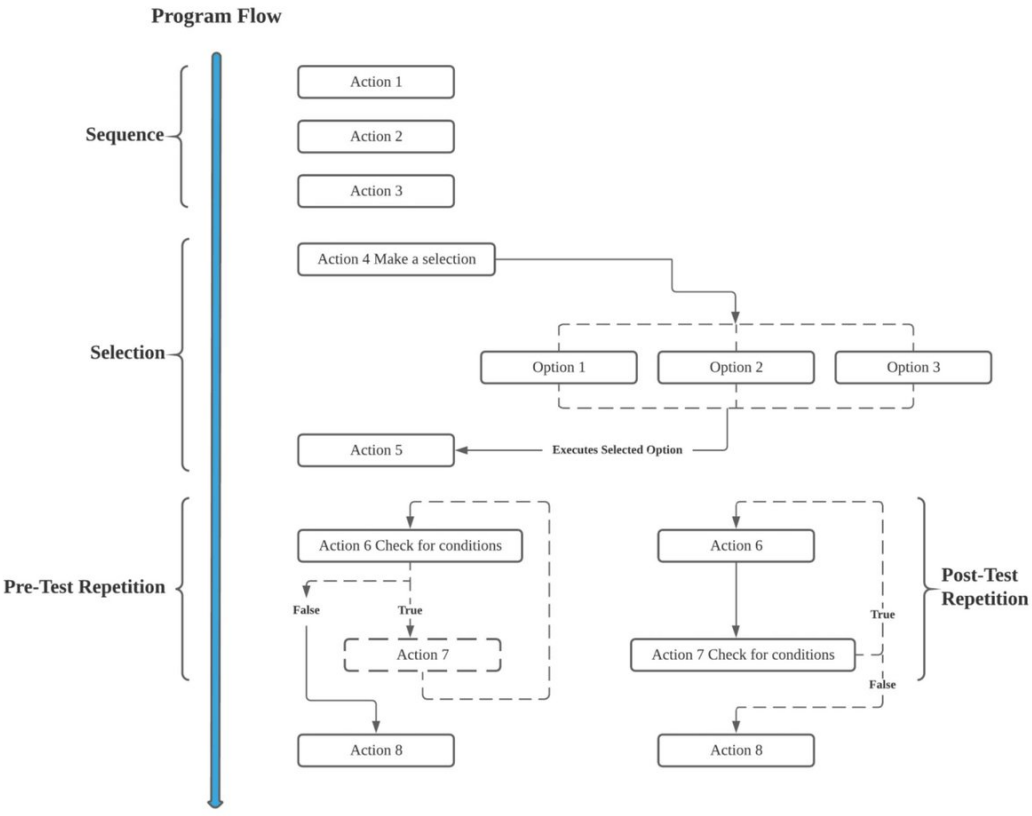


Figure 2. The Visualisation of The Concept of Control Flow

The first picture shows the visualisation of the concepts relating to class and object. We have learnt about class members including constructor, method, field and property. These members define what an object is, the information it contains and the actions the object can perform. Class is the blue print of an object and an object is an instantiation of a class. A new object can be created by calling the constructor using “new” keyword, each object has its own identity and information.

The second picture shows how programs run against time when using different control flows. Without any control flow statements, programs run in a sequence, meaning one action after another according to the order of the statements in the code. The selection statement means the program picks a branch and executes a block of code and then continues on afterwards. For example, switch statement is a selection statement. If statement is another form of selection, that verifies a condition and runs a block of code if the condition is true. It continues on or run another block of code if the condition is false. Repetition runs a block of code repetitively as long as a certain condition is true. If we check the condition before the first run is called Pre-test repetition, or we can check the condition after the first run is complete, then it is called Post-test repetition. The flow continues on when the condition is false.

18 Arrays

Create and manipulate arrays

Outcome	Weight
Build Programs	♦♦◊◊◊

This task demonstrates the implementation of a program.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the implementation of a program.

Date	Author	Comment
2020/08/30 23:17	Yiyang Hou	Can I ask for some more time to work on these tasks please
2020/09/05 18:34	Yiyang Hou	Ready to Mark
2020/09/08 00:38	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Arrays

Submitted By:
Yiyang Hou
houyiy
2020/09/05 18:34

Tutor:
Chetan Arora

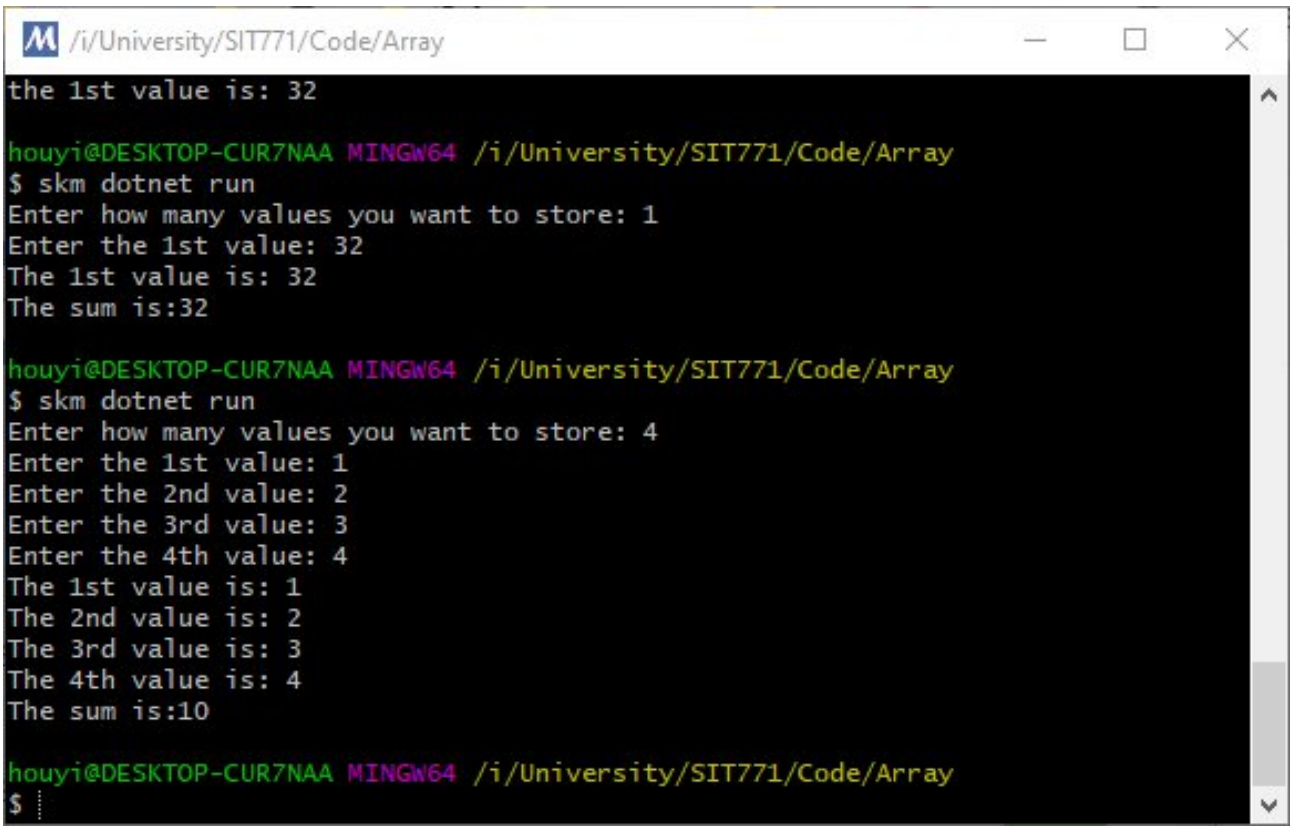
Outcome	**Weight**
Build Programs	♦♦◊◊◊
Justify	♦◊◊◊◊

This task demonstrates the implementation of a program.

September 5, 2020

```
using System;
using SplashKitSDK;

public class Program
{
    public static int ReadInteger(string prompt)
    {
        Console.Write(prompt);
        while (true)
        {
            try
            {
                return Int32.Parse(Console.ReadLine());
            }
            catch
            {
                Console.WriteLine("Please enter a valid integer");
            }
        }
    }
    public static double ReadDouble(string prompt)
    {
        Console.Write(prompt);
        while (true)
        {
            try
            {
                return double.Parse(Console.ReadLine());
            }
            catch
            {
                Console.WriteLine("Please enter a valid double number");
            }
        }
    }
    public static void Main()
    {
        int numberOfValues = ReadInteger($"Enter how many values you want to
        ↪  store: ");
        double[] values = new double[numberOfValues];
        for (int i = 0; i < numberOfValues; i++)
        {
            if( i == 0 )
            {
                values[i] = ReadDouble($"Enter the {i + 1}st value: ");
            }
            else if( i == 1)
            {
                values[i] = ReadDouble($"Enter the {i + 1}nd value: ");
            }
            else if( i == 2)
            {
                values[i] = ReadDouble($"Enter the {i + 1}rd value: ");
```

```
53              }
54              else
55              {
56              values[i] = ReadDouble($"Enter the {i + 1}th value: ");
57              }
58          }
59          for (int i = 0; i < numberOfValues; i++)
60          {
61              if( i == 0 )
62              {
63              Console.WriteLine($"The {i+1}st value is: {values[i]}");
64              }
65              else if( i == 1 )
66              {
67              Console.WriteLine($"The {i+1}nd value is: {values[i]}");
68              }
69              else if( i == 2 )
70              {
71              Console.WriteLine($"The {i+1}rd value is: {values[i]}");
72              }
73              else
74              {
75              Console.WriteLine($"The {i+1}th value is: {values[i]}");
76              }
77          }
78
79          double sum = 0;
80          for(int i = 0; i < numberOfValues; i++)
81          {
82              sum+=values[i];
83          }
84          Console.WriteLine($"The sum is:{sum}");
85
86
87      }
88  }
```


/i/University/SIT771/Code/Array
the 1st value is: 32
houyi@DESKTOP-CUR7NAA MINGW64 /i/University/SIT771/Code/Array
$ skm dotnet run
Enter how many values you want to store: 1
Enter the 1st value: 32
The 1st value is: 32
The sum is:32
houyi@DESKTOP-CUR7NAA MINGW64 /i/University/SIT771/Code/Array
$ skm dotnet run
Enter how many values you want to store: 4
Enter the 1st value: 1
Enter the 2nd value: 2
Enter the 3rd value: 3
Enter the 4th value: 4
The 1st value is: 1
The 2nd value is: 2
The 3rd value is: 3
The 4th value is: 4
The sum is:10
houyi@DESKTOP-CUR7NAA MINGW64 /i/University/SIT771/Code/Array
$

19 Lists

Create and manipulate lists

Outcome	Weight
Build Programs	♦♦◊◊◊

This task demonstrates the implementation of a program.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the implementation of a program.

Date	Author	Comment
2020/08/30 23:18	Yiyang Hou	Can I ask for some more time to work on these tasks please
2020/09/05 20:06	Yiyang Hou	Ready to Mark
2020/09/08 00:42	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Lists

Submitted By:
Yiyang Hou
houyiy
2020/09/05 20:06

Tutor:
Chetan Arora

Outcome	Weight
Build Programs	♦♦◊◊◊
Justify	♦◊◊◊◊

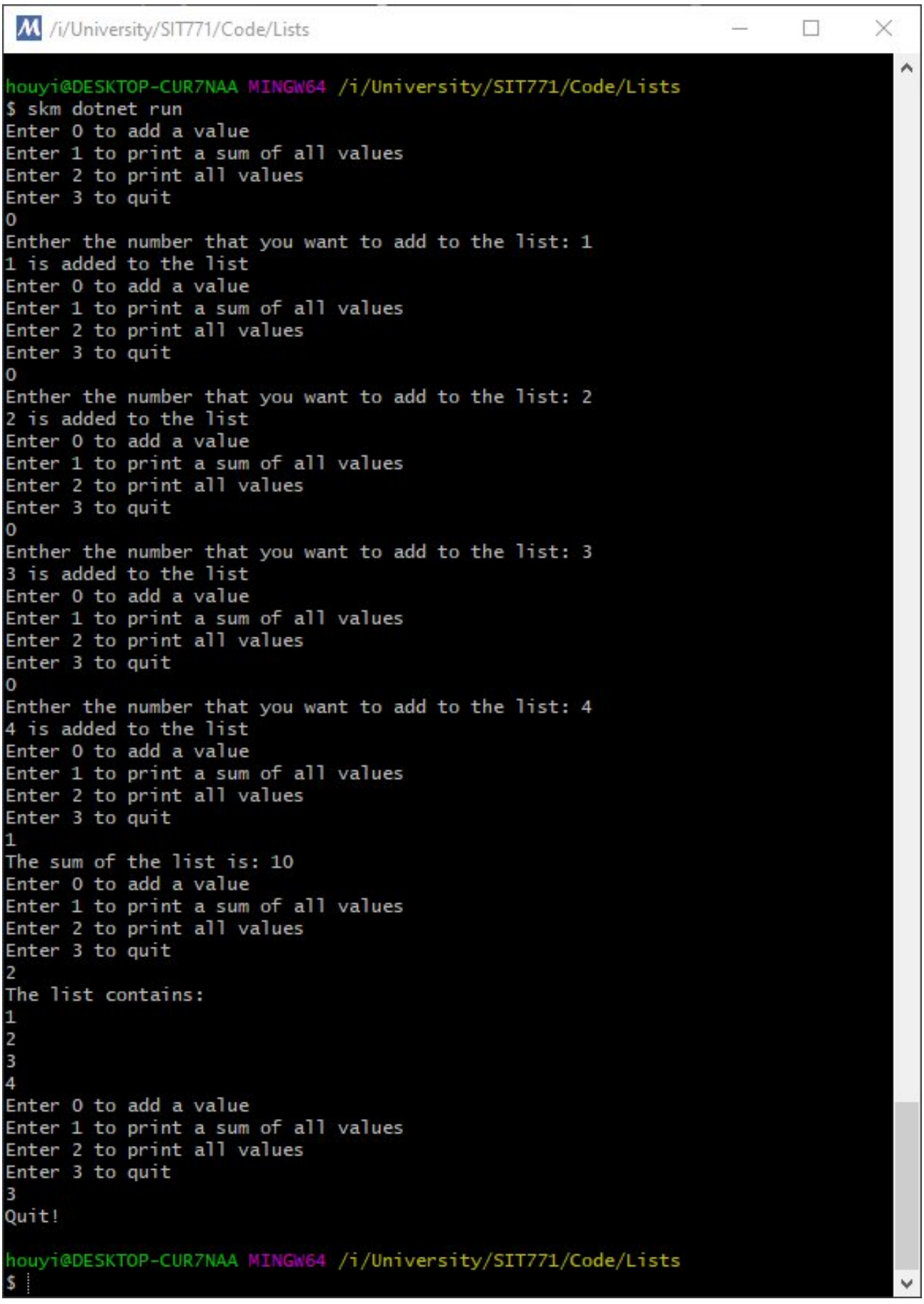
This task demonstrates the implementation of a program.

September 5, 2020

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

public class Program
{
    private static List<double> _values = new List<double>();
    public static int ReadInteger(string prompt)
    {
        Console.Write(prompt);
        while (true)
        {
            try
            {
                return Int32.Parse(Console.ReadLine());
            }
            catch
            {
                Console.WriteLine("Please enter a valid integer");
            }
        }
    }
    public static double ReadDouble(string prompt)
    {
        Console.Write(prompt);
        while (true)
        {
            try
            {
                return double.Parse(Console.ReadLine());
            }
            catch
            {
                Console.WriteLine("Please enter a valid double number");
            }
        }
    }

    public enum UserOption
    {
        NewValue,
        Sum,
        Print,
        Quit
    }
    public static void AddValueToList()
    {
        double addValue = ReadDouble("Enther the number that you want to add to the
        ↪  list: ");
        _values.Add(addValue);
        Console.WriteLine($"{addValue} is added to the list");
    }
    public static void Print()
```

```
53      {
54          Console.WriteLine("The list contains:");
55          foreach( double listElement in _values)
56          {
57              Console.WriteLine(listElement);
58          }
59      }
60
61      public static void Sum()
62      {
63          double sum = 0;
64          foreach( double listElement in _values)
65          {
66              sum += listElement;
67          }
68          Console.WriteLine( $"The sum of the list is: {sum}");
69      }
70
71      public static UserOption ReadUserOption()
72      {
73          Console.WriteLine("Enter 0 to add a value");
74          Console.WriteLine("Enter 1 to print a sum of all values");
75          Console.WriteLine("Enter 2 to print all values");
76          Console.WriteLine("Enter 3 to quit");
77
78          int option = 3;
79          Int32.TryParse(Console.ReadLine(), out option);
80
81          return (UserOption)option;
82      }
83      public static void Main()
84      {
85          UserOption userOption;
86          do
87          {
88              userOption = ReadUserOption();
89              switch(userOption)
90              {
91                  case UserOption.NewValue:
92                  AddValueToList();
93                  break;
94
95                  case UserOption.Sum:
96                  Sum();
97                  break;
98
99                  case UserOption.Print:
100                 Print();
101                 break;
102
103                 case UserOption.Quit:
104                 Console.WriteLine("Quit!");
105                 break;
```

```
106                 }
107             }
108             while( userOption != UserOption.Quit);
109         }
110 }
```


/i/University/SIT771/Code/Lists
houyi@DESKTOP-CUR7NAA MINGW64 /i/University/SIT771/Code/Lists
$ skm dotnet run
Enter 0 to add a value
Enter 1 to print a sum of all values
Enter 2 to print all values
Enter 3 to quit
0
Enther the number that you want to add to the list: 1
1 is added to the list
Enter 0 to add a value
Enter 1 to print a sum of all values
Enter 2 to print all values
Enter 3 to quit
0
Enther the number that you want to add to the list: 2
2 is added to the list
Enter 0 to add a value
Enter 1 to print a sum of all values
Enter 2 to print all values
Enter 3 to quit
0
Enther the number that you want to add to the list: 3
3 is added to the list
Enter 0 to add a value
Enter 1 to print a sum of all values
Enter 2 to print all values
Enter 3 to quit
0
Enther the number that you want to add to the list: 4
4 is added to the list
Enter 0 to add a value
Enter 1 to print a sum of all values
Enter 2 to print all values
Enter 3 to quit
1
The sum of the list is: 10
Enter 0 to add a value
Enter 1 to print a sum of all values
Enter 2 to print all values
Enter 3 to quit
2
The list contains:
1
2
3
4
Enter 0 to add a value
Enter 1 to print a sum of all values
Enter 2 to print all values
Enter 3 to quit
3
Quit!
houyi@DESKTOP-CUR7NAA MINGW64 /i/University/SIT771/Code/Lists
$

20 Many Accounts

Many Accounts

Outcome	Weight
Build Programs	♦♦♦◊◊

This task demonstrates the process of building a program, including the design process and the relevant evidence is provided.

Outcome	Weight
Design	♦◊◊◊◊

This task demonstrates the process of building a program, including the design process and the relevant evidence is provided.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the process of building a program, including the design process and the relevant evidence is provided.

Date	Author	Comment
2020/08/30 23:18	Yiyang Hou	Can I ask for some more time to work on these tasks please
2020/09/07 00:45	Yiyang Hou	Ready to Mark
2020/09/07 00:45	Yiyang Hou	Task Discussion:1.When having multiple accounts, all of the accounts are stored in the _accounts list. More accounts can be added to the list by calling AddAccount(), GetAccount() can be called to select the desired account object from the list by typing in the matching account name.2.In the current version, all the account objects are created on demand and are stored in the _accounts list, therefore any accounts that are in the list can perform deposit and withdrawal.3.In the current version, DoDeposit(), DoWithdraw() and DoTransfer() specify their target accounts on demand by calling FindAccount(), it gives the user the option to choose any accounts from the _accounts list by typing in the matching account name. If no account is found in any of these steps, it will return to the menu option.
2020/09/08 21:53	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Many Accounts

Submitted By:
Yiyang Hou
houyiy
2020/09/07 00:45

Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦♦◊◊
Design	♦◊◊◊◊
Justify	♦◊◊◊◊

This task demonstrates the process of building a program, including the design process and the relevant evidence is provided.

September 7, 2020

```
using System;
using SplashKitSDK;

public enum MenuOption
{
    Withdraw,
    Deposit,
    Print,
    Transfer,
    NewAccount,
    Quit
}
public class Program
{
    public static void Main()
    {
        Bank bank = new Bank();
        MenuOption userSelection;
       do
       {
            userSelection = ReadUserOption();
            switch(userSelection)
            {
                case MenuOption.Withdraw:
                DoWithdraw(bank);
                break;

                case MenuOption.Deposit:
                DoDeposit(bank);
                break;

                case MenuOption.Print:
                DoPrint(bank);
                break;

                case MenuOption.Transfer:
                DoTransfer(bank);
                break;

                case MenuOption.NewAccount:
                DoNewAccount(bank);
                break;

                case MenuOption.Quit:
                Console.WriteLine("QUIT!");
                break;
            }
       }
       while( userSelection != MenuOption.Quit);

    }

    public static MenuOption ReadUserOption()
```

```
54      {
55          int usersOption = 0;
56
57          do
58          {
59              Console.WriteLine("*****Menu***** \n-------------------\n1.
                ↪  ---Withdraw---\n2. ---Deposit---\n3. ---Print---\n4.
                ↪  ---Transfer---\n5. ---New Account---\n6.
                ↪  ---Quit---\n--------------------");
60              Console.Write("What would you like to do today: ");
61              string userInput = Console.ReadLine();
62              try{
63                  usersOption = Convert.ToInt32(userInput);
64                  if (usersOption < 1 || usersOption > 6)
65                  {
66                      Console.WriteLine("Invalid Input, please try again!\n");
67                  }
68              }
69              catch(Exception e)
70              {
71                  Console.WriteLine( $ "{e.Message}\n");
72                  usersOption = -1;
73              }
74          }
75          while(usersOption < 1 || usersOption > 6);
76          return (MenuOption)(usersOption - 1);
77      }
78
79
80      public static void DoWithdraw(Bank toBank)
81      {
82          Account toAccount = FindAccount(toBank);
83          if ( toAccount == null) return;
84          decimal withdrawAmount = 0;
85          Console.Write("How much would you like to WITHDRAW? Please enter: $");
86          string userWithdraw = Console.ReadLine();
87          try
88          {
89              withdrawAmount = Convert.ToDecimal(userWithdraw);
90              WithdrawTransaction withdrawTransaction = new
                ↪  WithdrawTransaction(toAccount, withdrawAmount);
91              toBank.ExecuteTransaction(withdrawTransaction);
92              withdrawTransaction.Print();
93          }
94          catch(Exception e)
95          {
96              Console.WriteLine( $ "{e.Message}\n");
97          }
98      }
99
100     public static void DoDeposit(Bank toBank)
101     {
102         Account toAccount = FindAccount(toBank);
```

```
103         if ( toAccount == null ) return;
104         decimal depositAmount = 0;
105         Console.Write("How much would you like to DEPOSIT? Please enter: $");
106         string userDeposit = Console.ReadLine();
107         try
108         {
109             depositAmount = Convert.ToDecimal(userDeposit);
110             DepositTransaction depositTransaction = new
                ↪  DepositTransaction(toAccount, depositAmount);
111             toBank.ExecuteTransaction(depositTransaction);
112             depositTransaction.Print();
113         }
114         catch(Exception e)
115         {
116             Console.WriteLine( $"{e.Message}\n");
117         }
118     }
119
120     public static void DoPrint(Bank bank)
121     {
122         Account account = FindAccount( bank );
123         if( account == null ) return;
124         account.Print();
125     }
126
127     public static void DoTransfer(Bank bank)
128     {
129         Account fromAccount = FindAccount (bank);
130         Account toAccount = FindAccount (bank);
131         if (fromAccount == null || toAccount == null) return;
132         decimal transferAmount = 0;
133         Console.Write( $"How much would you like to TRANSFER from
            ↪  {fromAccount.Name}'s account to {toAccount.Name}'s account? Please
            ↪  enter: $");
134         string userTransfer = Console.ReadLine();
135         try
136         {
137             transferAmount = Convert.ToDecimal(userTransfer);
138             TransferTransaction transferTransaction = new
                ↪  TransferTransaction(fromAccount, toAccount, transferAmount);
139             bank.ExecuteTransaction(transferTransaction);
140             transferTransaction.Print();
141         }
142         catch(Exception e)
143         {
144             Console.WriteLine( $"{e.Message}\n");
145         }
146     }
147
148     public static void DoNewAccount( Bank bank )
149     {
150         decimal startingBalance = 0;
151         Console.Write("Enter the name for the new account: ");
```

```
152         string name = Console.ReadLine();
153         Console.Write("Enter the starting balance for the new account: ");
154         try
155         {
156             startingBalance = decimal.Parse(Console.ReadLine());
157         }
158         catch
159         {
160             Console.WriteLine("Please enter a valid decimal number");
161         }
162 
163         Account account = new Account(name, startingBalance);
164         bank.AddAccount( account );
165         Console.WriteLine("New account created.");
166     }
167 
168     private static Account FindAccount( Bank fromBank )
169     {
170         Console.Write("Enter account name: ");
171         string name = Console.ReadLine();
172         Account result = fromBank.GetAccount(name);
173 
174         if ( result == null )
175         {
176             Console.WriteLine( $"No account found with the name {name}");
177         }
178 
179         return result;
180     }
181 }
```

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

public class Bank
{
    private List<Account> _accounts = new List<Account>();

    public void AddAccount( Account account )
    {
        _accounts.Add(account);
    }
    public Account GetAccount( string name )
    {
        for( int i = 0 ; i < _accounts.Count; i++ )
        {
            if(name == _accounts[i].Name)
            {
                return _accounts[i];
            }
        }
        return null;
    }

    public void ExecuteTransaction( WithdrawTransaction transaction)
    {
        transaction.Execute();
    }

    public void ExecuteTransaction( DepositTransaction transaction)
    {
        transaction.Execute();
    }

    public void ExecuteTransaction( TransferTransaction transaction)
    {
        transaction.Execute();
    }
}
```

```
using System;


public class Account
{
    private decimal _balance;
    private string _name;

    //Constructor
    public Account(string name, decimal startingBalance)
    {
        _name = name;
        _balance = startingBalance;

    }

    //Deposit method
    public bool Deposit(decimal amountToDeposit)
    {
        if (amountToDeposit > 0)
        {
            _balance += amountToDeposit;
            return true;
        }
        else
        {
            Console.WriteLine("**WARNING**\nThe deposit amount must be a positive
            ↪  number!");
            return false;
        }
    }

    //Withdraw mathod
    public bool Withdraw(decimal amountToWithdraw)
    {
        if(amountToWithdraw > 0 && amountToWithdraw <= _balance )
        {
            _balance -= amountToWithdraw;
            return true;
        }
        else if (amountToWithdraw <= 0)
        {
            Console.WriteLine("**WARNING**\nThe withdraw amount must be a positive
            ↪  number!");
            return false;
        }
        else
        {
            Console.WriteLine("**WARNING**\nThe withdraw amount must not exceed
            ↪  your exiting funds!");
            return false;
        }

```

```
51      }
52
53      //Read name property
54      public string Name
55      {
56          get { return _name; }
57      }
58
59      //Print method
60      public void Print()
61      {
62          Console.WriteLine($"Account Name: {Name}\nBalance: {_balance}\n");
63      }
64  }
```


C:\Users\Yiyang\Desktop\University\SIT771\Code\BankProgram_5.3
Yiyang@DESKTOP-CU87NAA MINGW64 /i/University/SIT771/Code/BankProgram_5.3
$ sln dotnet run
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---New Account---
6. ---Quit---
----
what would you like to do today: 5
Enter the name for the new account: Yiyang
Enter the starting balance for the new account: 10000
New account created.
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---New Account---
6. ---Quit---
----
what would you like to do today: 5
Enter the name for the new account: Tom
Enter the starting balance for the new account: 6000
New account created.
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---New Account---
6. ---Quit---
----
what would you like to do today: 4
Enter account name: Yiyang
Enter account name: Tom
How much would you like to TRANSFER from Yiyang's account to Tom's account? Please enter: $3000
"$3000 was transferred from Yiyang's Account to Tom's account.
    "$3000 was SUCCESSFULLY withdrawed from Yiyang's account.|The transaction was not rolled back.
    "$3000 was SUCCESSFULLY deposited into Tom's account.|The transaction was not rolled back.
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---New Account---
6. ---Quit---
----
what would you like to do today: 1
Enter account name: Yiyang
How much would you like to WITHDRAW? Please enter: $2000
"$2000 was SUCCESSFULLY withdrawed from Yiyang's account.|The transaction was not rolled back.
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---New Account---
6. ---Quit---
----
what would you like to do today: 2
Enter account name: Yiyang
How much would you like to DEPOSIT? Please enter: $1000
"$1000 was SUCCESSFULLY deposited into Yiyang's account.|The transaction was not rolled back.
*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---New Account---
6. ---Quit---
----
what would you like to do today: 3
Enter account name: Yiyang
Account Name: Yiyang
Balance: 6000

*****Menu*****
----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---New Account---
6. ---Quit---
----
what would you like to do today: 6
QUIT!

Yiyang@DESKTOP-CU87NAA MINGW64 /i/University/SIT771/Code/BankProgram_5.3
$

21 Many Robots

Add many robots to your program

Outcome	Weight
Build Programs	♦♦♦◊◊

This task demonstrates the building process of a program as well as the design process. Relevant evidence is also provided.

Outcome	Weight
Design	♦◊◊◊◊

This task demonstrates the building process of a program as well as the design process. Relevant evidence is also provided.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the building process of a program as well as the design process. Relevant evidence is also provided.

Date	Author	Comment
2020/09/07 22:02	Yiyang Hou	Can I have more time to work on these tasks please
2020/09/08 01:09	Yiyang Hou	Ready to Mark
2020/09/08 21:59	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Many Robots

Submitted By:
Yiyang Hou
houyiy
2020/09/08 01:09

Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦♦◇◇
Design	♦◇◇◇◇
Justify	♦◇◇◇◇

This task demonstrates the building process of a program as well as the design process. Relevant evidence is also provided.

September 8, 2020

```
using System;
using SplashKitSDK;

public class Program
{
    public static void Main()
    {
        Window gameWindow = new Window("Robot Dodge", 888, 666);
        RobotDodge game = new RobotDodge(gameWindow);

        while( !(game.Quit || SplashKit.WindowCloseRequested(gameWindow)))
        {
            game.HandleInput();
            game.Update();
            game.Draw();
        }

        gameWindow.Close();
        gameWindow = null;

    }
}
```

```
using SplashKitSDK;


public class Player
{
    private Bitmap _PlayerBitmap;

    public double X {get;set;}
    public double Y {get;set;}

    public bool Quit {get; private set;}


    public int Width
    {
        get
        {
            return _PlayerBitmap.Width;
        }
    }

     public int Height
    {
        get
        {
            return _PlayerBitmap.Height;
        }
    }

    public Player(Window windowObject)
    {
        _PlayerBitmap = new Bitmap("Player", "Player.png");
        Quit = false;

        X = (windowObject.Width - Width) / 2;
        Y = (windowObject.Height - Height) / 2;
    }

    public void Draw()
    {
        _PlayerBitmap.Draw(X, Y);
    }

    public void HandleInput()
    {
        int speed = 5;

        SplashKit.ProcessEvents();

        if(SplashKit.KeyReleased(KeyCode.EscapeKey)) Quit = true;

        if(SplashKit.KeyDown(KeyCode.LeftShiftKey) ||
        ↪  SplashKit.KeyDown(KeyCode.RightShiftKey))
```

```
53          {
54              speed = 15;
55          }
56
57          if(SplashKit.KeyDown(KeyCode.WKey) || SplashKit.KeyDown(KeyCode.UpKey))
58          {
59              Y -= speed;
60          }
61
62          if(SplashKit.KeyDown(KeyCode.SKey) || SplashKit.KeyDown(KeyCode.DownKey))
63          {
64              Y += speed;
65          }
66
67          if(SplashKit.KeyDown(KeyCode.AKey) || SplashKit.KeyDown(KeyCode.LeftKey))
68          {
69              X -= speed;
70          }
71
72          if(SplashKit.KeyDown(KeyCode.DKey) || SplashKit.KeyDown(KeyCode.RightKey))
73          {
74              X += speed;
75          }
76
77      }
78
79      public void StayOnWindow(Window limit)
80      {
81          const int GAP = 10;
82          if( X < GAP)
83          {
84              X = GAP;
85          }
86          if( X + Width > limit.Width - GAP)
87          {
88              X = limit.Width - GAP - Width;
89          }
90
91          if( Y < GAP)
92          {
93              Y = GAP;
94          }
95
96          if( Y + Height > limit.Height - GAP)
97          {
98              Y = limit.Height - GAP - Height;
99          }
100     }
101
102     public bool CollidedWith(Robot other)
103     {
104         return _PlayerBitmap.CircleCollision(X,Y, other.CollisionCircle);
105     }
```

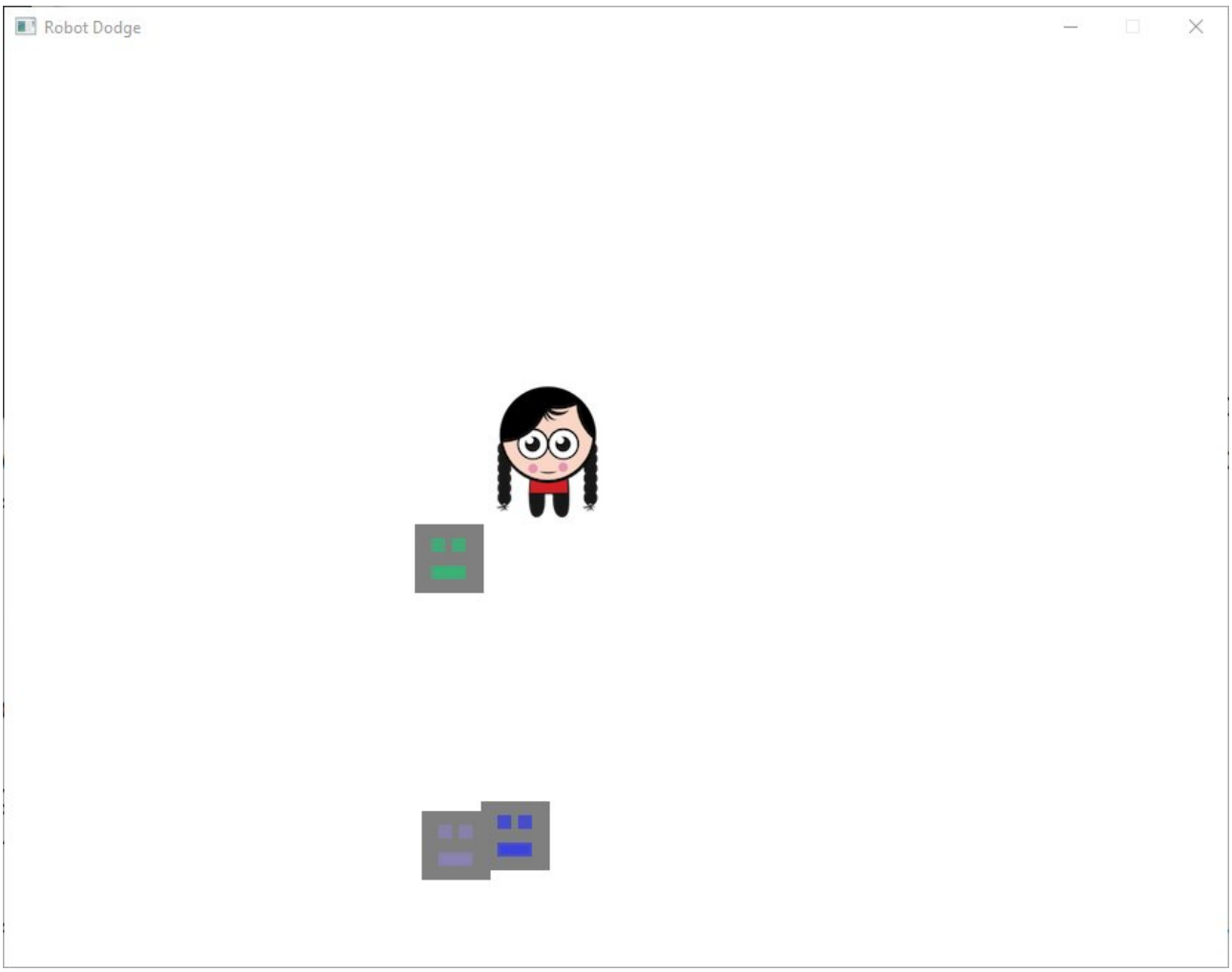
```
106
107 }
```

```
using SplashKitSDK;

public class Robot
{
    public double X { get; set; }
    public double Y { get; set; }
    private Vector2D Velocity { get; set; }
    public Color MainColor { get; set; }
    public int Width
    {
        get{ return 50; }
    }
    public int Height
    {
        get{ return 50; }
    }
    public Circle CollisionCircle
    {
        get{ return SplashKit.CircleAt( X + 25, Y + 25, 35.36); }
    }

    public Robot( Window gameWindow, Player player )
    {
        MainColor = Color.RandomRGB(200);
        if (SplashKit.Rnd() < 0.5 )
        {
            X = SplashKit.Rnd(gameWindow.Width);

            if (SplashKit.Rnd() < 0.5) Y = -Height;
            else Y = gameWindow.Height;
        }
        else
        {
            Y = SplashKit.Rnd(gameWindow.Height);

            if (SplashKit.Rnd() < 0.5 ) X = -Width;
            else X = gameWindow.Width;
        }

        const int SPEED = 4;

        Point2D fromPt = new Point2D()
        {
            X = X, Y = Y
        };

        Point2D toPt = new Point2D()
        {
            X= player.X, Y = player.Y
        };

        Vector2D dir;
        dir = SplashKit.UnitVector(SplashKit.VectorPointToPoint( fromPt, toPt ));
```

```
54
55          Velocity = SplashKit.VectorMultiply( dir, SPEED );
56
57      }
58      public void Draw()
59      {
60          double leftX, rightX;
61          double eyeY, mouthY;
62
63          leftX = X + 12;
64          rightX = X + 27;
65          eyeY = Y + 10;
66          mouthY = Y + 30;
67          SplashKit.FillRectangle( Color.Gray, X, Y, Width, Height);
68          SplashKit.FillRectangle( MainColor, leftX, eyeY, 10, 10);
69          SplashKit.FillRectangle( MainColor, rightX, eyeY, 10, 10);
70          SplashKit.FillRectangle( MainColor, leftX, mouthY, 25, 10);
71          SplashKit.FillRectangle( MainColor, leftX + 2, mouthY + 2, 21, 6);
72      }
73
74      public void Update()
75      {
76          X += Velocity.X;
77          Y += Velocity.Y;
78      }
79
80      public bool IsOffscreen( Window screen )
81      {
82          while( X < -Width || X > screen.Width || Y < -Height || Y > screen.Height )
83          {
84              return true;
85          }
86          return false;
87
88      }
89  }
```

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

public class RobotDodge
{
    private Player _Player;
    private Window _GameWindow;
    private List<Robot> _Robots = new List<Robot>();
    public bool Quit
    {
        get
        {
            return _Player.Quit;
        }
    }

    public RobotDodge( Window gameWindow )
    {
        _GameWindow = gameWindow;
        _Player = new Player( gameWindow );
    }

    public void HandleInput()
    {
        _Player.HandleInput();
        _Player.StayOnWindow(_GameWindow);
    }

    public void Draw()
    {
        _GameWindow.Clear(Color.White);
        foreach(Robot eachRobot in _Robots)
        {
            eachRobot.Draw();
        }
        _Player.Draw();
        _GameWindow.Refresh(60);
    }

    public void Update()
    {
        foreach(Robot eachRobot in _Robots)
        {
            eachRobot.Update();
        }

        if (SplashKit.Rnd()<0.01)
        {
            Robot newRobot = RandomRobot();
            _Robots.Add(newRobot);
        }

        CheckCollisions();
```

```
54      }
55      public Robot RandomRobot()
56      {
57          return new Robot(_GameWindow,_Player);
58      }
59
60      private void CheckCollisions()
61      {
62          List<Robot> removeRobots = new List<Robot>();
63          foreach(Robot eachRobot in _Robots)
64          {
65              if (_Player.CollidedWith(eachRobot) ||
                ↪  eachRobot.IsOffscreen(_GameWindow))
66              {
67                  removeRobots.Add(eachRobot);
68              }
69          }
70          foreach(Robot eachRemoveRobot in removeRobots)
71          {
72              _Robots.Remove(eachRemoveRobot);
73          }
74      }
75  }
```


Robot Dodge

22 Document Design

Create appropriate documentation for the Account class

Outcome	Weight
Evaluate Code	♦♦◊◊◊

This task demonstrates the evaluation of code as well as the design process of a program and is backed up with relevant evidence.

Outcome	Weight
Design	♦♦♦♦◊

This task demonstrates the evaluation of code as well as the design process of a program and is backed up with relevant evidence.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the evaluation of code as well as the design process of a program and is backed up with relevant evidence.

Date	Author	Comment
2020/09/07 22:02	Yiyang Hou	Can I have more time to work on these tasks please
2020/09/13 15:51	Yiyang Hou	Ready to Mark
2020/09/13 15:51	Yiyang Hou	Task discussion:1.In the class diagram, classes are presented as boxes, fields are placed in the 2nd portion of the box, while the methods go into the 3rd portion of the box. Dependency relationship is presented as a dashed arrow, association relationship is a solid arrow and aggregation relationship is a solid arrow with a diamond on the end of the whole side of the association.2.The diagram is the static structure of the program, therefore the code must include all the classes, fields, properties and methods that are presented in the diagram and follow the exact structure.3.We can use the class diagram to think through the solution by visualising the roles, the responsibilities of the roles and the collaborations between the roles. The advantage of doing this is that it gives the developer a big picture of the solution and can be used as a guide to implementation.
2020/09/15 23:19	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Document Design

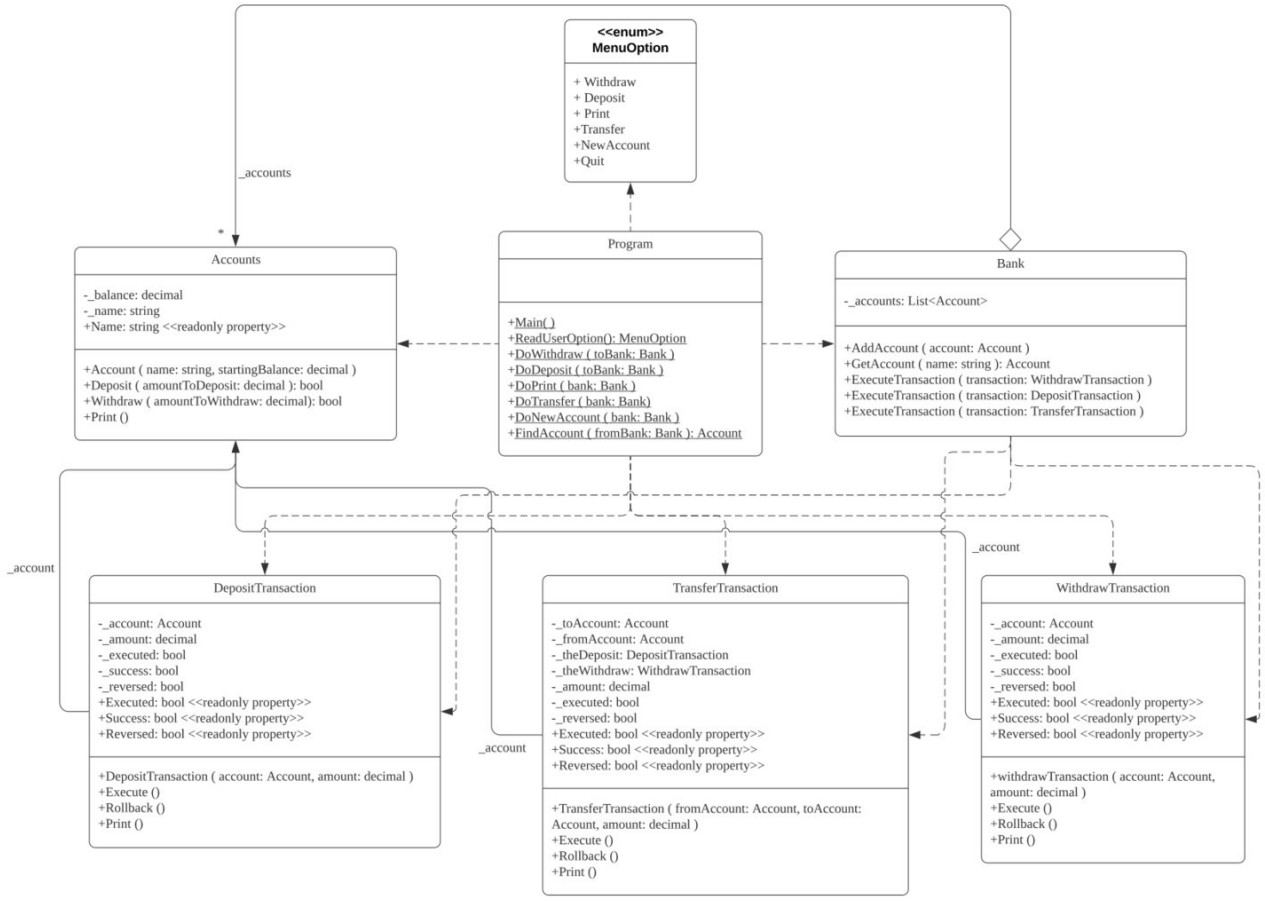
Submitted By:
Yiyang Hou
houyiy
2020/09/13 15:51

Tutor:
Chetan Arora

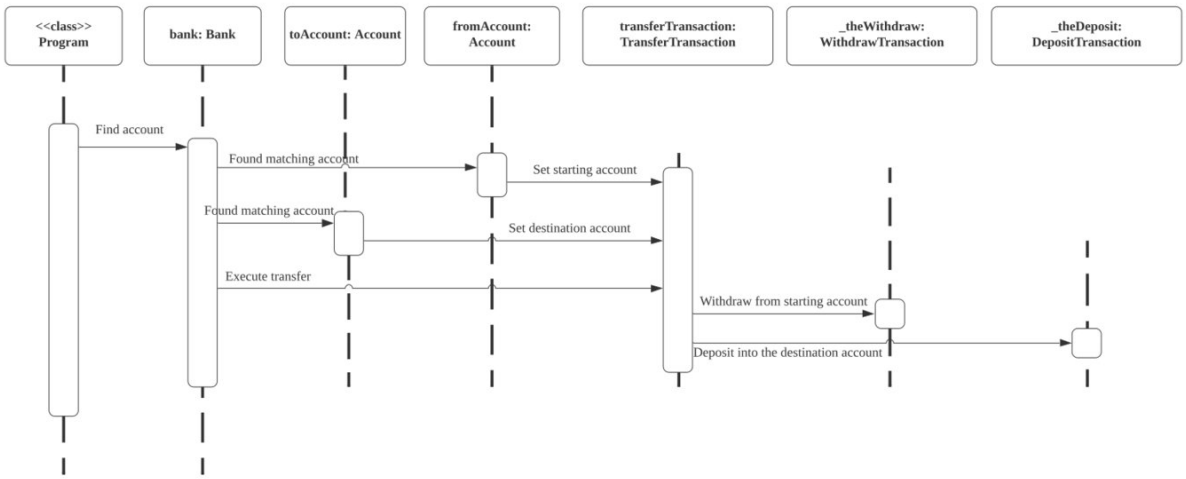
Outcome	**Weight**
Evaluate Code	♦♦◊◊◊
Design	♦♦♦♦◊
Justify	♦◊◊◊◊

This task demonstrates the evaluation of code as well as the design process of a program and is backed up with relevant evidence.

September 13, 2020



Class Diagram



Sequence Diagram for Transaction

23 Abstract Transactions

Change all transactions to be abstract

Outcome	Weight
Build Programs	♦♦♦◊◊

This task demonstrates the process of designing and building a program and is backed up with relevant evidence.

Outcome	Weight
Design	♦♦◊◊◊

This task demonstrates the process of designing and building a program and is backed up with relevant evidence.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the process of designing and building a program and is backed up with relevant evidence.

Date	Author	Comment
2020/09/14 00:14	Yiyang Hou	Can I have more time to work on these tasks please
2020/09/19 20:05	Yiyang Hou	Ready to Mark
2020/09/19 20:05	Yiyang Hou	1.In this solution, TransferTransaction, DepositTransaction and WithdrawTransaction are inherited from the abstract class Transaction, they pass up the amount values to the transaction constructor, all these demonstrate the concept of inheritance. In terms of polymorphism, when the virtual and abstract methods in the base class are called, due to the different implementations in the child classes, even with the same method called, the execution will be different according to the specific "kind" of transaction. In addition to that, is that the transaction list can contain elements of all kinds of transactions and each of them is able to perform the right type of Execute() and Print() of their own kind.2.If the method is virtual or abstract, then each child class is able to perform their own version of that method by using override.3.No changes are required for the Bank class in order to include a new transaction type. As long as the new type is a class that is inherited from Transaction, then the transaction list in the Bank class is able to contain that new type and perform any methods specifically for that particular type.4.We can get rid of a lot of the duplication in our code, because all the transaction types share a lot of the attributes, with inheritance, we can put these attributes in one class for every type to share.5.Polymorphism on the other hand allows all child classes to still have their uniqueness when required while sharing the attributes and actions that are in common.
2020/09/20 23:43	Chetan Arora	Discuss
2020/09/21 17:43	Yiyang Hou	Sure. So I guess I just need to do the interview with you in real time instead of answering questions in discussion mode on OnTrack?
2020/09/25 17:56	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Abstract Transactions

Submitted By:
Yiyang Hou
houyiy
2020/09/19 20:05

Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦♦◊◊
Design	♦♦◊◊◊
Justify	♦◊◊◊◊

This task demonstrates the process of designing and building a program and is backed up with relevant evidence.

September 19, 2020

```
using System;
using SplashKitSDK;

public enum MenuOption
{
    Withdraw,
    Deposit,
    Print,
    Transfer,
    NewAccount,
    TransactionHistory,
    Quit
}
public class Program
{
    public static void Main()
    {
        Bank bank = new Bank();
        MenuOption userSelection;
       do
       {
            userSelection = ReadUserOption();
            switch(userSelection)
            {
                case MenuOption.Withdraw:
                DoWithdraw(bank);
                break;

                case MenuOption.Deposit:
                DoDeposit(bank);
                break;

                case MenuOption.Print:
                DoPrint(bank);
                break;

                case MenuOption.Transfer:
                DoTransfer(bank);
                break;

                case MenuOption.NewAccount:
                DoNewAccount(bank);
                break;

                case MenuOption.TransactionHistory:
                DoPrintTransactionHistory(bank);
                break;

                case MenuOption.Quit:
                Console.WriteLine("QUIT!");
                break;
            }
       }
```

```
54          while( userSelection != MenuOption.Quit);
55
56      }
57
58      public static MenuOption ReadUserOption()
59      {
60          int usersOption = 0;
61
62          do
63          {
64              Console.WriteLine("*****Menu***** \n--------------------\n1.
                ↪  ---Withdraw---\n2. ---Deposit---\n3. ---Print---\n4.
                ↪  ---Transfer---\n5. ---Create a New Account---\n6. ---Print the
                ↪  Transaction History---\n7. ---Quit---\n--------------------");
65              Console.Write("What would you like to do today: ");
66              string userInput = Console.ReadLine();
67              try{
68                  usersOption = Convert.ToInt32(userInput);
69                  if (usersOption < 1 || usersOption > 7)
70                  {
71                      Console.WriteLine("Invalid Input, please try again!\n");
72                  }
73              }
74              catch(Exception e)
75              {
76                  Console.WriteLine( $ "{e.Message}\n");
77                  usersOption = -1;
78              }
79          }
80          while(usersOption < 1 || usersOption > 7);
81          return (MenuOption)(usersOption - 1);
82      }
83
84
85      public static void DoWithdraw(Bank toBank)
86      {
87          Account toAccount = FindAccount(toBank);
88          if ( toAccount == null) return;
89          decimal withdrawAmount = 0;
90          Console.Write("How much would you like to WITHDRAW? Please enter: $");
91          string userWithdraw = Console.ReadLine();
92          try
93          {
94              withdrawAmount = Convert.ToDecimal(userWithdraw);
95              WithdrawTransaction withdrawTransaction = new
                ↪  WithdrawTransaction(toAccount, withdrawAmount);
96              toBank.ExecuteTransaction(withdrawTransaction);
97              withdrawTransaction.Print();
98          }
99          catch(Exception e)
100         {
101             Console.WriteLine( $ "{e.Message}\n");
102         }
```

```
103     }
104
105     public static void DoDeposit(Bank toBank)
106     {
107         Account toAccount = FindAccount(toBank);
108         if ( toAccount == null ) return;
109         decimal depositAmount = 0;
110         Console.Write("How much would you like to DEPOSIT? Please enter: $");
111         string userDeposit = Console.ReadLine();
112         try
113         {
114             depositAmount = Convert.ToDecimal(userDeposit);
115             DepositTransaction depositTransaction = new
                ↪  DepositTransaction(toAccount, depositAmount);
116             toBank.ExecuteTransaction(depositTransaction);
117             depositTransaction.Print();
118         }
119         catch(Exception e)
120         {
121             Console.WriteLine( $ "{e.Message}\n");
122         }
123     }
124
125     public static void DoPrint(Bank bank)
126     {
127         Account account = FindAccount( bank );
128         if( account == null ) return;
129         account.Print();
130     }
131
132     public static void DoTransfer(Bank bank)
133     {
134         Account fromAccount = FindAccount (bank);
135         Account toAccount = FindAccount (bank);
136         if (fromAccount == null || toAccount == null) return;
137         decimal transferAmount = 0;
138         Console.Write( $ "How much would you like to TRANSFER from
            ↪  {fromAccount.Name}'s account to {toAccount.Name}'s account? Please
            ↪  enter: $");
139         string userTransfer = Console.ReadLine();
140         try
141         {
142             transferAmount = Convert.ToDecimal(userTransfer);
143             TransferTransaction transferTransaction = new
                ↪  TransferTransaction(fromAccount, toAccount, transferAmount);
144             bank.ExecuteTransaction(transferTransaction);
145             transferTransaction.Print();
146         }
147         catch(Exception e)
148         {
149             Console.WriteLine( $ "{e.Message}\n");
150         }
151     }
```

```
152
153     public static void DoNewAccount( Bank bank )
154     {
155         decimal startingBalance = 0;
156         Console.Write("Enter the name for the new account: ");
157         string name = Console.ReadLine();
158         Console.Write("Enter the starting balance for the new account: ");
159         try
160         {
161             startingBalance = decimal.Parse(Console.ReadLine());
162         }
163         catch
164         {
165             Console.WriteLine("Please enter a valid decimal number");
166         }
167
168         Account account = new Account(name, startingBalance);
169         bank.AddAccount( account );
170         Console.WriteLine("New account created.");
171     }
172
173     public static void DoPrintTransactionHistory( Bank bank )
174     {
175         bank.PrintTransactionHistory();
176     }
177
178     private static Account FindAccount( Bank fromBank )
179     {
180         Console.Write("Enter account name: ");
181         string name = Console.ReadLine();
182         Account result = fromBank.GetAccount(name);
183
184         if ( result == null )
185         {
186             Console.WriteLine( $"No account found with the name {name}");
187         }
188
189         return result;
190     }
191 }
```

```cs
using System;
using SplashKitSDK;
using System.Collections.Generic;

public class Bank
{
    private List<Account> _accounts = new List<Account>();
    private List<Transaction> _transactions = new List<Transaction>();

    public void AddAccount( Account account )
    {
        _accounts.Add(account);
    }
    public Account GetAccount( string name )
    {
        for( int i = 0 ; i < _accounts.Count; i++ )
        {
            if(name == _accounts[i].Name)
            {
                return _accounts[i];
            }
        }
        return null;
    }

    public void ExecuteTransaction( Transaction transaction )
    {
        _transactions.Add(transaction);
        transaction.Execute();
    }
    public void PrintTransactionHistory()
    {
        foreach ( Transaction transactionHistory in _transactions)
        {
            transactionHistory.Print();
        }
    }
}
```

```cs
using System;


public class Account
{
    private decimal _balance;
    private string _name;

    //Constructor
    public Account(string name, decimal startingBalance)
    {
        _name = name;
        _balance = startingBalance;

    }

    //Deposit method
    public bool Deposit(decimal amountToDeposit)
    {
        if (amountToDeposit > 0)
        {
            _balance += amountToDeposit;
            return true;
        }
        else
        {
            Console.WriteLine("**WARNING**\nThe deposit amount must be a positive number!");
            return false;
        }
    }

    //Withdraw mathod
    public bool Withdraw(decimal amountToWithdraw)
    {
        if(amountToWithdraw > 0 && amountToWithdraw <= _balance )
        {
            _balance -= amountToWithdraw;
            return true;
        }
        else if (amountToWithdraw <= 0)
        {
            Console.WriteLine("**WARNING**\nThe withdraw amount must be a positive number!");
            return false;
        }
        else
        {
            Console.WriteLine("**WARNING**\nThe withdraw amount must not exceed your exiting funds!");
            return false;
        }

```

```
51      }
52
53      //Read name property
54      public string Name
55      {
56          get { return _name; }
57      }
58
59      //Print method
60      public void Print()
61      {
62          Console.WriteLine( $"Account Name: {Name}\nBalance: {_balance}\n");
63      }
64  }
```

```
using System;
using SplashKitSDK;
public abstract class Transaction
{
    protected decimal _amount;
    private bool _executed = false;
    private bool _reversed = false;
    private DateTime _dateStamp;
    public abstract bool Success
    {
        get;
    }
    public bool Executed
    {
        get
        {
            return _executed;
        }
    }
    public bool Reversed
    {
        get
        {
            return _reversed;
        }
    }
    public DateTime DateStamp
    {
        get
        {
            return _dateStamp;
        }
    }
    public Transaction( decimal amount )
    {
        _amount = amount;
    }
    public abstract void Print();
    public virtual void Execute()
    {
         if( _executed )
        {
            throw new Exception("Cannot execute this transaction as it has already
            ↪  been exected.");
        }

        _executed = true;
        _dateStamp = DateTime.Now;
    }
    public virtual void Rollback()
    {
        if( !_executed )
        {
```

```
53              throw new Exception("Cannot rollback this transaction as it has not
                ↪  been executed.");
54          }
55          if( _reversed )
56          {
57              throw new Exception("Cannot rollback this transaction as it has already
                ↪  been reversed.");
58          }
59          _reversed = true;
60          _dateStamp = DateTime.Now;
61      }
62  }
```

```
using System;
using SplashKitSDK;

public class WithdrawTransaction : Transaction
{
    private Account _account;
    private bool _success = false;
    public override bool Success
    {
        get
        {
            return _success;
        }
    }

    public WithdrawTransaction(Account account, decimal amount) : base(amount)
    {
        _account = account;
    }

    public override void Execute()
    {
        base.Execute();
        _success = _account.Withdraw(_amount);
    }

    public override void Rollback()
    {
        base.Rollback();
        _account.Deposit(_amount);
    }

    public override void Print()
    {
        Console.Write (DateStamp);
        if( Success )
        {
            Console.Write($"*${_amount} was SUCCESSFULLY withdrawed from
            ↪ {_account.Name}'s account.");
        }
        else
        {
            Console.Write("*This Withdraw transaction was FAILED!");
        }

        if( Reversed )
        {
            Console.WriteLine("|The transaction was rolled back.");
        }
        else
        {
            Console.WriteLine("|The transaction was not rolled back.");
        }
```

```
53
54      }
55
56
57  }
```

```
using System;
using SplashKitSDK;

public class DepositTransaction : Transaction
{
    private Account _account;
    private bool _success = false;

    public override bool Success
    {
        get
        {
            return _success;
        }
    }

    public DepositTransaction(Account account, decimal amount) : base(amount)
    {
        _account = account;
    }

    public override void Execute()
    {
        base.Execute();
        _success = _account.Deposit(_amount);
    }

    public override void Rollback()
    {
        base.Rollback();
        _account.Withdraw(_amount);
    }

    public override void Print()
    {
        Console.Write (DateStamp);
        if( Success )
        {
            Console.Write($"*${_amount} was SUCCESSFULLY deposited into
            ↪  {_account.Name}'s account.");
        }
        else
        {
            Console.Write("*This deposit transaction was FAILED!");
        }

        if( Reversed )
        {
            Console.WriteLine("|The transaction was rolled back.");
        }
        else
        {
            Console.WriteLine("|The transaction was not rolled back.");
```

```
53              }
54
55          }
56
57
58  }
```

```
using System;

public class TransferTransaction : Transaction
{
    private Account _toAccount;
    private Account _fromAccount;

    private DepositTransaction _theDeposit;
    private WithdrawTransaction _theWithdraw;
    public override bool Success
    {
        get
        {
            return (_theDeposit.Success && _theWithdraw.Success);
        }
    }
    public TransferTransaction(Account fromAccount, Account toAccount, decimal
    ↪ amount) : base(amount)
    {
        _fromAccount = fromAccount;
        _toAccount = toAccount;
        _theWithdraw = new WithdrawTransaction(fromAccount, amount);
        _theDeposit = new DepositTransaction(toAccount, amount);
    }

    public override void Execute()
    {
        base.Execute();
        _theWithdraw.Execute();
        if( _theWithdraw.Success )
        {
            _theDeposit.Execute();

            if( !_theDeposit.Success && _theDeposit.Executed)
            {
                _theWithdraw.Rollback();
            }
        }
    }

    public override void Rollback()
    {
        base.Rollback();
        if(_theWithdraw.Success)
        {
            _theWithdraw.Rollback();
        }
        if(_theDeposit.Success)
        {
            _theDeposit.Rollback();
        }
    }

```

```
53      public override void Print()
54      {
55          Console.Write (DateStamp);
56          if( Success )
57          {
58              Console.WriteLine( $"*${_amount} was transferred from
                ↪  {_fromAccount.Name}'s account to {_toAccount.Name}'s account.");
59              Console.Write( $"      ");
60              _theWithdraw.Print();
61              Console.Write( $"      ");
62              _theDeposit.Print();
63          }
64          else
65          {
66              Console.WriteLine("*This transaction was FAILED!");
67              Console.Write( $"      ");
68              _theWithdraw.Print();
69              Console.Write( $"      ");
70              _theDeposit.Print();
71          }
72
73          if( Reversed )
74          {
75              Console.WriteLine("|The transaction was rolled back.");
76          }
77
78
79      }
80
81  }
```


C:\Users\ryty7777\Code\BankProgram_7.1
ryty7@DESKTOP-GLE7KAA MINGW64 /C/Users/ryty7777/Code/BankProgram_7.1
It is dated: Tue
-----Menu-----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---Create a New Account---
6. ---Print the Transaction History---
7. ---Quit---
What would you like to do today: 5
Enter the name for the new account: Ryang
Enter the starting balance for the new account: 10000
new account created.
-----Menu-----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---Create a New Account---
6. ---Print the Transaction History---
7. ---Quit---
What would you like to do today: 5
Enter the name for the new account: Tom
Enter the starting balance for the new account: 3000
new account created.
-----Menu-----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---Create a New Account---
6. ---Print the Transaction History---
7. ---Quit---
What would you like to do today: 2
Enter account name: ryang
How much would you like to DEPOSIT? Please enter: $2000
18/08/2020 7:19:48 PM $2000 was SUCCESSFULLY deposited into ryang's account.|The transaction was not rolled back.
-----Menu-----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---Create a New Account---
6. ---Print the Transaction History---
7. ---Quit---
What would you like to do today: 1
Enter account name: Tom
How much would you like to WITHDRAW? Please enter: $1000
18/08/2020 7:20:21 PM $1000 was SUCCESSFULLY withdrawed from Tom's account.|The transaction was not rolled back.
-----Menu-----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---Create a New Account---
6. ---Print the Transaction History---
7. ---Quit---
What would you like to do today: 4
Enter account name: ryang
Enter account name: Tom
How much would you like to TRANSFER from ryang's account to Tom's account? Please enter: $600
18/08/2020 7:21:44 PM $600 was transferred from ryang's Account to Tom's account.
18/08/2020 7:21:44 PM $600 was SUCCESSFULLY withdrawed from ryang's account.|The transaction was not rolled back.
18/08/2020 7:21:44 PM $600 was SUCCESSFULLY deposited into Tom's account.|The transaction was not rolled back.
-----Menu-----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---Create a New Account---
6. ---Print the Transaction History---
7. ---Quit---
What would you like to do today: 3
Enter account name: ryang
Account Name: Ryang
Balance: $500
-----Menu-----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---Create a New Account---
6. ---Print the Transaction History---
7. ---Quit---
What would you like to do today: 6
18/08/2020 7:19:48 PM $2000 was SUCCESSFULLY deposited into ryang's account.|The transaction was not rolled back.
18/08/2020 7:20:21 PM $1000 was SUCCESSFULLY withdrawed from Tom's account.|The transaction was not rolled back.
18/08/2020 7:21:44 PM $600 was transferred from ryang's Account to Tom's account.
18/08/2020 7:21:44 PM $600 was SUCCESSFULLY withdrawed from ryang's account.|The transaction was not rolled back.
18/08/2020 7:21:44 PM $600 was SUCCESSFULLY deposited into Tom's account.|The transaction was not rolled back.
-----Menu-----
1. ---Withdraw---
2. ---Deposit---
3. ---Print---
4. ---Transfer---
5. ---Create a New Account---
6. ---Print the Transaction History---
7. ---Quit---
What would you like to do today: 7
ryty7@DESKTOP-GLE7KAA MINGW64 /C/Users/ryty7777/Code/BankProgram_7.1

24 Different Robots

Dadd different kinds of robots

Outcome	Weight
Build Programs	♦♦♦◊◊

This task demonstrates the process of design and building a program and is backed up with relevant evidence.

Outcome	Weight
Design	♦♦◊◊◊

This task demonstrates the process of design and building a program and is backed up with relevant evidence.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the process of design and building a program and is backed up with relevant evidence.

Date	Author	Comment
2020/09/14 00:14	Yiyang Hou	Can I have more time to work on these tasks please
2020/09/20 02:43	Yiyang Hou	Ready to Mark
2020/09/20 02:43	Yiyang Hou	In the submission page, it suggests that we need to upload the program file, however, it is not modified in this task and the task sheet asks to submit the Robot-Dodge code instead, so I only enclosed the Robot-Dodge file in the submission, hopefully that is acceptable.
2020/09/25 17:57	Chetan Arora	### Discussion ModeWe are going to use the Discussion mode of OnTrack, you will be able to listen to my question(s) once and need to reply then. Please press the play button when you're free and available to answer the question(s). There are no retakes, so please make sure you reply to the questions.
2020/09/25 17:58	Chetan Arora	discussion comment
2020/09/25 17:58	Chetan Arora	Discuss
2020/09/27 20:43	Yiyang Hou	I'm so sorry about been so hesitant during the recording, I thought we were going to discuss about task 7.1 so I had the wrong file opened before I started recording, I had to rush to find the robot dodge code during the recording. I hope my answers addressed the points and are still acceptable.
2020/10/07 15:47	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Different Robots

Submitted By:
Yiyang Hou
houyiy
2020/09/20 02:43

Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦♦◊◊
Design	♦♦◊◊◊
Justify	♦◊◊◊◊

This task demonstrates the process of design and building a program and is backed up with relevant evidence.

September 20, 2020

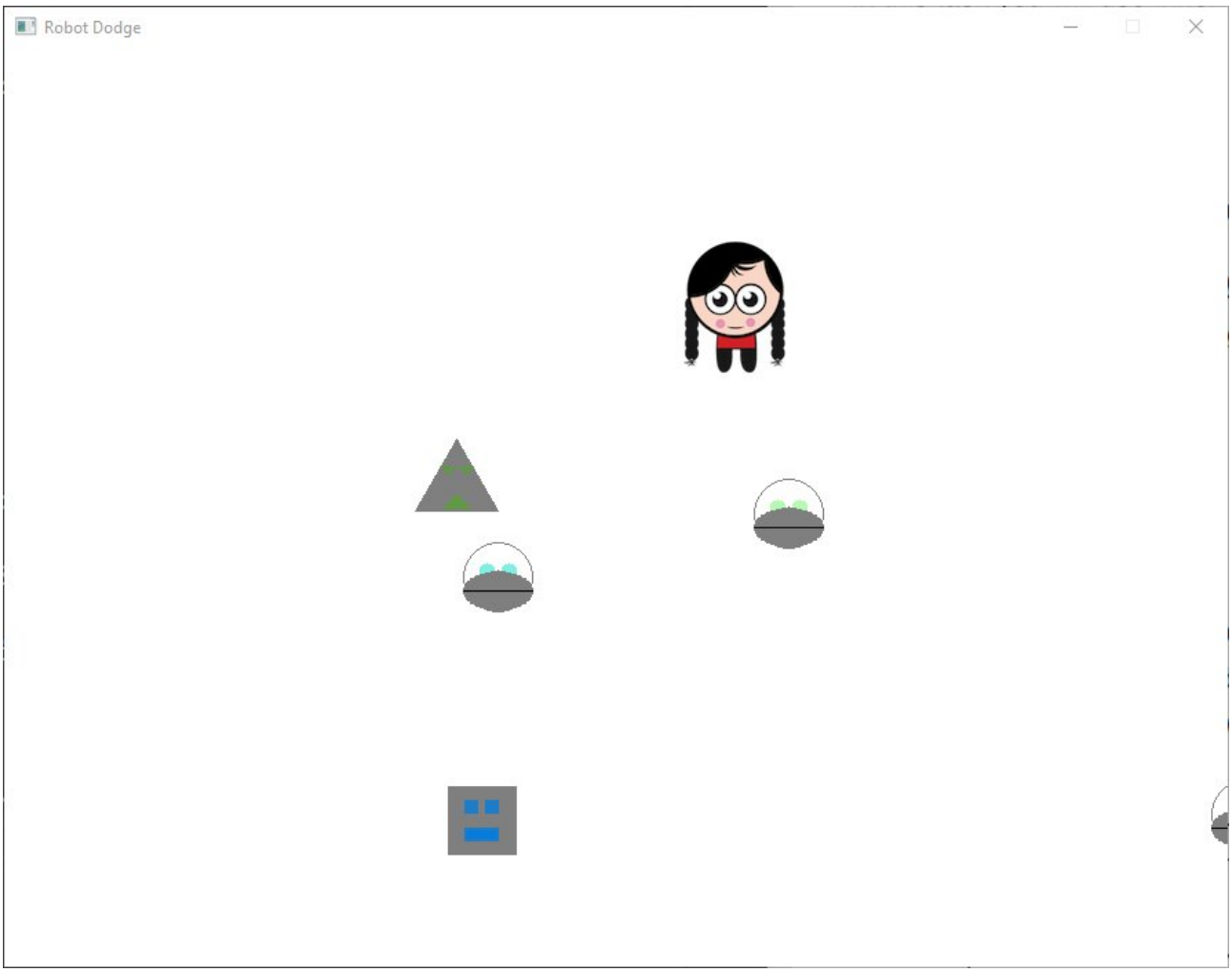
```
using SplashKitSDK;
using System.Collections.Generic;

public class RobotDodge
{
    private Player _Player;
    private Window _GameWindow;
    private List<Robot> _Robots = new List<Robot>();
    public bool Quit
    {
        get
        {
            return _Player.Quit;
        }
    }

    public RobotDodge( Window gameWindow )
    {
        _GameWindow = gameWindow;
        _Player = new Player( gameWindow );
    }

    public void HandleInput()
    {
        _Player.HandleInput();
        _Player.StayOnWindow(_GameWindow);
    }

    public void Draw()
    {
        _GameWindow.Clear(Color.White);
        foreach(Robot eachRobot in _Robots)
        {
            eachRobot.Draw();
        }
        _Player.Draw();
        _GameWindow.Refresh(60);
    }

    public void Update()
    {
        foreach(Robot eachRobot in _Robots)
        {
            eachRobot.Update();
        }

        if (SplashKit.Rnd()<0.01)
        {
            Robot newRobot = RandomRobot();
            _Robots.Add(newRobot);
        }

        CheckCollisions();
```

```
54      }
55      public Robot RandomRobot()
56      {
57          if(SplashKit.Rnd()<=0.3)
58          {
59              return new Boxy(_GameWindow,_Player);
60          }
61          else if(SplashKit.Rnd()>0.3 && SplashKit.Rnd() <= 0.6)
62          {
63              return new Angular(_GameWindow,_Player);
64          }
65          else
66          {
67              return new Roundy(_GameWindow,_Player);
68          }
69
70      }
71
72      private void CheckCollisions()
73      {
74          List<Robot> removeRobots = new List<Robot>();
75          foreach(Robot eachRobot in _Robots)
76          {
77              if (_Player.CollidedWith(eachRobot) ||
                ↪  eachRobot.IsOffscreen(_GameWindow))
78              {
79                  removeRobots.Add(eachRobot);
80              }
81          }
82          foreach(Robot eachRemoveRobot in removeRobots)
83          {
84              _Robots.Remove(eachRemoveRobot);
85          }
86      }
87  }
```

```
using SplashKitSDK;

public abstract class Robot
{
    public double X { get; set; }
    public double Y { get; set; }
    private Vector2D Velocity { get; set; }
    public Color MainColor { get; set; }
    public int Width
    {
        get{ return 50; }
    }
    public int Height
    {
        get{ return 50; }
    }
    public Circle CollisionCircle
    {
        get{ return SplashKit.CircleAt( X + 25, Y + 25, 35.36); }
    }

    public Robot( Window gameWindow, Player player )
    {
        MainColor = Color.RandomRGB(200);
        if (SplashKit.Rnd() < 0.5 )
        {
            X = SplashKit.Rnd(gameWindow.Width);

            if (SplashKit.Rnd() < 0.5) Y = -Height;
            else Y = gameWindow.Height;
        }
        else
        {
            Y = SplashKit.Rnd(gameWindow.Height);

            if (SplashKit.Rnd() < 0.5 ) X = -Width;
            else X = gameWindow.Width;
        }

        const int SPEED = 4;

        Point2D fromPt = new Point2D()
        {
            X = X, Y = Y
        };

        Point2D toPt = new Point2D()
        {
            X= player.X, Y = player.Y
        };

        Vector2D dir;
        dir = SplashKit.UnitVector(SplashKit.VectorPointToPoint( fromPt, toPt ));
```

```
54
55          Velocity = SplashKit.VectorMultiply( dir, SPEED );
56
57      }
58      public abstract void Draw();
59
60      public void Update()
61      {
62          X += Velocity.X;
63          Y += Velocity.Y;
64      }
65
66      public bool IsOffscreen( Window screen )
67      {
68          while( X < -Width || X > screen.Width || Y < -Height || Y > screen.Height )
69          {
70              return true;
71          }
72          return false;
73
74      }
75  }
76
77  public class Boxy : Robot
78  {
79      public Boxy( Window gameWindow, Player player) : base(gameWindow, player)
80      {
81      }
82      public override void Draw()
83      {
84          double leftX, rightX;
85          double eyeY, mouthY;
86
87          leftX = X + 12;
88          rightX = X + 27;
89          eyeY = Y + 10;
90          mouthY = Y + 30;
91          SplashKit.FillRectangle( Color.Gray, X, Y, Width, Height);
92          SplashKit.FillRectangle( MainColor, leftX, eyeY, 10, 10);
93          SplashKit.FillRectangle( MainColor, rightX, eyeY, 10, 10);
94          SplashKit.FillRectangle( MainColor, leftX, mouthY, 25, 10);
95          SplashKit.FillRectangle( MainColor, leftX + 2, mouthY + 2, 21, 6);
96      }
97
98  }
99
100 public class Roundy : Robot
101 {
102     public Roundy( Window gameWindow, Player player) : base (gameWindow, player)
103     {
104     }
105
106     public override void Draw()
```

```
107     {
108         double leftX, midX, rightX;
109         double midY, eyeY, mouthY;
110 
111         leftX = X + 17;
112         midX = X + 25;
113         rightX = X + 33;
114 
115         midY = Y + 25;
116         eyeY = Y + 20;
117         mouthY = Y + 35;
118 
119         SplashKit.FillCircle(Color.White, midX, midY, 25);
120         SplashKit.DrawCircle(Color.Gray, midX, midY, 25);
121         SplashKit.FillCircle(MainColor, leftX, eyeY, 5);
122         SplashKit.FillCircle(MainColor, rightX, eyeY, 5);
123         SplashKit.FillEllipse(Color.Gray, X, eyeY, 50, 30);
124         SplashKit.DrawLine(Color.Black, X, mouthY, X + 50, Y + 35);
125     }
126 }
127 
128 public class Angular : Robot
129 {
130     public Angular( Window gameWindow, Player player) : base (gameWindow, player)
131     {
132     }
133     public override void Draw()
134     {
135         double leftX, midX, rightX;
136         double eyeY, mouthY;
137 
138         leftX = X;
139         midX = X + 30;
140         rightX = X + 60;
141         eyeY = Y + 20;
142         mouthY = Y + 50;
143         SplashKit.FillTriangle(Color.Gray, X, Y+52, midX, Y, rightX, Y+52);
144         SplashKit.FillTriangle(MainColor, leftX + 18.84, eyeY, leftX + 23.84,
            ↪  eyeY+5, leftX + 28.84, eyeY);
145         SplashKit.FillTriangle(MainColor, leftX + 31.52, eyeY, leftX + 36.52,
            ↪  eyeY+5, leftX + 41.52, eyeY);
146         SplashKit.FillTriangle(MainColor, leftX + 20, mouthY, leftX + 30,
            ↪  mouthY-10, leftX + 40, mouthY);
147     }
148 }
```


Robot Dodge

25 Concept Visualisation 2

Visualise the programming concepts we've covered so far

Outcome	Weight
Principles	♦♦♦◊◊

This task demonstrates the understanding of the principles of Object-Oriented Programming. Relevant evidence is provided.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates the understanding of the principles of Object-Oriented Programming. Relevant evidence is provided.

Date	Author	Comment
2020/09/21 22:49	Yiyang Hou	Ready to Mark
2020/09/22 21:01	Chetan Arora	Complete
2020/09/22 21:01	Chetan Arora	_ Lambdas and Delegates are not intuitive

Deakin University

Object Oriented Development

OnTrack Submission

Concept Visualisation 2

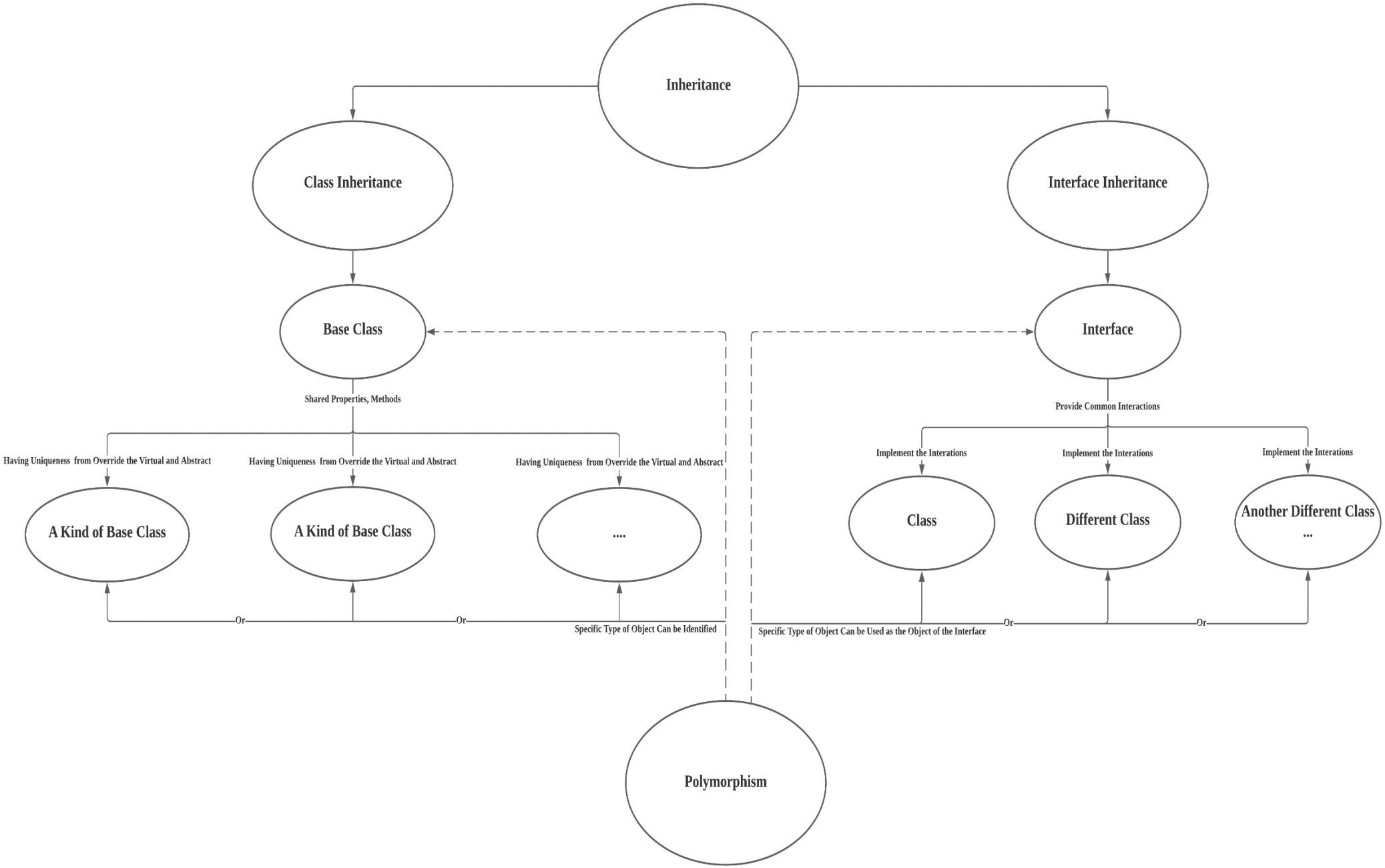
Submitted By:
Yiyang Hou
houyiy
2020/09/21 22:49

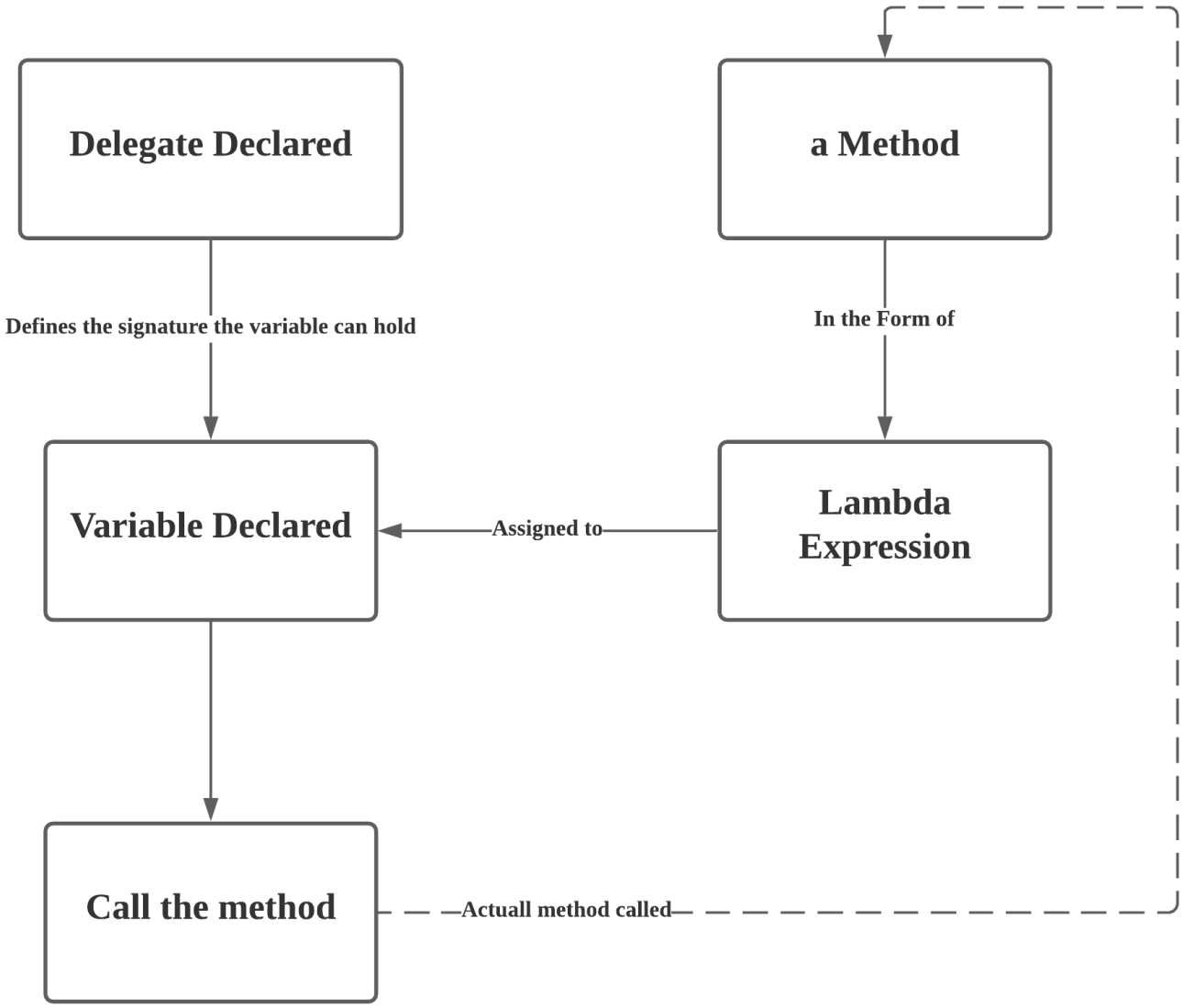
Tutor:
Chetan Arora

Outcome	**Weight**
Principles	♦♦♦◊◊
Justify	♦◊◊◊◊

This task demonstrates the understanding of the principles of Object-Oriented Programming. Relevant evidence is provided.

September 21, 2020


Inheritance
Class Inheritance
Interface Inheritance
Base Class
Interface
Shared Properties, Methods
Provide Common Interactions
Having Uniqueness from Override the Virtual and Abstract
Having Uniqueness from Override the Virtual and Abstract
Having Uniqueness from Override the Virtual and Abstract
Implement the Interations
Implement the Interations
Implement the Interations
A Kind of Base Class
A Kind of Base Class
....
Class
Different Class
Another Different Class
...
Or
Or
Specific Type of Object Can be Identified
Specific Type of Object Can be Used as the Object of the Interface
Or
Or
Polymorphism


Delegate Declared
a Method
Defines the signature the variable can hold
In the Form of
Variable Declared
Assigned to
Lambda Expression
Call the method
Actuall method called

In the first visualisation diagram, it illustrates the concept of class inheritance and interface inheritance. Under the class inheritance, it shows that a number of subclasses of related types can inherit shared properties and methods from a base class to reduce code duplication. While virtual, abstract and override techniques can be used to maintain uniqueness of each subclass. The concept of polymorphism allows the uniqueness can be identified according to the specific type of the subclass when required. On the other side, interface inheritance can be visualised. Interface provides common interactions for different types. The interface can be implemented differently as per to the specific type. The idea of polymorphism is also illustrated here, as once the interface is implemented by a class, the object of that type can then be used as the object of the interface.

In the second visualisation diagram, it illustrates the concept of delegate and the use of Lambda expression with it. After the delegate is declared and the signature of stored method is defined, a variable of that type can then be declared. A method can be implemented in the in-line form which is a Lambda expression and assign it to the variable. The method can be called through the variable.

26 Another Language

Pick up another programming language

Outcome	Weight
Build Programs	♦♦♦◊◊

This task demonstrates how to build a program.

Outcome	Weight
Justify	♦◊◊◊◊

This task demonstrates how to build a program.

Date	Author	Comment
2020/09/27 00:52	Yiyang Hou	Ready to Mark
2020/09/27 00:52	Yiyang Hou	Task discussion:1.Learn about the similarity and the difference between the new language and the language that I know is the first thing I would do to approach a new language. On top of that is just to learn the syntax.2.I already know about the concept of class and objects, the structure of the program and the control flow.
2020/10/01 20:27	Chetan Arora	Complete

Deakin University

Object Oriented Development

OnTrack Submission

Another Language

Submitted By:
Yiyang Hou
houyiy
2020/09/27 00:52

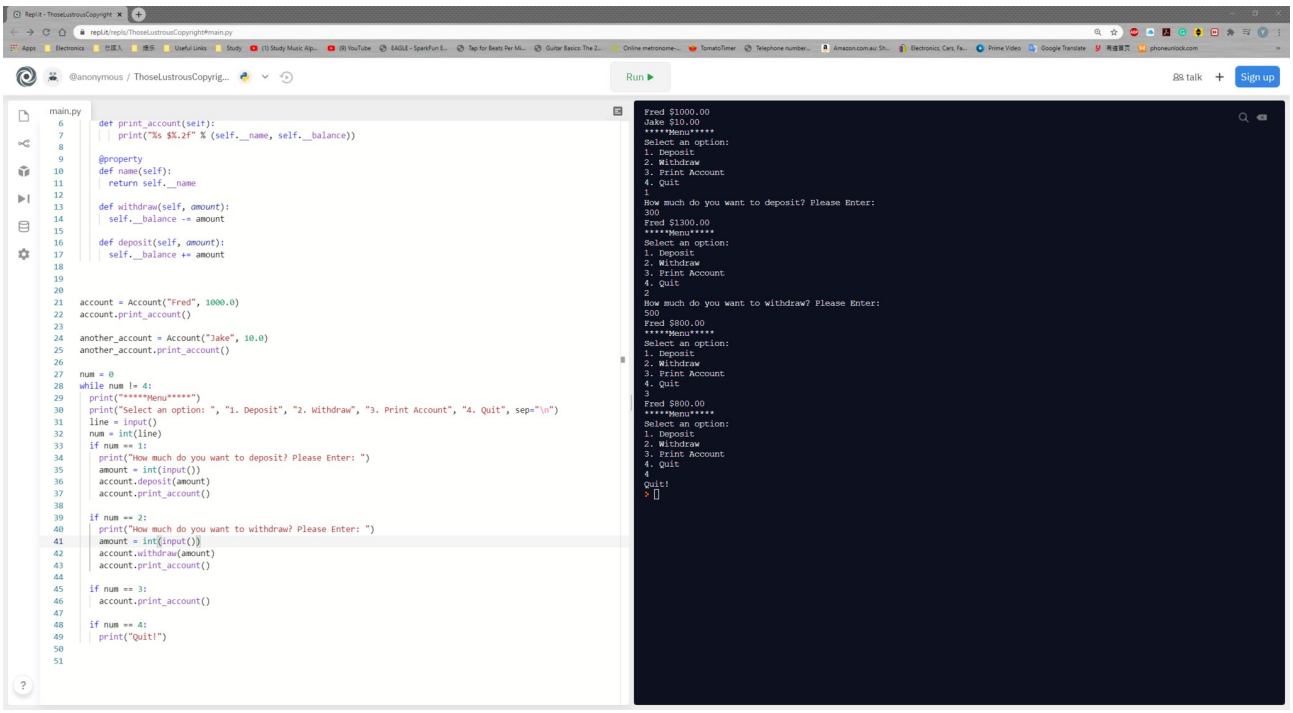
Tutor:
Chetan Arora

Outcome	**Weight**
Build Programs	♦♦♦◊◊
Justify	♦◊◊◊◊

This task demonstrates how to build a program.

September 27, 2020

```
class Account:
    def __init__(self, name, startingBalance):
        self.__name = name
        self.__balance = startingBalance

    def print_account(self):
        print("%s $%.2f" % (self.__name, self.__balance))

    @property
    def name(self):
      return self.__name

    def withdraw(self, amount):
      self.__balance -= amount

    def deposit(self, amount):
      self.__balance += amount



account = Account("Fred", 1000.0)
account.print_account()

another_account = Account("Jake", 10.0)
another_account.print_account()

num = 0
while num != 4:
  print("*****Menu*****")
  print("Select an option: ", "1. Deposit", "2. Withdraw", "3. Print Account", "4.
   ↪  Quit", sep="\n")
  line = input()
  num = int(line)
  if num == 1:
    print("How much do you want to deposit? Please Enter: ")
    amount = int(input())
    account.deposit(amount)
    account.print_account()

  if num == 2:
    print("How much do you want to withdraw? Please Enter: ")
    amount = int(input())
    account.withdraw(amount)
    account.print_account()

  if num == 3:
    account.print_account()

  if num == 4:
    print("Quit!")
```


@anonymous / ThoseLustrousCopyrig...
Run
talk
Sign up
main.py
def print_account(self):
print("%s $%.2f" % (self.__name, self.__balance))
@property
def name(self):
return self.__name
def withdraw(self, amount):
self.__balance -= amount
def deposit(self, amount):
self.__balance += amount
account = Account("Fred", 1000.0)
account.print_account()
another_account = Account("Jake", 10.0)
another_account.print_account()
num = 0
while num != 4:
print("*****Menu*****")
print("Select an option: ", "1. Deposit", "2. Withdraw", "3. Print Account", "4. Quit", sep="\n")
line = input()
num = int(line)
if num == 1:
print("How much do you want to deposit? Please Enter: ")
amount = int(input())
account.deposit(amount)
account.print_account()
if num == 2:
print("How much do you want to withdraw? Please Enter: ")
amount = int(input())
account.withdraw(amount)
account.print_account()
if num == 3:
account.print_account()
if num == 4:
print("Quit!")
Fred $1000.00
Jake $10.00
*****Menu*****
Select an option:
1. Deposit
2. Withdraw
3. Print Account
4. Quit
1
How much do you want to deposit? Please Enter:
300
Fred $1300.00
*****Menu*****
Select an option:
1. Deposit
2. Withdraw
3. Print Account
4. Quit
2
How much do you want to withdraw? Please Enter:
500
Fred $800.00
*****Menu*****
Select an option:
1. Deposit
2. Withdraw
3. Print Account
4. Quit
3
Fred $800.00
*****Menu*****
Select an option:
1. Deposit
2. Withdraw
3. Print Account
4. Quit
4
Quit!
>

27 Draft Learning Summary Report

Draft learning summary report

Outcome	Weight
Evaluate Code	♦◊◊◊◊

This report summarises my achievement of the unit learning outcomes.

Outcome	Weight
Principles	♦◊◊◊◊

This report summarises my achievement of the unit learning outcomes.

Outcome	Weight
Build Programs	♦◊◊◊◊

This report summarises my achievement of the unit learning outcomes.

Outcome	Weight
Design	♦◊◊◊◊

This report summarises my achievement of the unit learning outcomes.

Outcome	Weight
Justify	♦♦♦♦♦

This report summarises my achievement of the unit learning outcomes.

Date	Author	Comment
2020/10/09 00:34	Yiyang Hou	Ready to Mark

Deakin University

Object Oriented Development

OnTrack Submission

Draft Learning Summary Report

Submitted By:
Yiyang Hou
houyiy
2020/10/09 00:34

Tutor:
Chetan Arora

Outcome	Weight
Evaluate Code	♦◇◇◇◇
Principles	♦◇◇◇◇
Build Programs	♦◇◇◇◇
Design	♦◇◇◇◇
Justify	♦♦♦♦♦

This report summarises my achievement of the unit learning outcomes.

October 9, 2020

SIT771

Object-oriented Development

Learning Summary Report

Yiyang Hou
220570074

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	✓
Pass tasks complete	✓

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	

Minimum Distinction Checklist (in addition to Credit Checklist)

	Included
Something Awesome 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: **Yiyang Hou**

Portfolio Overview

This portfolio includes work that demonstrates that I have achieve all Unit Learning Outcomes for SIT771 Object-oriented Development to a **Credit** level.

[After progressing through the unit, I have learnt the principals of object-oriented programming and their application, how to design and build my own program as well as how to evaluate a piece of given code.

In credit task 4.2C, I was able to completed the task by evaluating and refactoring the target code. That demonstrates that I have achieved the unit learning outcome of Evaluate Code.

The series of robot dodge programs gave me the opportunity to practice my coding skills and have demonstrated my achievement of the unit learning outcome of Build Programs. In addition to that, they also helped with my understanding of how to read, evaluate and communicate solution structures, it demonstrates my achievement of the unit learning outcome of Design.

The concept visualisation tasks helped me with consolidating my understanding of the key principals of object-oriented programming and have demonstrated my achievement of the unit learning outcome of Principals.

Finally, I have completed all the tasks by providing the relevant evidence such as the code, screenshots and diagrams, that demonstrates that I have achieved the unit learning outcome of Justify.

I was able to complete all the tasks up to the level of credit with relative ease, in conjunction with the forementioned justifications, I believe that I have demonstrated that I have reached a credit level.]

Reflection

The most important things I learnt:

[The most important thing that I have learnt must be the key principals of object-oriented programming. Those notions were difficult to understand at first, but became clearer with proper practice. Once I got grasp on those principals all the tasks become easier for me, even learning another language becomes easy.]

The things that helped me most were:

[The thing that helped me the most was the syntax guides. A lot of the times, I know how to approach the objective of a task, but stuck with getting the syntax wrong, the syntax guides have always been very clear and helpful.]

I found the following topics particularly challenging:

[The concept of object and class was very challenging for me, as this is the foundation to object-oriented programming, so I struggled with the first several tasks because of it. As I do some extra studies, I have finally understood the concept.]

I found the following topics particularly interesting:

[I found every topic interesting, because everything I have learnt in this unit is very useful and practical. I can visualise that there is a place for every topic covered in this unit in an actual program.]

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

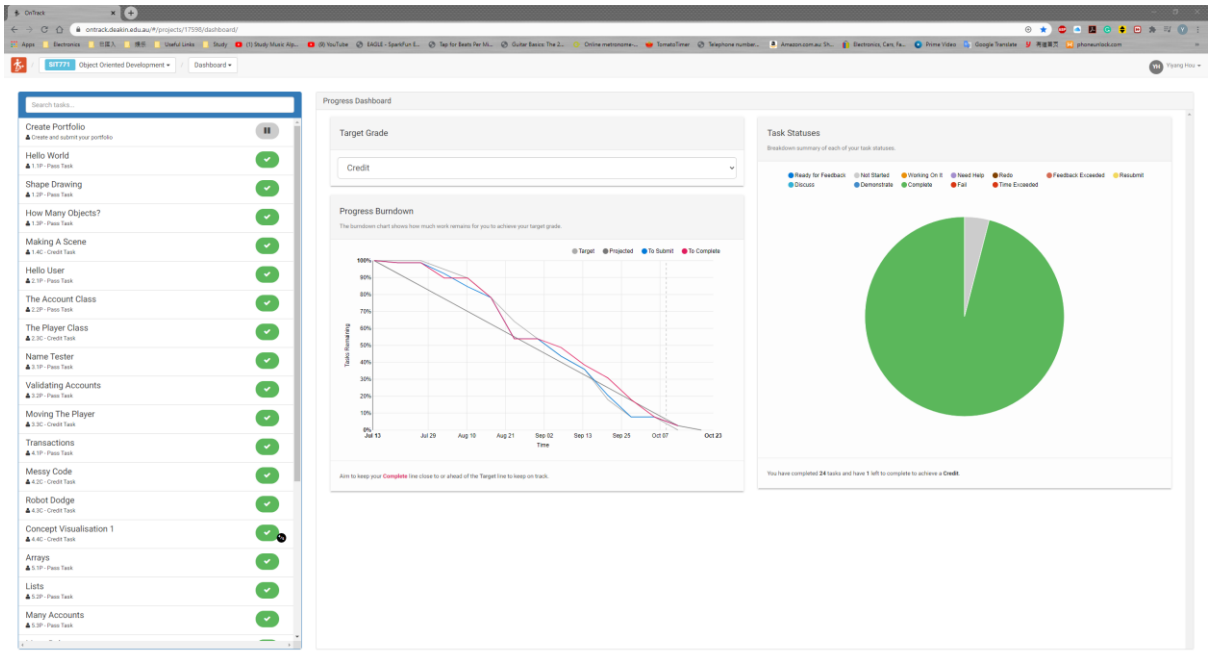
[I learnt the key principles of object-oriented programming really well, because every time I stuck in a task, I go back to these principles, they can always help me to solve the problems I had at the time.]

I still need to work on the following areas:

[I still need to work on my design skills. In my attempt of completing the distinction tasks, I found that designing my own solution is my weakness. I will need to work on that aspect going forward.]

My progress in this unit was …:

[According to the graph, I was a little behind the schedule at a point, but I quickly picked up the pace and managed to stay on the track.



This unit will help me in the future:

[I believe that everything I learnt from this unit will be valuable for me whether for my studies or my career in the industry. Because this unit introduces the world of programming to me and be able to program and understanding the related concepts are the building blocks for the software industry. So I will be applying the knowledge and skills that I have learnt from this unit in my future studies as well as my career.]

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

[I tried to do my best not to procrastinate and leave thing to the last moment, but because the online learning mode is very new to me, I could have done better in terms of progress management and eliminating distractions. With the experience of this trimester, I will confidently apply the learning skills that I have learnt this trimester to the future studies, whether online or on campus.]