

College of Engineering, Construction and Living Sciences

Bachelor of Information Technology

ID511: Programming 2

Level 5, Credits 15

Theory Examination

# Assessment Overview

In this **individual** assessment, you will answer **questions** covering various taught concepts in this course. The main purpose of this assessment is to simulate a series of real-world theoretical coding problems and demonstrate your ability to solve them.

# Learning Outcomes

At the successful completion of this course, learners will be able to:

1. Build interactive, event-driven GUI applications using pre-built components.
2. Declare and implement user-defined classes using encapsulation, inheritance and polymorphism.

**Conditions of Assessment**

You will complete this assessment during your learner-managed time. This assessment will need to be completed by

**Wednesday, 15 November 2023** at **12.10 PM**.

# Pass Criteria

This assessment is criterion-referenced (CRA) with a cumulative pass mark of **50%** over all assessments in **ID511: Programming 2**.

# Authenticity

This assessment is **open-book**. However, all parts of your submitted assessment document **must** be completely your work. **Do not** use AI generative applications such as **ChatGPT**. In addition, **do not** discuss your **theory examination** answers with **anyone** during the exam. Failure to do this will result in a mark of **zero** for this assessment.

# Policy on Submissions, Extensions, Resubmissions and Resits

The school’s process concerning submissions, extensions, resubmissions and resits complies with **Otago Polytechnic** policies. Learners can view policies on the **Otago Polytechnic** website located at [https://www.op.ac.nz/about-](https://www.op.ac.nz/about-us/governance-and-management/policies) [us/governance-and-management/policies](https://www.op.ac.nz/about-us/governance-and-management/policies).

# Submission

You **must** submit this assessment document via **Microsoft Outlook** no later than **12.10 PM**. The course lecturer’s email address is [**grayson.orr@op.ac.nz**.](mailto:grayson.orr@op.ac.nz.) Please ensure you include **ID511: Programming 2 Theory Examination** in the subject line of your email.

# Extensions

Extensions **are not** applicable for this assessment.

# Resubmissions

Resubmissions **are not** applicable for this assessment.

# Resits

Resits and reassessments **are not** applicable in **ID511: Programming 2**.

# Instructions

There is a total of **100 marks**. Please answer the **questions** below. **Note:** All your answers must be contained within this assessment document.

## Multiple-choice (6 marks)

**Please highlight your answer.**

1. Which of the following statements creates a new instance of the Person class? (1 mark)
   1. Person person = new Person();
   2. ~~Person person = Person.Create();~~
   3. ~~Person person;~~
   4. ~~Person person = Person.New();~~
2. In a parent class, which of the following keywords are used to mark a method to be overridden? (2 marks)
   1. Virtual
   2. Abstract
   3. ~~Override~~
   4. ~~All the above~~
3. You have been given a List of type Bike called bikes. Which of the following statements returns the last item in bikes? (1 mark)
   1. ~~bikes[bikes.Length - 1];~~
   2. bikes[bikes.Count - 1];
   3. ~~bikes[bikes.Size - 1];~~
   4. ~~bikes[bikes.count – 1];~~
4. You have been given a program that has five classes. ClassOne derives from ClassTwo, ClassTwo derives from ClassThree, ClassThree derives from ClassFour and ClassFour derives from ClassFive. Which of the following constructors will be created second if a new instance of ClassOne is created? (1 mark)
   1. ~~ClassTwo()~~
   2. ~~ClassThree()~~
   3. ClassFour()
   4. ~~ClassFive()~~
5. Which of the following statements toggle a Timer between true and false? (1 mark)
   1. ~~timer1.Enabled = true;~~
   2. ~~timer1.Enabled = false;~~
   3. ~~timer1.Enabled = timer1.Enabled;~~
   4. timer1.Enabled = !timer1.Enabled;

## Fill in the blank (5 marks)

1. An object’s fields define its \_state\_. Hint: A five-letter word. (1 mark)
2. An object’s methods define its \_behaviour\_. Hint: A nine-letter word. (1 mark)
3. A good method \_name\_ makes it clear what the method does. Hint: A four-letter word. (1 mark)
4. An object is an \_instance\_ of a class. Hint: An eight-letter word. (1 mark)
5. An \_enumeration\_ is a value type that is used to define a set of named constants. Hint: An eleven-letter word. (1 mark)

## Abstract data types (11 marks)

1. Explain the difference between a stack and queue. (3 marks)

A stacks use the last in, first out principal, where queues use the first in, first out principal.

1. What are two alternative names for a map? (2 marks)

Dictionary, associative array

1. Explain four operations that a set typically supports. (4 marks)

Insertion: Adding a new element into the set

Deletion: Removing an element from the set

Membership testing: Checking if an element exists in the set

Set union: Creating a new set that includes all the elements from 2 sets, but removes duplicates

Assume the following code:

A close-up of a computer code

Description automatically generated

1. Explain the method used to return the top item in myQueue without removing it. (1 mark)

myQueue.Peek() this will return the first item in the queue without removing it

1. Which property is used to return the size of myQueue? (1 mark)

myQueue.Count returns and int value of number of elements in myQueue

## Classes (22 marks)

1. Explain the difference between public, private and protected class members, i.e., fields and methods. Provide code examples to illustrate your points. (3 marks)

Public methods and variables are “visible” to all other classes, methods and variables. They only need to be referenced correctly. Please use the code below as a example. ClassTwo can access both ClassOne’s variables and methods because they are public and “visible” to all other classes.

public class ClassOne

{

public int num1 = 3;

public string MethodOne()

{

return "method one string";

}

}

public class ClassTwo

{

ClassOne c1 = new();

public void MethodTwo()

{

int num2 = c1.num1;

}

}

Private classes will not allow any variable or method to be public within it. This makes everything “non-visible” to other classes. Again, please use the code below as an example. ClassTwo would not be able to get anything from ClassOne because the members are “non-visible” to it.

private class ClassOne

{

private int num1 = 3;

private string MethodOne()

{

return "method one string";

}

}

public class ClassTwo

{

ClassOne c1 = new();

public void MethodTwo()

{

int num2 = c1.num1;

string words2 = c1.MethodOne();

}

}

Protected classes act like private ones, except the children on the class will act like the parent is public.

Please use code below as an example. ClassOne is a protected class, and is parent of ClassTwo. ClassTwo will be able to get the values because it is the child. ClassThree will not get any values because it is not a child.

protected class ClassOne

{

protected int num1 = 3;

protected string MethodOne()

{

return "method one string";

}

}

public class ClassTwo : ClassOne

{

public void MethodTwo()

{

int num2 = num1;

string words2 = MethodOne();

}

}

public class ClassThree

{

ClassOne c1 = new();

public void MethodThree()

{

int num3 = c1.num1();

string words = c1.MethodOne();

}

}

1. Describe the purpose of a class constructor and how they are different from regular methods. (2 marks)

Calling a constructor allows an object of the class to be instanced. Values that you want different per instance can be changed in the call. Normal methods cannot instance an object of a class.

A constructor is different from normal methods because it has the same name as the class, and some methods need an instance to run, but the constructor creates the instance.

1. You are tasked with creating a class to represent a basic bank account. Design a class called BankAccount with the following specifications:
   * + Three private fields – accountNumber of type string, accountHolderName of type string and balance of type of double. (3 marks)
     + A constructor that initialises accountNumber, accountHolderName and balance when a new instance of the BankAccount class is created. (4 marks)
     + A public method called Deposit that accepts a parameter called amount of type double. If the amount given is negative, i.e., -10, then display the following message – Invalid deposit amount. Else, add the amount parameter to the balance field. (5 marks)
     + A public method called Withdraw that takes a parameter called amount of type of double. If the amount given is negative or greater than the balance field, then display the following message – Invalid withdrawal amount. Else, subtract the amount parameter to the balance field. (5 marks)

(code is below)

## Encapsulation (1 mark)

1. Create a property for the balance field. (1 mark)

(code is below)

## Objects (6 marks)

1. Create a new instance of the BankAccount class with the following values – account number: 100-000-000, account holder name: Jane Doe and balance: 1000.50. (2 marks)
2. Using the new instance, deposit 500.00, then display the following – New balance is <balance property>. (2 marks)
3. Using the new instance, withdraw 150.50, then display the following – New balance is <balance property >. (2 marks)

(code is below)

public class Controller

{

private static BankAccount bankAccount = new("100 - 000 - 000", "Jane Doe", 1000.50);

static void Main(string[] args)

{

bankAccount.Deposit(500.00);

Console.WriteLine($"New balance is {bankAccount.Balance:c2}");

bankAccount.Withdraw(150.50);

Console.WriteLine($"New balance is {bankAccount.Balance:c2}");

}

}

public class BankAccount

{

//fields

private string accountNumber, accountHolderName;

private double balance;

//constructor

public BankAccount(string accountNumber, string accountHolderName, double balance)

{

this.accountNumber = accountNumber;

this.accountHolderName = accountHolderName;

this.balance = balance;

}

//encapsulation

public double Balance { get => balance; set => balance = value; }

//methods

public void Deposit(double amount)

{

if (amount < 0)

{

Console.WriteLine("Invalid deposit amount");

}

else

{

balance += amount;

}

}

public void Withdraw(double amount)

{

if (amount < 0 || amount > balance)

{

Console.WriteLine("Invalid withdrawal amount");

}

else

{

balance -= amount;

}

}

}

## Inheritance (15 marks)

## Explain two advantages of using inheritance. (2 marks)

* 1. You can code class members once in the parent, and not have to repeat that code in every single child.
  2. If you want to change a class member you don’t have to change it for each class manually, just change it in the parent.

## Explain the difference between an abstract and virtual method. (3 marks)

Virtual classes can have a default method if the child doesn’t override it, abstract classes cannot have a default and MUST be overridden.

## Given the following two classes, Animal and Bird, where Bird is derived from Animal. Describe what inheritance means in this context and provide an example of a field and method that the Bird class might inherit from the Animal class. (3 marks)

In real life all birds are animals. This reflects in the classes where Bird is a subcategory, or child, of Animal.   
An example of a field could be: int name, and method could be Eat.

Assume the following code:

A screenshot of a computer program

Description automatically generated

1. Explain the base keyword. (1 mark)

Base is required in the constructor of a child class. It tells the constructor what fields to use from the parent.

1. Complete the following diagram to show the relationship between the Boat, Sailboat and Motorboat classes. (3 marks)

Boat

Motorboat

Sailboat

1. Assume a new instance of the Sailboat class is created, list all the methods that the new instance can use. (3 marks)
   1. Sail()
   2. Boat constructor
   3. Motorboat constructor

## Polymorphism (5 marks)

1. Explain what method overloading is. Provide a code example to illustrate your point. (3 marks)

Overloading is when multiple methods are made that have the same name but take different parameters

Eg, Multiply(int num1, int num2) Multiply(double num1, double num2) Multiply(int num1, double num2)

1. What are two types of polymorphism? (2 marks)
   1. Overloading
   2. Overriding

## LINQ (4 marks)

1. Given the following List of type string, write a LINQ query that returns all iconic brands starting with the letter "A" or "a". (2 marks)

A screenshot of a computer program

Description automatically generated

var brandQuery = from brand in iconicBrands

where brand[0] == 'a' || brand[0] == 'A'

select brand;

1. Given the following class, write a LINQ query that returns the salary of all employees in the “Engineering” department. (2 marks)

A screenshot of a computer code

Description automatically generated

var employeeQuery = from employee in Employees

where employee.department == "Engineering"

select employee.salary;

## Unit testing (6 marks)

You have been provided a file called Calc.cs. Create a TestMethod for the following methods:

1. public double Add(double firstNum, double secondNum). (2 marks)

[TestMethod]

public void AddReturnsSum()

{

Calc calc = new();

double num1 = 3.5, num2 = 7.61, expected = 11.11;

double actual = calc.Add(num1, num2);

Assert.AreEqual(expected, actual);

}

1. public double Divide(double firstNum, double secondNum).
   1. Test the result of firstNum divided by secondNum. (2 marks)

[TestMethod]

public void DivideReturnsDivided()

{

Calc calc = new();

double num1 = 24.5, num2 = 3.5, expected = 7.0;

double actual = calc.Divide(num1, num2);

Assert.AreEqual(expected , actual);

}

* 1. Test the result of firstNum divided by zero. (2 marks)

[TestMethod]

public void DivideByZeroReturnZero()

{

Calc calc = new();

double num1 = 3.0, num2 = 0.0, expected = 0.0;

double actual = calc.Divide(num1, num2);

Assert.AreEqual(expected, actual);

}

## Debugging (6 marks)

1. Explain two techniques that can be used to debug a program. (2 marks)
   1. Using the console. You can use the console to show variables and changes while the code is running.
   2. Using the Visual Studio built-in debuggeing tool. You can watch all values, set breakpoints and step through the code one line at a time.
2. Given the following code, identify two errors and explain how to fix them. (4 marks)

A screenshot of a computer program

Description automatically generated

The for loop is missing curly braces. The braces need to go around sum += i.   
for (int i = 1; i <= n; i++)

{

sum += i;

}

The while loop is also missing its curly braces. They need to go around the contents of the while.

while (n > 0)

{

Console.WriteLine($"Countdown: {n}");

n--;

}

## System design (6 marks)

1. Assume the screenshot below. This program generates and displays six unique numbers between 1 and 50 when the Choose numbers button is clicked. Explain how you would write this program. Note: Do not provide any code. (6 marks)

A screenshot of a computer game

Description automatically generated with medium confidence

I would make an empty array of 6 length. When the button is pressed, make a foreach loop of the array. Fill the first slot with a random number 1-50. For all the following slots in the array roll another random number and check if the array already contains it. If it does, roll again until its unique, fill the slot with that random num, repeat until array is full.

## Parson’s Puzzle (7 marks)

1. Assume the screenshot below. When the Draw arrowhead button is clicked, it displays a yellow arrowhead with a blue border on the form. You have been given lines of unordered code to help you write a program that displays this. Select the correct line of code from each pair and write the selected lines of code in the correct order. Add curly braces as necessary. (3 marks)

A screenshot of a computer

Description automatically generated

1. SolidBrush brush = new SolidBrush(Color.Yellow);
2. ~~SolidBrush brush = new SolidBrush(Color.Blue);~~
3. Pen pen = new Pen(Brushes.Blue, 5.0F);
4. ~~Pen pen = new Pen(Brushes.Yellow, 5.0F);~~
5. graphics.FillPolygon(brush, new Point [ ] { new Point(20, 100), new Point(500, 20), new Point(240, 100), new Point(500, 200) });
6. ~~graphics.FillPolygon(pen, new Point [ ] { new Point(20, 100), new Point(500, 20),new Point(240, 100), new Point(500, 200) });~~
7. private void button1\_Click(object sender, EventArgs e)
8. ~~private graphics button1\_Click(object sender, EventArgs e)~~
9. graphics.DrawLines(pen, new Point [ ] { new Point(20, 100), new Point(500, 20), new Point(240, 100), new Point(500, 200), new Point(20, 100) });
10. ~~graphics.DrawArrowhead(pen, new Point [ ] { new Point(20, 100), new Point(500, 20), new Point(240, 100), new Point(500, 200), new Point(20, 100) });~~
11. ~~Graphics graphics = new Graphics();~~
12. Graphics graphics = CreateGraphics();

private void button1\_Click(object sender, EventArgs e)

{

Graphics graphics = CreateGraphics();

SolidBrush brush = new SolidBrush(Color.Yellow);

Pen pen = new Pen(Brushes.Blue, 5.0F);

graphics.FillPolygon(brush, new Point[] { new Point(20, 100), new Point(500, 20), new Point(240, 100), new Point(500, 200) });

graphics.DrawLines(pen, new Point[] { new Point(20, 100), new Point(500, 20), new Point(240, 100), new Point(500, 200), new Point(20, 100) });

}

1. Assume the screenshot below. When the Form is clicked, it displays a grid with X and Y coordinates. You have been given lines of unordered code to help you write a program that displays this. Select the correct line of code from each pair and write the selected lines of code in the correct order. Add curly braces as necessary. (4 marks)

A grid of black and white lines

Description automatically generated with low confidence

// Drawing a string on the X axis

1. ~~graphics.DrawString(x.ToString(), f, Brushes.Black, new Point(0, x));~~
2. graphics.DrawString(x.ToString(), f, Brushes.Black, new Point(x, 0));
3. Font font = new Font("Tahoma", 6, FontStyle.Regular);
4. ~~Font font = new Font("Tahoma", 25, FontStyle.Regular);~~

// Drawing a line on the Y axis

1. graphics.DrawLine(Pens.Black, new Point(0, y), new Point(Width, y));
2. ~~graphics.DrawLine(Pens.Black, new Point(y, 0), new Point(Width, y));~~
3. private void Form1\_MouseClick(object sender, MouseEventArgs e)
4. ~~private void Form1\_MouseEnter(object sender, MouseEventArgs e)~~
5. ~~for (int x = 0; x < Height; x += 20)~~
6. for (int x = 0; x < Width; x += 20)

// Drawing a string on the Y axis

1. ~~graphics.DrawString(y.ToString(), f, Brushes.Black, new Point(y, 0));~~
2. graphics.DrawString(y.ToString(), f, Brushes.Black, new Point(0, y));
3. for (int y = 0; y < Height; y += 20)
4. ~~for (int y = 0; y < Height; y += 40)~~

// Drawing a line on the X axis

1. graphics.DrawLine(Pens.Black, new Point(x, 0), new Point(x, Height));
2. ~~graphics.DrawLine(Pens.Black, new Point(x, 0), new Point(Height, x));~~

private void PongGame\_MouseClick(object sender, MouseEventArgs e)

{

Font font = new Font("Tahoma", 6, FontStyle.Regular);

for (int x = 0; x < Width; x += 20)

{

graphics.DrawLine(Pens.Black, new Point(x, 0), new Point(x, Height));

graphics.DrawString(x.ToString(), font, Brushes.Black, new Point(x, 0));

}

for (int y = 0; y < Height; y += 20)

{

graphics.DrawString(y.ToString(), font, Brushes.Black, new Point(0, y));

graphics.DrawLine(Pens.Black, new Point(0, y), new Point(Width, y));

}

}

(please note for the above code I changed `f` to be `font`)