# Skill Assessment

## Criteria

### Unit code, name and release number

ICAPRG418 - Apply intermediate programming skills in another language (1)

ICTICT403 – Apply software development methodologies (1)

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

ICT50718 - Diploma of Software Development

## 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 26th May 2021

Version: *1.0*

Date created: *6 July 2018*

Date modified: *28/05/2021*

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: 28 May 2021. For current information please refer to our website or your teacher as appropriate.

## Assessment instructions

| Assessment details | Instructions |
| --- | --- |
| **Assessment overview** | The objective of this assessment is to assess your skills as would be required to carry out intermediate programming activities using another programming language |
| **Assessment Event number** | 3 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?** | Pen |
| **What the assessor will provide?** | Computers, Internet, Examination Paper |
| **Due date and time allowed** | 27th October, 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 answer

**Unit: Apply intermediate programming skills in another languages Questions**

1. **List five database classes with two properties for each database class.**

The .NET framework includes a variety of data providers. A data provider is used for connected to a database, executing commands, and retrieving results. Each provider provides data access to different data sources. For e.g., the .NET Framework Data Provider for SQL Server provides access to Microsoft SQL Server and utilises the System.Data.SqlClient namespace.

The following database classes belongs to the System.Data.SqlClient namespace:

* SqlConnection: represents a unique session/connection to a SQL Server data source. Its’ properties include the following:
* ConnectionString – gets or sets the string used to open a SQL Server database.
* DataSource – gets the name of the instance of SQL Server to which to connect.
* SqlCommand: represents a Transact-SQL statement or stored procedure to execute against a SQL Server database. Its’ properties include the following:
* CommandText – gets or sets the Transact-SQL statement, table name or stored procedure to execute.
* Connection – gets or sets the SqlConnection used by this instance of the SqlCommand
* SqlDataReader: provides a way of reading a forward-only stream of rows from a SQL Server database. It must be created by executing a SqlCommand object’s ExecuteReader method, rather than directly using a constructor. Its properties include the following:
* HasRows – gets a value that indicates whether the SqlDataReader contains one or more rows
* IsClosed – retrieves a Boolean to indicate whether the SqlDataReader instance has been closed.
* SqlException: represents the exception that is thrown when SQL Server returns a warning or error. Its’ properties include the following:
* LineNumber – gets the line number within Transact-SQL command batch or stored procedure that generated the error
* Class – gets the severity level of the error returned from the .NET Framework Data Provider for SQL Server

1. **Write the statements to populate a SqlDataReader object.**

The code snippet below populates a SqlDataReader object from a database on the local instance of the SQL Server data source called ‘Northwind’.

using System;

using System.Data;

using System.Data.SqlClient;

class Program

{

static void Main()

{

string str = "Data Source=(local);Initial Catalog=Northwind;"

+ "Integrated Security=SSPI";

ReadOrderData(str);

}

private static void ReadOrderData(string connectionString)

{

string queryString =

"SELECT OrderID, CustomerID FROM dbo.Orders;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(queryString, connection);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

// Call Read before accessing data.

while (reader.Read())

{

ReadSingleRow((IDataRecord)reader);

}

// Call Close when done reading.

reader.Close();

}

}

private static void ReadSingleRow(IDataRecord record)

{

Console.WriteLine(String.Format("{0}, {1}", record[0], record[1]));

}

}

1. **List the error codes for detecting violation of foreign key and primary key constraints. – “List the error codes only”**

When executing a SqlCommand’s commandText on a database, the T-SQL statement may violate foreign key and/or primary key constraints. When this occurs, the Sql Server returns an SqlException.

Below is a try catch code snippet to handle a SqlException. This code generates a SqlException and then displays the exception.

public static void ShowSqlException(string connectionString)

{

string queryString = "EXECUTE NonExistantStoredProcedure";

StringBuilder errorMessages = new StringBuilder();

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(queryString, connection);

try

{

command.Connection.Open();

command.ExecuteNonQuery();

}

catch (SqlException ex)

{

for (int i = 0; i < ex.Errors.Count; i++)

{

errorMessages.Append("Index #" + i + "\n" +

"Message: " + ex.Errors[i].Message + "\n" +

"LineNumber: " + ex.Errors[i].LineNumber + "\n" +

"Source: " + ex.Errors[i].Source + "\n" +

"Procedure: " + ex.Errors[i].Procedure + "\n");

}

Console.WriteLine(errorMessages.ToString());

}

}

}

1. **Write the statements to populate a combo box from the following database table using DisplayMember (Coursename) and ValueMember (Courseid) properties of the combo box:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table Name: Course** | | | | |
| **Columns** |  |  |  |  |
| **Column Name** | **Data Type** | **Primary Key** | **Not null** | **Auto Increment** |
| **Courseid** | **smallint** | **Yes** | **Yes** | **Yes** |
| **Coursename** | **varchar(40)** | **No** | **Yes** | **No** |
| **Courseduration** | **smallint** | **No** | **Yes** | **No** |

The following code snippet populates a combo box from the database.

public Page1()

{

// Populate the combobox with a List of Course objects

List<Course> courses = retrieveCourses(); //The method retrieveCourses() connects to the database and returns a list of Courses

ComboBox myComboBox = new ComboBox();

myComboBox.DataSource = courses;

myComboBox.DisplayMember = “Coursename”;

myComboBox.ValueMember = “Courseid”

}

public List<Course> retrieveCourses()

{

List<Course> Courses = new List<Course>();

//Connects to DB, retrieves course names.

string connectionString = "localhost\\SQLEXPRESS;Database=cSharpSkillsDB; Integrated Security=true;";

using (SqlConnection myConn = new SqlConnection(connectionString))

  {

   using (SqlCommand sqlCommand = new SqlCommand("SELECT \* FROM Course", myConn))

     {

       myConn.Open();

        SqlDataReader reader = sqlCommand.ExecuteReader();

         while (reader.Read())

          {

            Courses.Add(new Course()

              {

                Courseid = reader.GetInt16(0),

                 Coursename = reader.GetString(1),

                 duration = reader.GetInt16(2)

               });

           }

       }

  }

        return Courses;

}

1. **Write the statements to populate a generic List<clsUnit> from the database table Unit where clsUnit is the class name with the following details:**

**Table Name: Unit**

**Column Name Data Type Primary Key Not null Auto Increment**

**Unitid smallint Yes Yes Yes**

**Unitname varchar(40) No Yes No**

**Unitduration smallint No Yes No**

**Class Name: Unit**

**Field Name Data Type**

**Unitid Integer**

**Unitname String**

**Unitduration Integer**

The following code snippet populates a List<clsUnit> with Unit objects instantiated using information from the database.

public Page1()

{

List<clsUnit> myUnits = retrieveUnits();

}

public List<clsUnit> retrieveUnits()

{

    List<clsUnit> Units = new List<clsUnit>();

    //Connects to DB, retrieves Units.

    string connectionString = "Server=localhost\\SQLEXPRESS;Database=cSharpSkillsDB; Integrated Security=true;";

    using (SqlConnection myConn = new SqlConnection(connectionString))

    {

        using (SqlCommand sqlCommand = new SqlCommand("SELECT \* FROM Course", myConn))

        {

            myConn.Open();

            SqlDataReader reader = sqlCommand.ExecuteReader();

            while (reader.Read())

            {

                Units.Add(new clsUnit()

                {

                    Unitid = reader.GetInt16(0),

                    Unitname = reader.GetString(1),

                    Unitduration = reader.GetInt16(2)

                });

            }

        }

    }

    return Units;

}

1. **Write statements to create a decimal two dimensional array containing four rows and five columns. Display all the elements of this array using nested loops.**

decimal[,] array2Da = new decimal[4, 5] {

  {1.1M, 2.2M, 3.3M, 4.4M, 5.5M},

  {3.1M, 4.1M, 5.1M, 6.1M, 7.1M},

  {5.1M, 6.1M, 7.1M, 8.1M, 9.1M},

  {7.2M, 8.2M, 9.2M, 10.2M, 11.2M}

};

for (int i = 0; i < 4; i++)

{

    for (int j = 0; j < 5; j++)

    {

        Console.WriteLine(array2Da[i, j]);

    }

}

1. **List the difference between out and ref keywords.**

Both the terms ‘out’ and ‘ref’ are available as parameter keywords in C#. These terms are used when a method returns multiple values. This means when a parameter passes with either the ‘out’ or ‘ref’ keyword in the method, then that method can manipulate the same value that was passed in.

See below for an example:

public TwoDArray()

{

    InitializeComponent();

    string name = "John";

    string greeting = "Good Morning";

    string uppercaseName = upperName(name, out greeting);

Console.WriteLine("Name in uppercase: " + uppercaseName); //Displays ‘Name in uppercase: JOHN’

Console.WriteLine("Converted greeting: " + greeting); //Displays ‘Converted greeting: Good Night’

}

public string upperName(string name, out string greeting)

{

    greeting = "Good Night";

    return name.ToUpper();

}

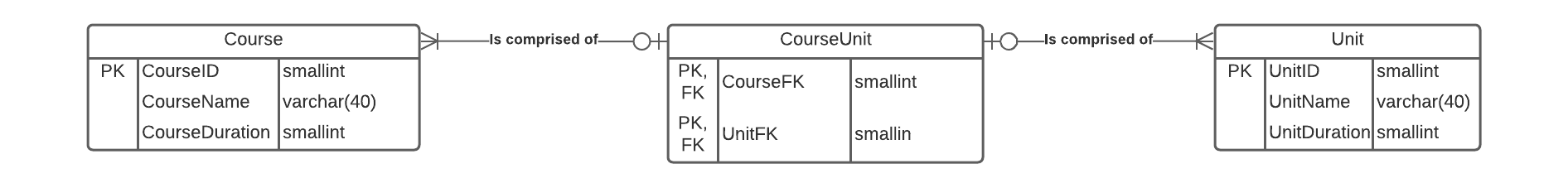
The predominant difference between the two keywords is that ‘out’ is used to state that the parameter passed **must** be modified by the method, whereas ‘ref’ is used to state that the parameter passed **may** be modified by the method.

1. **Create a database diagram for the following scenario: - “Correct the relationships between parent and child tables”**

Course

Unit

Below is a Crow’s Foot ERD diagram.



1. **Write the create statements to create tables, primary keys, composite primary keys and foreign keys for this scenario. – “Foreign keys in the child table CourseUnit missing”**

create DATABASE cSharpSkillsDB

create TABLE Course (

Courseid smallint NOT NULL IDENTITY(1,1) PRIMARY KEY,

Coursename varchar(40) NOT NULL,

Courseduration smallint NOT NULL,

)

create TABLE Unit (

Unitid smallint NOT NULL IDENTITY(1,1) PRIMARY KEY,

Unitname varchar(40) NOT NULL,

Unitduration smallint NOT NULL

)

CREATE TABLE CourseUnit (

CourseFK smallint NOT NULL,

UnitFK smallint NOT NULL,

PRIMARY KEY (CourseFK, UnitFK)

)

1. **Write the statements to populate a data grid view object from the Course table.**

Please see below for a code snippet that populates a DataGridView object named myDataGridView.

string connectionString = "Data Source=localhost\\SQLEXPRESS; Initial Catalog=cSharpSkillsDB;Integrated Security=True";

using (SqlConnection connection = new SqlConnection(connectionString))

{

    connection.Open();

    SqlCommand command = new SqlCommand("SELECT \* FROM Course", connection);

    SqlDataAdapter adapter = new SqlDataAdapter();

    adapter.SelectCommand = command;

    DataTable table = new DataTable();

    adapter.Fill(table);

    myDataGridView.AutoGenerateColumns = true;

    myDataGridView.DataSource = table;

}

**Unit: Apply software developments methodologies**

1. **List and describe two software development methodologies.**

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

Both methodologies progress through the standard stages of the System Development Life Cycle (SDLC). However, while the Waterfall model takes on a linear sequence of progression, the Agile method takes on an iterative sequence of progression.

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.

Alternatively, the Agile model is an iterative process with repeated and incremental improvements.

Each iteration delivers a working program build. With each additional iteration, there is the opportunity to review the program and either remove deemed to be unnecessary features or incorporate additional features.

Unlike the Waterfall model, Agile takes on an adaptive approach. The planning phase is not as detailed (instead product requirements are expected to change dynamically), and development adjusts accordingly.

1. **Which diagrams provide a dynamic view of a system (that is, show change**

**over time)? – “Sequence diagram is not a dynamic diagram ”**

Dynamic, aka Behavioural diagrams model the behaviour of the system over time. Such diagrams include:

* Sequence Diagrams: displays the interaction between users, screens, objects and entities within the system by visually sequentially documenting the passage of messages between objects over time.
* Activity Diagrams: displays how different workflows in the system are constructed, how they start, and the various possible decision paths that can take place from start to finish.
* State Diagrams: displays how an object changes from one state to another and the rules that govern that state.

1. **List any four static diagrams in UML modelling and their purposes. – “Fourth static diagram is missing”**

Static, aka Structural Diagrams models the static structure of the system with the use of objects, attributes, operations and relationships.

Such diagrams include:

* Class Diagrams: displays a set of classes, their attributes, operations (or methods), and the relationships between them.
* Composite Structure Diagrams: displays the internal structure and components of a class. Aggregated classes forms parts of a class but may not necessarily be a class itself, where a part is any element that is used to construct the containing class.
* Object Diagram: displays a set of objects and their relationships. It provides a static view of the object instances found in the class diagram.

1. **What is an object? How do objects work together?**

An Object is an instance of a Class. A class, on the other hand, is a user defined data structure. It acts as a template that define the properties and/or methods of the object instances of the class.

An object instance is a specific object created from a particular class. An object may contain data, (in the form of properties), and methods (code blocks that provide a functionality).

Objects can access and often modify the property values of itself, or of other objects.

1. **What information can you show on a class icon?**

A class icon is assumed to be referring to a Class Notation found in a class diagram. As such, it consists of three parts:

1. Class Name: The name of the class. For example, ‘MyClass’
2. Class Attributes: The attributes or fields of the class, followed by a colon and then the attribute type. For example, ‘attribute1 : int’
3. Class Operations (Methods): The functionalities of the class, followed by open and close brackets that define the method parameters. This is followed by a colon and then the return type of the method. For example, ‘TakeOrders() : boolean’

In addition, the access modifiers can be defined for each attribute and operation. This may be:

* Public (+)
* Private (-)
* Protected (#)
* Package (~)
* Derived (/)
* Static (underlined)

Finally, the relationships between classes can be demonstrated as:

* Inheritance: when a child class takes on the functionality of the parent class, this relationship is visualised with a straight connected line with a closed arrowhead pointing towards the parent class.
* Bidirectional association: when two classes are aware of each other, this relationship is visualised with a straight line between the two classes.
* Unidirectional association: when only one class is aware of the other class, this relationship is visualised with a straight connecting line that points an open arrowhead from the knowing class to the known class.

1. **Why is it necessary to have use case diagrams in UML modelling?**

A UML use case diagram specifies the expected behaviour of a software, and demonstrates the relationship between use cases, actors and the system.

Use Case Diagrams are necessary and useful for the following reasons:

* Used to gather the requirements of a system
* Used to get an external view of a system
* To identify the external and internal factors that influence the system
* Show the interaction among the requirements and actors

## 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.***