# COSC 1530

# Introduction to Computer Use II: Programming

Winter 2005 (Section M)

Topic A: Introduction to Problem Solving and Visual Basic

Friday, January 6 2006

Bill Kapralos



## Overview (1):

- **Before We Begin**
  - Some administrative details
  - Some questions to consider
- **Introduction to Computer Programming**
  - What exactly is computer programming ?
  - What is the purpose of computer programming ?
  - Programming Paradigms

## Overview (2):

- **Object Oriented Programming**
  - Fundamental concepts
- **Introduction to Programming with MS Visual Basic 6.0**
  - MS Visual Basic 6.0
  - The VB Integrated Development Environment (IDE)

## Before We Begin

## Administrative Details (1):

- **My Office**
  - Computer Science & Engineering (CSE) 2015

## Some Questions to Consider (1):

- What is computer programming ?
- What is a computer program ?
- What is an algorithm ?
- What should you strive for when writing an algorithm ?
- What is the most effective way to learn any programming language ?

# Introduction to Programming (cont.)

## Intro. to Computer Programming (8):

- **Why Program Computers ? (cont.)**
  - Amazingly, everything we have seen a computer do is the result of a sequence of extremely simple arithmetic and logic operations done very quickly
  - The challenge of computer programmers is to come up with the instructions to put those simple instructions together in ways that are useful and appropriate

## Intro. to Computer Programming (9):

- **Why Program Computers ? (cont.)**
  - Computers all around us → programming computers is not solely meant for computer scientists! Computer programming can be a benefit for many other disciplines as well
    - Mathematics and statistics
    - Economics
    - Psychology
    - Physical and natural sciences etc.
  - Understanding software development is a crucial element of many professional career paths

## Intro. to Computer Programming (10):

- **How Do We Program Computers ?**
  - We "talk" to computers using programming languages
    - A series of instructions written by a programmer according to a given set of rules or conventions ("syntax")
    - Similar to human languages used in human-to-human communication → many types of languages, each with its own grammar and syntax
    - Typically we start with an algorithm and then covert it to the appropriate programming language
    - The algorithm is "generic" and can be converted to any programming language

## Intro. to Computer Programming (11):

- **Programming Languages**
  - A great number of programming languages are currently in existence → no one language is suitable for writing all types of programs
  - Some common example programming languages include
    - Java, C/C++, Basic, COBOL, Pascal, Fortran etc.
    - Each has its own syntax
    - Generally, if you become proficient with one language, its easy to pick up another one → just a matter of learning the syntax
  - Can separate programming languages into low-level and high-level

## Intro. to Computer Programming (12):

- **Low-Level Programming Language**
  - A language that provides little or no abstraction from a computer's microprocessor
  - "Low" does not imply inferior to high level programming languages but rather refers to the reduced amount of abstraction (e.g., hiding of details or specifics) between the language and itself
    - Sometimes called machine language → numeric code to represent the most basic computer operations
    - Typically used by specialist e.g., computer scientists

## Intro. to Computer Programming (13):

- **Low-Level Programming Language (cont.)**
  - Sample code listing (8086 assembly language)

```
_dt_ym0 proc far
push ax
push bx
push dx
xor ax,ax
mov dx,TIMER1_CNT
in al,dx
or al,al
jnz FIXCOUNT0
xor ax,ax
out dx,ax
```

## Intro. to Computer Programming (14):

- **High-Level Programming Language**
  - Notation uses English-like words and phrases → instructions are typically readable and can make sense without having to be a computer scientist!
    - Makes it possible for scientists, engineers and business people to solve problems using familiar terminology and notation rather than cryptic machine instructions
    - Of course, there must exist a facility to convert these high level commands into a form understandable by the machine → this is the job of the compiler

## Intro. to Computer Programming (15):

- **High-Level Programming Language (cont.)**
  - Sample code listing (Java code)

```
public class PairOfDice {

  public int die1;   // Number showing on the first die.
  public int die2;   // Number showing on the second die.

  public PairOfDice() {
     roll();  // Call the roll() method to roll the dice.
  }

  public PairOfDice(int val1, int val2) {
     die1 = val1;  // Assign specified values to the instance variables.
     die2 = val2;
  }

} // end class PairOfDice
```

## Programming Paradigms (1):

- **Main Types of Programming Languages**
  - Procedural programming
    - The program specifies the exact sequence of all operations
    - Programming logic determines the next instruction to execute in response to conditions and user requests
    - Example programming languages includes → Basic, Fortran, Pascal, C

## Programming Paradigms (2):

- **Main Types of Programming Languages (cont.)**
  - Object oriented and event driven programming
    - Event driven programs are no longer procedural → do not follow a sequential logic
    - As a programmer you do not take control and determine the sequence of operations → the user takes over!
    - User can press keys, click on certain buttons within a window etc. → these actions cause an event to occur which triggers a particular procedure (sub-program) you have written to execute in response to the event

## Programming Paradigms (3):

- **Object Oriented Programming**
  - Also known as "OOP" for short
  - General idea
    - A computer program may be seen as composed of a collection of individual units, or objects, that act on each other, as opposed to a traditional view in which a program may be seen as a collection of functions or procedures or simply as a list of instructions to the computer
    - Each object is capable of receiving messages, processing data, and sending messages to other objects

## Programming Paradigms (4):

- **Object Oriented Programming (cont.)**
  - More flexibility, changes to programs can be easily made and is widely popular in large scale software engineering
    - Allows for re-use of code
  - Proponents of OOP claim that
    - OOP is easier to learn for those new to computer programming than previous approaches
    - Software (code) is often simpler to develop and to maintain, lending itself to more direct analysis, coding, and understanding of complex situations and procedures than other programming methods.

## Programming Paradigms (5):

- **Object Oriented Programming (cont.)**
  - Most OOP programs are also event driven
  - Visual Basic is an example of an object oriented, event driven programming language
    - Although it is considered to be an OOP language, it doesn't contain all elements of an OOP language such as Java for example
    - Each new release does bring it closer to a true OOP language however

# Object Oriented Programming

## Fundamental Concepts (1):

- **Central Concepts to OOP**
  - Object → an entity that you can manipulate in your program generally by calling methods associated with the object
    - Think of it as a "thing" (e.g., a noun)
    - Think of an object as a "black box" with a public interface (methods you can call) and a hidden implementation (the code and data that are necessary to make these methods work)
    - As a user of the object, you don't have to worry about the implementation details!

## Fundamental Concepts (2):

- **Central Concepts to OOP (cont.)**
  - Class → defines the methods (e.g., sub-programs that perform (compute) a particular task) and any properties for the objects
    - Every object belongs to a class → every object is an instance of a particular class
    - Classes are "factories" for objects (e.g., used to generate objects)
    - Can have many objects of a particular class and the properties of each object may have different values

## Fundamental Concepts (3):

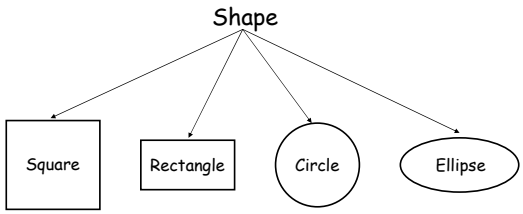
- **Central Concepts to OOP (cont.)**
  - Inheritance → a mechanism for enhancing existing classes
    - Suppose you need to implement (create) a new class and a class representing a more general concept is available, then the new class can use (inherent) from the existing class
    - Suppose you have a class called "Shape" but you need to define a "Rectangle" → a rectangle is itself a shape albeit a specific shape!

## Fundamental Concepts (4):

- **Central Concepts to OOP (cont.)**
  - Properties → tell us something about an object such as its name, color, size, location or how it behaves
    - Think of properties as "adjectives" that describe objects (which have been defined as "nouns")
  - Methods → associated with objects
    - Think of methods as "verbs"
    - Perform a specific task
    - For example, in Shape class, we may have a method called "getArea" that returns the area of the shape object → we don't know how it's calculated but we know we get the area (abstraction)

## Fundamental Concepts (5):

- **Putting It All Together**
  - An example → shape class




## Fundamental Concepts (6):

- **Putting It All Together (cont.)**
  - Shape class is the more general definition and then we have specific shapes such as squares, circles etc.
    - Each shape shares certain general properties such as area but calculating area is specific to the type of shape being considered (e.g., calculating area for circle is different than calculating area for square)
    - Each shape can inherent the area property from the general Shape class but then define a method that will calculate the specific area for the shape
    - Can have many instances of a particular shape

## Fundamental Concepts (7):

- **Putting It All Together (cont.)**
  - Keep in mind that OOP is a very big topic → we can spend an entire course on OOP
    - In this course we are not going to any great details regarding OOP but we should just be aware of some of the main concepts since VB is itself an OOP language e.g., have an idea of what an object is and what methods and properties are
    - More on this as we work with VB

## Fundamental Concepts (8):

- **Putting It All Together (cont.)**