SESSION 1. INTRODUCTION

SIT232 - OBJECT ORIENTED DEVELOPMENT

Administrative Information

Unit Staff:

Dr (Duc) Thanh Nguyen
Unit Chair and Lecturer for Geelong and Cloud-Linked

Email: duc.nguyen@deakin.edu.au
Phone: +61 3 5247 9506
Website: ducthanhnguyen.weebly.com

Dr Sergey Polyakovskiy
Lecturer for Burwood

Email : sergey.polyakovskiy@deakin.edu.au
Room: T2.19
Phone : +61 3 92468813

Administrative Information

EMERGENCY EVACUATION INFORMATION

STUDENT STATEMENT

The Chief Operating Officer has requested the following statement to be read to all students by permanent and sessional teaching staff at the start of all lectures, classes and tutorials during the first week of the first and second semester.

"The University wishes to make you aware of the Emergency Evacuation Procedures that are in place for your safety.

Please look around and familiarize yourself with the exits in this room and for each new venue in which you attend classes. You should also familiarise yourself with the building's emergency exits and assembly areas which can be found on Fire Emergency Floor Plans placed in the hallway of all buildings.

In the unlikely event of an evacuation, you will hear a continuous loud beeping tone. On this alert signal you should prepare yourself to evacuate the building.

If you hear a continuous two tone siren sound please evacuate in an orderly fashion to the assembly areas. All building occupants must evacuate on this evacuation signal.

Remember not to use lifts during evacuation.

Please follow the instructions given by Emergency Wardens, Security staff or Emergency Services personnel.

At the end of the evacuation, you will be advised when it is safe to return to the building."

Unit Materials

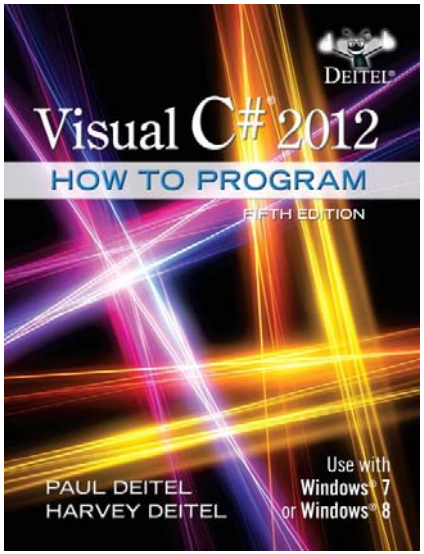
Materials available in CloudDeakin:

- Unit guide
- News
- Workbook (use it as your textbook)
- Practicals
- Lecture slides
- Training videos

Unit Materials

Recommended books:

- Rough, J., Object-Oriented Development: Workbook, 2016.
- Deitel, P., and Deitel, H., Visual C# 2012: How to program, 5th edition, Pearson Education, 2013



Other materials:

- Microsoft developer network (MSDN) http://msdn.microsoft.com
- Deakin library

Timetable and Consultation (Geelong)

Lectures

Tuesday	14:00 – 14:50	KA4.406

Practical sessions

Tuesday	10:00 – 131:50	KA5.304
Tuesday	12:00 – 13:50	KA5.304

- You could also email me for you concerns. Your email's subject should start with SIT232. Normally I would respond to emails within 48 hours.
- You could also post your concerns on Discussion and seek answers from your classmates.
- If you need in-person consultations, please book for appointments.

Timetable (Burwood)

Lectures		
Tuesday	14:00 – 14:50	LT12 (X2.05)
Practical sessions		
Thursday	08:00 – 09:50	HE2.011
Monday	11:00 – 12:50	HE2.011
Monday	18:00 – 19:50	HE2.011
Tuesday	18:00 – 19:50	HE2.011
Friday	17:00 – 18:50	HE2.011
Thursday	18:00 – 19:50	HE2.011

- I am happy to see you all any time when I am in the office T2.19.
- It's better to send me a message first as am teaching most of the time.
- Use the forum if you think there is something to discuss and other students may be involved.

Assessment

- **Weekly practical tasks** (worth of 10%)

  You will get maximum 1 mark per practical
  and 10 marks for all the practicals.

- **Two programming projects** (each is worth of 15%)

  See CloudDeakin for further information including submission instructions

- **Examination (worth of 60%)**

To pass you only require an overall mark of 50

Contact the unit chair if you have any concerns about satisfying the requirements of the unit as soon as possible

Plagiarism

- Plagiarism is the copying of another person's ideas or expressions without appropriate acknowledgment and presenting these ideas or forms of expression as your own.
- It includes not only written works such as books or journals but data or images that may be presented in tables, diagrams, designs, plans, photographs, film, music, formulae, web sites and computer programs.
- Plagiarism also includes the use of (or passing off) the work of lecturers or other students as your own.

Plagiarism

Please be aware that if the Faculty Academic Progress and Discipline Committee finds a student has committed an act of academic misconduct it may impose one or more of the following penalties

- Allocate a zero mark or result or other appropriate mark or result for the assessment task;
- Allocate a zero mark or result or other appropriate mark or result for the Unit;
- Suspend from a Unit or a Course for up to 4 Study Periods;
- Exclude from the University;
- Pay the cost of investigating the misconduct;
- Require the Student to refrain from association with specified person/s for purposes of study or assessment;
- Reprimand and caution the student;
- Require resubmission of one or more assessment tasks;
- Require a student to undertake alternative assessment for the Unit on terms determined by the faculty committee;
- Terminate candidature; Recommend to the vice-chancellor or nominee that the degree not be awarded

For this Unit

Working with other students

- You may discuss/collaborate with other students to better understand a problem and to determine an approach to solving the problem.
- You are not permitted to share your solutions (whether finished or in progress) with other students under any circumstances.
- Your assignment submission, i.e., code, documentation, etc., must be entirely your own work.

Referencing sources

- Any code that has been copied/adapted should be clearly referenced (including any code from assignment questions)

  Note there should be little need for this anyway, otherwise you are not learning the content well enough to pass

Unit outline

1. Introduction
2. Control structures, classes, and data types
3. Methods and arrays
4. Relationships
5. Inheritance
6. Polymorphism
7. UML
8. Object-oriented modeling
9. Exception handling
10. Strings and regular expressions
11. Files and generics

Learning objectives

1. Apply object-oriented concepts including abstraction, encapsulation, inheritance, and polymorphism.
2. Solve programming problems using object-oriented techniques and the C# programming language.
3. Modify an existing object-oriented application to satisfy new functional requirements.
4. Construct object-oriented designs and express them using standard UML notation.

Introduction to Object Oriented Development

Outline

- Objectives
- Introduction to OOD
- Introduction to C#
- Variables
- Literals
- Data Type Conversion
- Arithmetic Expressions
- Syntax Errors

Objectives

At the end of this session you should:

- Be familiar with the overall architecture of object- oriented programs and have learned fundamental concepts including objects, classes, attributes, operations, abstraction, and encapsulation;
- Able to construct simple programs in the C# programming language using variables, expressions, and console input/output; and
- Be able to identify syntax errors in a C# program and be able to remove them.

Introduction to OOD

- Our study of programming to date has considered:
  - Variables
  - Decision structures
  - Looping structures
  - Methods/functions

- How do we translate this into developing complex applications?

Introduction to OOD

- Object-oriented applications consist of a collection of cooperating objects exploiting
  - *Abstraction*: focus on the important/relevant aspects of the problem while ignoring unimportant/irrelevant
  - *Encapsulation*: maintain a separation between the interface to an object and its implementation

- Consider driving a car
  - Abstraction: start/stop engine, accelerate, brake, turn
  - Encapsulated: petrol system, cylinders, exhaust system, drive shaft, differential, and much more

Introduction to OOD

- Objects are the fundamentalbuilding block which can either be
  - *Real*, such as a light switch, student, book, keyboard
  - *Virtual*, such as an array, queue, textbox, avatar

- Each object has
  - *Attributes*: a small piece of data describing the object
    - Examples: on/off, ID number, queue length/no. of print jobs
  - *Operations*: how the object behaves (acts and reacts)
    - Examples: turn on, enrol student, add element to queue

Introduction to OOD

- We define objects by writing classes

  An object is an instance of a class, the class acts as a template for the object/s

- A class definition consists of
  - Variables
  - Properties
  - Methods
- These represent the attributes and operations of an object's interface

Introduction to C#

In this unit we will be using the C# programming language

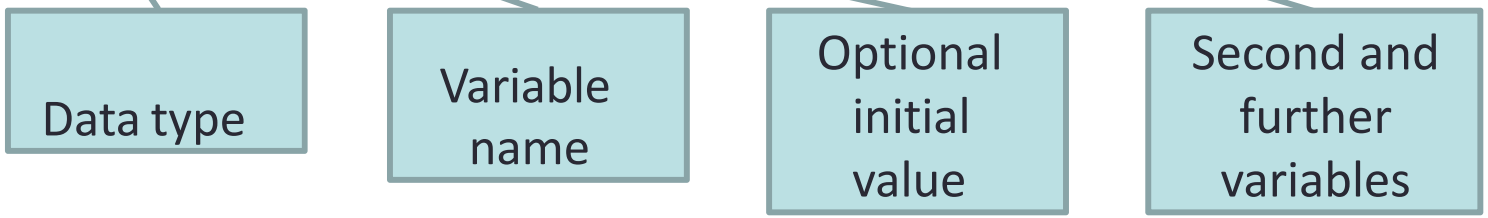
- Modern language originally developed by Microsoft but since has been standardised by ECMA and ISO
- C# uses a syntax core used by almost all the dominant languages including C, C++ and Java
- Becoming a very important programming language

Variables

- There is often a need to store some data
  - To reserve some memory, we must provide two pieces of information:
    - The data type, i.e., what sort of data is to be stored
    - A name to call this storage location
  - This storage is known as a variable

- Syntax:

type name1[= value][, name2[= value][, ...]];



Variables

A variable that stores a single piece of information is known as a *simple data type*

Simple data types provided by C#:

Category	Size	C# Data Type	Convert Support	Values
Signed integral	8	sbyte	Convert.ToSByte	-128 to 127
	16	short	Convert.ToInt16	-32,768 to 32,767
	32	int	Convert.ToInt32	-2,147,483,648 to 2,147,483,647
	64	long	Convert.ToInt64	-9,233,372,036,854,775,808 to 9,223,372,036,854,775,807
Unsigned integral	8	byte	Convert.ToByte	0 to 255
	16	ushort	Convert.ToUInt16	0 to 65,535
	32	uint	Convert.ToUInt32	0 to 4,294,967,295
	64	ulong	Convert.ToUInt64	0 to 18,446,744,073,709,551,615
Floating point	32	float	Convert.ToSingle	1.5*10-45 to 3.4*1038, 7-digit precision
	64	double	Convert.ToDouble	5.0*10-324 to 1.7*10308, 15-digit precision
High-precision decimal	128	decimal	Convert.ToDecimal	1.0*10-28 to 7.9*1028, 28-digit precision
Boolean	8	bool	Convert.ToBoolean	true or false
Unicode character	16	char	Convert.ToChar	Any character from the Unicode character set (all letters, numbers, symbols, control characters, etc.)

Literals

Literals are the values assigned to variables, used in calculations, used in method calls, and so on

- int, e.g., 55;
- uint, e.g., 55U or 55u;
- long, e.g., 55L or 55l;
- ulong, e.g., 55UL, 55ul, 55Ul, 55uL, 55LU, 55lu, 55lU, or 55Lu;
- float, e.g., 5.5F or 5.5f;
- double, e.g., 5.5, 5.5D, 5.5d;
- bool – true or false;
- char, e.g., 'X';
- decimal, e.g., 5.5M or 5.5m;
- string, e.g., "Welcome to SIT232!".

Data Type Conversion

There is often a need to change how data is represented in memory, i.e., to change data type

- Everything a user types in is in a textual format (character or string)
- Numbers must be converted before they can be used in mathematical formulae
- Textual data is also extremely inefficient at storing numeric data

Data Type Conversion

To convert between data types, we usethe Convert utility, e.g.,

```
string sourceData = "55";
int destinationData = Convert.ToInt32(sourceData);
```

Category	Size	C# Data Type	Convert Support	Values
Signed integral	8	sbyte	Convert.ToSByte	-128 to 127
	16	short	Convert.ToInt16	-32,768 to 32,767
	32	int	Convert.ToInt32	-2,147,483,648 to 2,147,483,647
	64	long	Convert.ToInt64	-9,233,372,036,854,775,808 to 9,223,372,036,854,775,807
Unsigned integral	8	byte	Convert.ToByte	0 to 255
	16	ushort	Convert.ToUInt16	0 to 65,535
	32	uint	Convert.ToUInt32	0 to 4,294,967,295
	64	ulong	Convert.ToUInt64	0 to 18,446,744,073,709,551,615
Floating point	32	float	Convert.ToSingle	1.5*10-45 to 3.4*1038, 7-digit precision
	64	double	Convert.ToDouble	5.0*10-324 to 1.7*10308, 15-digit precision
High-precision decimal	128	decimal	Convert.ToDecimal	1.0*10-28 to 7.9*1028, 28-digit precision
Boolean	8	bool	Convert.ToBoolean	true or false
Unicode character	16	char	Convert.ToChar	Any character fromthe Unicode character set (all letters, numbers, symbols, control characters, etc.)

Arithmetic Expressions

Expressions are a critical component of every program you will ever write

- Expressions are constructed using an *operator* and one or more *operands*
- Three types of operators
  - Unary operator: uses a single operand, e.g., -a
  - Binary operator: uses two operands, e.g., a + b
  - Ternary operator: uses three operands (examined later)

Arithmetic Expressions

Arithmetic expressions are expressions used to perform simple mathematical calculations, i.e., arithmetic

- There are five arithmetic operators:
  - Addition: **oper1 + oper2**
  - Subtraction: **oper1 – oper2**
  - Multiplication: **oper1 * oper2**
  - Division: **oper1 / oper2**
  - Remainder after division: **oper1 % oper2**
- It is also possible to combine these operators into *compound expressions*, e.g., a / b * c

Arithmetic Expressions

- For compound expressions, there is an order to which the operators must be performed, e.g., consider 8 / 2 * 4
  - If division is done first, the answer is 4 * 4 = 16
  - If multiplication is done first, the answer is 8 / 8 = 1
- The ordering is defined by *precedence* and *associativity*
  - Precedence defines the priority of operators, where highest priority is performed first
  - Associativity defines the order to perform operators that have equal precedence
    - Left associative– perform operators from left to right
    - Right associative– perform operators from right to left

Arithmetic Expressions

Primary	`x.y  f(x)  a[x]  x++  x--  new  typeof  checked  unchecked`
Unary	`+  -  !  ~  ++x  --x  (T)x`
Multiplicative	`*  /  %`
Additive	`+  -`
Shift	`<<  >>`
Relational and type testing	`<  >  <=  >=  is  as`
Equality	`==  !=`
Logical AND	`&`
Logical XOR	`^`
Logical OR	`|`
Conditional AND	`&&`
Conditional OR	`||`
Conditional	`?:`
Assignment	`=  *=  /=  %=  +=  -=  <<=  >>=  &=  ^=  |=`

- Precedence: highest to lowest
- Associativity: all operators are left associative except the assignment and conditional operators which are right associative

Source: http://msdn.microsoft.com/en-us/library/aa691323(VS.71).aspx

Syntax Errors

Every programming language defines a syntax including:

- The format of a statement;
- Mathematical expressions;
- Control structures (if, for, while, etc.);
- Function definition and invocation;
- Class definition and instantiation (object creation);
- An so on.

Syntax Errors

- When interpreting code that you have written, if the code does not match the syntax rules the compiler will report a *syntax error* indicating:
  - Which file the error was found in;
  - The line where the error was found; and
  - A general descriptive message.
- Note: Compilers generally start at the beginning of a file and process instructions one line at a time.
  - A compiler may not find a syntax error until several lines later, so always check the lines above as well

Summary

- Administration Information
- Session 01. Introduction
  - Objectives
  - Introduction to OOD
  - Introduction to C#
  - Variables
  - Literals
  - Data Type Conversion
  - Arithmetic Expressions
  - Syntax Errors

Summary

Training Videos:

- VS: Create, Build, and Run C# Projects
- VS: Getting Started Developing with C#
- C#: Variables and Literals
- C#: Console I/O Basics
- C#: Arithmetic Expressions
- VS: Working with Syntax Errors