# Knowledge Assessment

## Criteria

### Unit code, name and release number

ICTPRG418 - Apply intermediate programming skills in another language (1)

### Qualification/Course code, name and release number

ICT50718 - Diploma of Software Development (1)

## Student details

### Student number

808457598

### Student name

Julie Lam

## Assessment Declaration

* This assessment is my original work and no part of it has been copied from any other source except where due acknowledgement is made.
* No part of this assessment has been written for me by any other person except where such collaboration has been authorised by the assessor concerned.
* I understand that plagiarism is the presentation of the work, idea or creation of another person as though it is your own. Plagiarism occurs when the origin of the material used is not appropriately cited. No part of this assessment is plagiarised.

### Student signature and Date

Julie Lam

Version: *1.0*

Date created: *6 July 2018*

Date modified: *04/04/2021*

For queries, please contact:

*Name(s) and position(s)*

*Unit name and address*

© 2020 TAFE NSW, Sydney  
RTO Provider Number 90003 | CRICOS Provider Code: 00591E

This file can be found at: *Document1 (right-click, then choose* ***Update Field*** *before publishing to correct this).*

The contents in this document is copyright © TAFE NSW 2020, and should not be reproduced without the permission of the TAFE NSW. Information contained in this document is correct at time of printing: 4 April 2021. For current information please refer to our website or your teacher as appropriate.

## Assessment instructions

Table 1 Assessment instructions

| Assessment details | Instructions |
| --- | --- |
| **Assessment overview** | The objective of this assessment is to assess your knowledge as would be required to carry out intermediate programming activities using C# language |
| **Assessment Event number** | 2 of 3 |
| **Instructions for this assessment** | This is a written assessment and it will be assessing you on your knowledge of the unit.  This assessment is in 2 parts:   1. Short answer questions 2. Assessment feedback |
| **Submission instructions** | On completion of this assessment, you are required to upload it or hand it to your trainer for marking.  Ensure you have written your name at the bottom of each page of this assessment.  It is important that you keep a copy of all electronic and hardcopy assessments submitted to TAFE and complete the assessment declaration when submitting the assessment. |
| **What do I need to do to achieve a satisfactory result?** | To achieve a satisfactory result for this assessment all questions must be answered correctly. |
| **What do I need to provide?** | Computer with online software, Internet, MS Office, MS Visual Studio, MS SQL Server |
| **What the assessor will provide?** | Online learning resources, Assessments |
| **Due date and time allowed** | 17th November, 2020 |
| **Supervision** | This assessment may take place in the classroom or as a ‘take-home’ task.  The student may access their referenced text, learning notes and other resources. |
| **Assessment feedback, review or appeals** | In accordance with the TAFE NSW policy *Manage Assessment Appeals,* all students have the right to appeal an assessment decision in relation to how the assessment was conducted and the outcome of the assessment. Appeals must be lodged within **14 working days** of the formal notification of the result of the assessment.  If you would like to request a review of your results or if you have any concerns about your results, contact your Teacher or Head Teacher. If they are unavailable, contact the Student Administration Officer.  Contact your Head Teacher for the assessment appeals procedures at your college/campus. |

## Part 1: Short Answers Questions

Answer the following questions:

1. **Describe in detail:**

* **Dynamic variables**

Dynamic variables conceptually avoid compile-time type checking. Instead, the data type is resolved

at run time.

In most cases however, the compiler compiles dynamic types into object types. The actual type is

then resolved at run time. This is based on the assigned value.

For e.g,

dynamic MyDynamicVar = 200;

Resolves data type as integer.

MyDynamicVar = "Hello World!!";

Resolves data type as string.

Another feature of dynamic variables is that it implicitly converts to other types.

For e.g.,

dynamic myDynamicVar = 100;

int myIntVar = myDynamicVar;

Here we’ve declared a dynamic variable and initialised it with an integer value. When we declare an integer variable, we can successfully initialise it with the dynamic variable. This is because the dynamic variable value is stored as an object. When it is then stored in an integer variable behind the scenes the object is converted to an integer.

Based on this same implementation, it is also possible to redefine the dynamic variable with value of a different data type, like so:

myDynamicVar = "Hello";

A drawback of dynamic variables and its’ evasion of compile-time type checking is that the compiler

cannot check for the correct methods and properties of a dynamic type. This can lead to run-time

errors but no compile-time errors.

* **Medium-size application development**

Regardless of the application size, it is good practice to adhere to certain conventions. This includes the following:

• An Agile approach to software development: this iterative approach focuses on delivering working software frequently, collaborating closely with the client, and a commitment to well-written, modularised code.

• Code documentation: eases code maintenance, improves readability, and efficiently allows other developers to work with the source code.

• Issue Tracking: available as software tools to verify and validate the software solution. Such tools record known defects, an explanation of why they exist, and the intended course of correction.

However, as the size of an application grows the necessity for these best practices compound.

For example, a small application which must satisfy a simple, limited criterion may not fully adopt an Agile methodology. As a result, the source code may not be written for modularisation in mind, and therefore would not be accommodating to client changes in the software requirements. In a small application, these changes can be accommodated to a limit even when code has not been written to best practice.

However, as the application size grows, the likelihood of changes in the client’s needs also increases. The complexity and disarray of the source code compounds when trying to accommodate, thereby halting the software development pace.

In addition, the source code for small applications should be a quick read. However, as the application expands, so does the source code. Reading clear and comprehensive code documentation is an efficient method of familiarising oneself with the code and becomes a greater time-saving tool as the source code expands into an unwieldly size.

Finally, keeping an eye on detected bugs or open issues could be achieved by taking a mental note when the application is small enough. However, a larger, medium-sized application will inevitably be open to more bugs. It therefore becomes unfeasible and increasingly prone to human error when attempting to only take a mental note of errors. Issue tracking software creates a log of detected errors and their status and allows multiple stakeholders to quickly check the status of an issue without needing to backtrack through the application.

* **Standard array and file handling algorithms**

Standard array algorithms include sorting and searching algorithms.

There are three basic sorting algorithms: Bubble Sort, Insertion Sort, and Selection Sort.

When describing each algorithm, the assumed goal is to sort an int array from lowest to highest.

* Bubble Sort

This technique compares two adjacent values in an array, and if they’re out of place, they are swapped. This comparison repeatedly occurs with every two adjacent values in the array, the highest number eventually bubbles closer to the end of the array with each iteration.

This is repeated until the second highest number also bubbles to the end of the array, and so on.

This algorithm requires O(n2) exchanges, which reflects that performance deteriorates exponentially as the array size increases.

* Insertion Sort

This technique starts on the left and works towards the right. It compares the current item with the items to its left.

If the current item is smaller than any of the items on the left, then the current item switches positions until it is inserted in to the correct position.

This repeats with the next item in the array being set at the current item, and compared accordingly.

This algorithm requires O(N2) exchanges. Although this approach has the same performance as Bubble Sort, on averages it requires half as many comparisons.

* Selection Sort

This technique focuses on selecting the smallest value from the unsorted array and moving it to the sorted array.

The current minimum is always set to the first value in the unsorted array. This value is then compared to every other number in the array in search for a smaller value. If a value smaller than the initial value is found, the current minimum value is updated accordingly. The comparison continues throughout the entire array.

Once the smallest value is found, its’ position is swapped with the first position in the array.

This is repeated until the second highest number is located, and then this is inserted into the second position.

This algorithm requires O(N) exchanges.

The term File Handling refers to operations such as creating a, reading from, and writing to a file.

Such algorithms for reading or writing to a file involves converting the file into a stream. A stream is a sequence of bytes, and there can be two types: an input stream and an output stream. Input streams are for reading the file, whilst output streams are for writing to the file.

In C#, we may use the StreamWriter class to write to a file. Below is an example snippet of code we could use to do so:

// Creates a file named sample.txt

StreamWriter sw = new StreamWriter("sample.txt");

// Writes onto the console

Console.WriteLine("Enter the Text that you want to write on File");

// Read input from the user

string str = Console.ReadLine();

// Writes a line

sw.WriteLine(str);

// Writes into output stream

sw.Flush();

// Closes the stream

sw.Close();

We could also read from a file using the StreamReader class. Below is an example code snippet we could use to do so:

// Creates a new input stream from the sample.txt file and opens it

StreamReader sr = new StreamReader("sample.txt");

Console.WriteLine("Content of the File");

// Reads line from input stream

string str = sr.ReadLine();

// Reads and prints the file line by line to the console

while (str != null)

{

    Console.WriteLine(str);

    str = sr.ReadLine();

}

Console.ReadLine();

// Closes the stream

sr.Close();

* **User-defined data structures**

A user-defined data structure is a data structure defined by the user. It allows the user to create a data type that represents more than one predefined data type.

A user-defined data structure can contain a collection of variables (or fields), thereby combining multiple data types into a single data type. By declaring a variable as a user-defined data type, all the fields it contains can be accessed simultaneously by calling on the variable object.

Classes are a common example of a user-defined data structure. It allows related properties which may be in different data types to be stored as different fields and can contain methods to allow class behaviour.

Classes allow for inheritance, where the child class inherits all the attributes and methods from the parent and can extend to contain their own specific attributes and/or methods.

Struct is another example of a user defined data structure. Like a class type, a struct type can encapsulate more than one data type and contain behaviour. However, a struct type is typically used to design data-centric types with little to no behaviour. This may be used for example, when reading csv files, and storing each value is associated with a struct property within a struct object.

Enum is another example of a user defined data structure. An Enum is a value type that is used when we want to use values from a set of known values. It is used when we have a known predefined limited set of values.

1. **Describe a limited range of development methodologies and their application**

There are multiple methodologies available for approaching software development. Two current and popular methods are the Waterfall Model and the Agile Scrum Method.

In the Waterfall model, each phase is a distinct step in production. Each phase must first be completed before moving onto the next, and progression occurs in a sequential order.

Each phase of the Waterfall methodology is described below:

1. Determine System Requirements: A requirement specification document is created that outlines all possible requirements of the system.
2. System Design: The system is designed based on the determined system requirements.
3. Implementation: The system is broken down into smaller programs that perform a unique function, these are referred to as units. Each unit is developed and tested in isolation.
4. Integration and Testing: The units are incorporated together to form a larger program. The program is then tested to check on unit cohesion.
5. System Deployment: Once the testing phase is complete, the program is released into the market.
6. Maintenance: Once issues or deficiencies become apparent while in the market updates are released to enhance the program.

This methodology is used with applications where requirements are well distinguished, and changes are unlikely to occur.

Alternatively, the Agile approach to software development is made up of a set of principles, as outlined below:

* The expertise of individuals and their collaboration take precedence over adhering to strict processes and tools
* The production of functional, testable software takes precedence over comprehensive documentation
* Consistent collaboration with the customer takes precedence over meeting contract deadlines
* Expect changes to occur and adapt accordingly rather than follow a rigid plan

The Agile Scrum Process is a lightweight framework built upon the Agile philosophy.

The fundamental structure of Scrum requires a Scrum Master to foster an environment where:

* A Product Owner determines and outlines the work involved for a complex problem onto a Product Backlog. The Product Owner is the expert who is intimately aware of the client's needs. This person acts as the buffer between the developers and the client.
* The Scrum Team completes a selection of the work to produce a unit of value during a Sprint. A Sprint is a fixed duration of time, e.g. a month. A selection of various work is grouped and completed within this period and delivered as a functional component.
* The Scrum Team and its stakeholders review the process and adjust accordingly for the next Sprint.
* The previous steps are repeated.

This methodology is used in applications where changes are expected to occur.

1. **Outline the principles of language development.**

Programming languages allow developers to provide instructions to computers. Like spoken languages, programming languages share many similarities across:

* Syntax and Structure: certain fundamental commands exist across most popular high-level languages, such as if/else statements, for loops, and while loops.
* Functionality: various programming languages can achieve the same functionality although with varying methods.
* Lifespan: programming languages spread and are adopted based on its popularity with developers. Regardless of the language if it falls out common usage then the language eventually becomes deprecated.

Programming languages have evolved over time. Although OOP (Object Oriented Programming) has been implemented extensively in recent history, particularly from the 1960s onwards, OOP as we know it today is quite different to the original creator’s (Alan Kay) vision.

In fact, the rise of high-level languages in the 1950s, such as FORTAN, COBOL and LISP were not OO. LISP instead utilised a functional programming approach.

The first programming language widely recognized as ‘object oriented’ was Simula in 1965. It featured some of the fundamental features that are implemented in OOP languages today, including objects, and eventually classes, class inheritance and virtual methods.

The next object-oriented language was Smalltalk, which was was developed by a group of programmers which included Alan Kay, the one who coined the term “object-oriented”.

According to Alan Kay, the fundamental principles of OOP were:

* Message Passing
* Encapsulation
* Dynamic Binding

Notably, inheritance and polymorphism, fundamental principles in modern OOP were not considered fundamental principles of OOP.

Today, languages are veering away from adhering strictly to only the OOP paradigm. The benefits of functional programming are starting to be incorporated within OO languages. Modern OO languages, such as JS and C#, are becoming more multi-paradigm.

Upload the answer in word document on the moodle.

## Part 2: Assessment Feedback

*NOTE: This section* ***must*** *have the assessor signature and student signature to complete the feedback.*

### Assessment outcome

Satisfactory

Unsatisfactory

### Assessor Feedback

Has the Assessment Declaration on page 1 of the assessment been signed and dated by the student?

☐ Are you assured that the evidence presented for assessment is the student’s own work?

Was the assessment event successfully completed?

If no, was the resubmission/re-assessment successfully completed?

Was reasonable adjustment in place for this assessment event?  
*If yes, ensure it is detailed on the assessment document.*

Comments:

### Assessor name, signature and date:

### Student acknowledgement of assessment outcome

Would you like to make any comments about this assessment?

### Student name, signature and date

***NOTE: Make sure you have written your name at the bottom of each page of your submission before attaching the cover sheet and submitting to your assessor for marking.***