NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Chapter 1: Object-Oriented Modelling

CE2002 Object Oriented Design & Programming

Dr Zhang Jie
Assoc Prof, School of Computer Science and Engineering (SCSE)

Learning Objectives

After the completion of this chapter, you should be able to:

- Differentiate between procedural and object-oriented programming.
- Explain object-oriented concepts.

Topic 1: Procedural vs. Object-Oriented Programming

Programming Paradigm

- Unstructured Programming (UP)
  - Example, Assembly language, COBOL, Basic
- Procedural Programming (PP)
  - Example, C, Pascal, ADA, Fortran
- Object-oriented Programming (OOP)
  - Java, C++, C#, Objective C, Smalltalk

```
clrscr proc near
  mov ax,0b800h
  mov es,ax
  mov di,0
  mov al,' '
  mov ah,07d
  loop_clear_12:
      mov word ptr es:[di],ax
      inc di
      inc di
      cmp di,4000
      jle loop_clear_12
      ret
endp
```

```
10  INPUT guess
20  GOTO 40
30  PRINT "This line is skipped"
40  IF guess = answer THEN GOTO 60
50  GOTO 10
60  PRINT "You've won a million"
```

History of Programming Languages

Programming Community Index for August 2011

Position Aug 2011	Position Aug 2010	Delta in Position	Programming Language	Ratings Aug 2011	Delta Aug 2010	Status
1	1	=	Java	19.409%	+1.42%	A
2	2	=	C	17.390%	-0.48%	A
3	3	=	C++	8.433%	-1.23%	A
4	4	=	PHP	6.134%	-3.05%	A
5	6	↑	C#	6.042%	+1.06%	A
6	9	↑↑↑	Objective-C	5.494%	+2.34%	A
7	5	↓↓	(Visual) Basic	5.013%	-0.40%	A
8	7	↓	Python	3.415%	-0.81%	A
9	8	↓	Perl	2.315%	-1.11%	A
10	11	↑	JavaScript	1.557%	-0.84%	A
11	23	↑↑↑↑↑↑↑↑↑↑↑	Lua	1.362%	+0.83%	A
12	12	=	Ruby	1.329%	-0.65%	A
13	10	↓↓↓	Delphi/Object Pascal	1.076%	-1.35%	A
14	16	↑↑	Lisp	0.905%	+0.28%	A
15	22	↑↑↑↑↑↑↑	Transact-SQL	0.823%	+0.27%	A-
16	28	↑↑↑↑↑↑↑↑↑↑↑	Ada	0.699%	+0.30%	B
17	19	↑↑	RPG (OS/400)	0.660%	+0.05%	B
18	17	↓	Pascal	0.659%	+0.04%	A--
19	46	↑↑↑↑↑↑↑↑↑↑↑	F#	0.604%	+0.37%	B
20	-	=	Assembly*	0.599%	-	B

Programming Community Index

Top-10 Programming Languages

Models of Programming Languages

Procedural Programming:

- Also known as imperative languages.
- Based upon the concept of the procedure call-functions.
- A list or set of instructions telling a computer what to do step-by-step, and how to perform from the first code to the second code.
- Primary feature is a sequence of statements that represent commands.
- Use of *assignment* statements to change values of variables (data) in memory locations.

Edvac-von Neumann.
Retrieved November 11, 2016 from Wikimedia Commons
https://en.m.wikipedia.org/wiki/File:Edvac-vonNeumann.jpg.

C Language

- Developed by Kernihan and Ritchie for implementing the UNIX operating system
- Objective:
  - Provides language that has access to low-level data
  - Generates efficient code
- Programs often expressed with compact code at the expense of readability

```
char (* (*x() ) [] ) ()
```

x: function returning pointer to array[] of pointers to function returning char

1) Brian Kernighan.
Retrieved November 11, 2016 from Wikimedia Commons https://upload.wikimedia.org/wikipedia/commons/a/ae/Brian_Kernighan_in_2012_at_Bell_Labs_1.jpg.

2) Dennis Ritchie.
Retrieved November 11, 2016 from Wikimedia Commons https://upload.wikimedia.org/wikipedia/commons/6/6b/Dennis_ritchie_linux.png.

C Language

- Popularity grew in conjunction with UNIX's acceptance as an operating system (used to port UNIX).

- Excellent for construction of portable systems programs:
  - Portability is enhanced by the ANSI standard for C, the set of rules for C compilers.

Object-Oriented Programming

- Based on the notion of object, which can be described as a collection of memory locations together with all operations that can change values of these memory locations.
- Computation is represented as the interaction among, or communication between objects.

- Fundamental entity is *object* which receives and sends *messages*, performs computations, and has a local state that it can modify.
  - Solve problems by objects sending messages to one another.

Simula, Smalltalk, C++, Java, C#, etc.

C++ Language

- Developed by Bjarne Stroustrup (AT&T)
- Superset of C that includes object-oriented mechanisms without compromising the efficiency and simplicity of C
- Strict type-checking
- Class provides the mechanism for data abstraction and encapsulation

Bjarne Stroustrup

Bjarne Stroustrup
Retrieved November 11, 2016 from Wikimedia Commons
https://upload.wikimedia.org/wikipedia/commons/5/5e/Stroustrup_kent_state.jpg.

C++ Language

- Not pure OOP:
  - C++ does not force you to use its OOP features
  - Can create code that uses only C++'s non-OOP features

- C++: A black Firebird, the all macho car comes with optional seatbelt (lint).

Java

- Developed by SUN for embedded consumer goods; now popular for Web.
- Based on C/C++ but smaller, more reliable, more portable, and has restrictions like:
  - no pointers;
  - no multiple-inheritance; and
  - speed (interpreted byte-code) - JIT compilers.
- Java: All-terrain very slow vehicle.
- “Compile Once, Run Anywhere”.

1) James Gosling
Retrieved November 11, 2016 from Wikimedia Commons
https://commons.wikimedia.org/wiki/File:James_Gosling_2008.jpg

2) Fondo Java
Retrieved November 11, 2016 from Wikimedia Commons
https://commons.wikimedia.org/wiki/File:Fondo_Java.png

Java Translation and Execution

Java Virtual Machine (JVM)
Java Runtime Environment (JRE)

C#

- Developed by Microsoft, led by Anders Hejlsberg (creator of Turbo Pascal, Delphi)
- Based on C++, but many similar features of Java
  - No pointers
  - No multiple-inheritance
  - No globals
  - Strictly OO
  - Automatic garbage collection
  - Multi-threading
  - Cross-platform

Anders Hejlsberg.
Retrieved November 11, 2016 from Wikimedia Commons
https://commons.wikimedia.org/wiki/File:Anders_Hejlsberg.jpg

C

```c
#include <stdio.h>

int  x, y, z ;        // global variable

void function1(...)
{    x = y + 1 ;
....}

int function2(...)
{....  function1(..) ; }

int function3(...)
{....function2(..) ;
     y = z * 2;    }

int main(void)
{
....
     function3(..);
.....
}
```

Procedural Programming

Java

```java
import java.util.*;
class ClassA{          // in ClassA.java
      int x, y = 0 ;    // attributes
      public int methodA(...)
      { ..... }
.....
}
```

```java
.....
class ClassB{          // in ClassB.java
      public int methodB(...)
      { ..... }
.....
}
```

Object-Oriented Programming

```java
class Driver{ // in Driver.java
      public static void main(...)
      {
         ClassA  a = new ClassA(1, 2);
         ClassB  b = new ClassB();
         a.methodA(..);
           b.methodB(..);
      }
}
```

Function 1
Function 2
Procedural Programming
Global Data
Function 3
Function 4

Object A
Data
Method1
Method2
Method3
Method4
Object-Oriented Programming
Data
Method1
Method 2
Object B

C vs. Java: Simple Comparison

```c
#include <stdio.h>



int getFact(int n) {
  int c, fact = 1;
  if ( n < 0 )
        printf("Number should be non-negative.");
  else
        for (c = 1; c <= n; c++)
             fact = fact * c;
   return fact;
}

int main()
{
  int n = 1;

  printf("Enter a number to calculate it's factorial\n");
  scanf("%d", &n);

  printf("Factorial of %d = %d\n", n, getFact(n));

  return 0;
}
// save as <anyname>.c
```

```java
import java.util.Scanner;

class Factorial   // class name
{
  public static int getFact(int n) {
     int c, fact = 1;
     if ( n < 0 )
       System.out.println("Number should be non-negative.");
     else
          for ( c = 1 ; c <= n ; c++ )
          fact = fact*c;
     return fact ;
  }

  public static void main(String args[])
  {
     int n = 1;

     System.out.println("Enter an integer to calculate it's factorial");
     Scanner in = new Scanner(System.in);
     n = in.nextInt();
     System.out.println("Factorial of "+n+" is = “ + getFact(n));



  }
}
//  must save as <class name>.java => Factorial.java
```

Additional Reading

- Procedural vs. Object-Oriented
  - Text book: Page 7-12
- Java Programming
  - Lecture Slides: Blackboard/ NTU Learn -> Content
  - Recorded Lectures: Blackboard/ NTU Learn
- C vs. Java
- Java Quick Guide
  - Blackboard/ NTU Learn -> Content -> Course -> Additional Resources

Topic 2: Classes and Objects

What is an Object?

- Objects are everywhere.

- Characteristics of objects:
  - Identity
  - State
  - Behaviour

One Example

Characteristics?

Person

Height

Weight

Identity: Barack Obama

Attributes/ States

height: 187cm
weight: 81.6 kg
gender: M
age: 49
wealth:
.....

Behaviours

Eat

Study/ Sleep

Person is a *Class*
Obama is an Object (instance) of Person Class

Another Example

Attributes

- Identity: Machine serial number
- State/ Properties: Color, maximum speed, etc.
- Behavior/ Methods: Turn, brake and accelerate

Four Basic Components of Object-Oriented Model

- **Objects:** An entity that contains both the *attributes* that describe the state of a real-world object and the *actions* that are associated with the real-world objects.
- **Messages:** Requests from one object to another for the receiving object to produce some desired result.
- **Methods:** Descriptions of operations/ actions that an object performs when it receives a message.
- **Classes:** A template for objects which consists of methods and state descriptions that objects belong to.

Four Main Concepts/ Features of Object-Oriented Model

- Abstraction
- Encapsulation/ Information Hiding
- Inheritance
- Polymorphism

Abstraction

Definition:

"An abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer."

Grady Booch, *Object-Oriented Analysis and Design with Applications*, 1994

Abstraction

- **Object:** An entity that contains both the *attribute* that describes the state of a real-world object and the *actions* that are associated with the real-world object.

- **State** of an object: *"encompasses all of the (usually static) properties of the object plus the current (usually dynamic) values of each of these properties"* {Grady Booch, *Object-Oriented Analysis and Design with Applications*, 1994}.

- **Attributes:** The data or variables that characterise the state of an object.

Abstraction

- **Behaviour** of an object:
  - *Methods* describe the behaviour associated with an object. They represent the:
    - Services that the object provides to other objects.
    - Operation that the object may perform upon other objects.
    - Interfaces provided by the object.

Abstraction Barrier

Object	**Aston Martin Car**	
Attributes	**State**	**Behaviour**
Color	Black	Steer
NoOfDoors	2	ChangeGear
….	….	….

Encapsulation and Information Hiding

- **Encapsulation** builds a barrier to protect an object's private data. Access to private data can be done through the public methods of the object's class (e.g. get and set methods).

- **Information hiding** hides the details or implementation of the class from users.

- With encapsulation, the users of a class only need to know **what** a class does and **how** to call the methods to perform the tasks.

- The users **do not** need to know the **implementation** details for the methods.

Encapsulation and Information Hiding

My name is Shadow, I am 48 years old and weigh 70 kg.

Encapsulation and Information Hiding

Actual value of internal state is determined by methods that the object performs in response to ***messages*** received.

Encapsulation and Information Hiding

They help to:

- Protect what you have
- Protect your data

Inheritance

- A mechanism that defines a new class which inherits the properties and behaviours (methods) of a parent class.
- **Superclass** or **base class (parent)**:
  - The class from which another class inherits structures and/or behaviours
- **Subclass** or **derived class (child)**:
  - A class that inherits from one or more other classes
  - Any inherited behaviour may be redefined in subclass, *superseding* inherited definition
- Can create new classes without extensive duplication of code
- Reuse parent's code

Inheritance Example

Multiple Inheritance

- A class inherits from more than one superclass
- Problem that arises is which property/ method to inherit if property/ method is present in more than one superclass
- User specified precedence (Smalltalk)
- Rename one property/ method
- Accept both if signatures are different

```
print(String s)
print(Char c)
```

Multiple Inheritance

Polymorphism

- Greek word that literally mean *many shapes* or *multiple forms*

- Same message can be sent to different objects**
  - Each object performs operation appropriate to its class
  - Do similar things differently

- Sending object does not need to know the class of receiving object or how the object will respond

- Most powerful concept of OO

Polymorphism

```
void  call (Animal  animal)
{
    animal.move();
}
```

Summary

Key points from this chapter:

- Four basic components:
  - Objects
  - Messages
  - Methods
  - Classes
- Four main properties (concepts):
  - Abstraction
  - Encapsulation/ Information Hiding
  - Inheritance
  - Polymorphism

Class **2GPhone**

Behaviour
Ring
....

Class **MobilePhone**

Attributes	Behaviour	Behaviour
Color	Connect	Ring
RingTone	Encrypt	Answer
ScreenSize	.....	HangUp
		SendSMS

Class **3GPhone**

Attributes	Behaviour
DataRate	OnWiFi
	SendMMS
	Ring
	....

OO Conceptual Terms

- Object
- Class
- Attribute
- Operation
- Interface
- Implementation
- Association
- Aggregation
- Composition
- Generalisation
- Super-Class
- Sub-Class
- Abstract Class
- Concrete Class
- Polymorphism

Additional Reading

- Class and Object
  - Text Book: Page 12-20
- Four Main Concepts of OO
  - Text Book: Page 20-33