OBJECT ORIENTED PROGRAMMING

Introduction to OOP

by
Dr. Nor Saradatul Akmar Zulkifli
Faculty of Computer Systems & Software Engineering
saradatulakmar@ump.edu.my

Content Overview

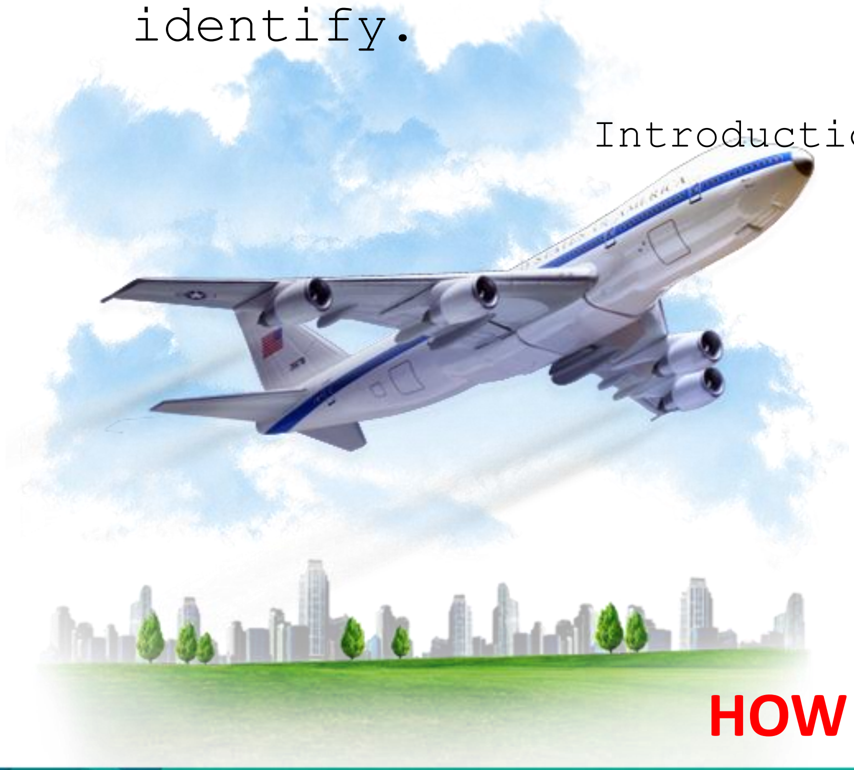
- Motivation of OOP
- History – Why OOP?
- Difference between procedural and OO programming
- Introduction to Object & Class
- Four design principles of OOP

Learning Objective

- To explain the basic of Object Oriented Programming model
- To understand the differences between procedural and OO Programming
- To understand the differences between class & object (instances)
- To identify class & object based on the problem requirement(s)
- To understand four design principles of OOP

Imagine you are given the task of designing an airline reservation system that keeps track of flights for a commuter airline. List the classes you think would be necessary for designing such a system. Describe the data values and methods you would associate with each class you identify.

C.Thomas Wu
Introduction to Object-Oriented Programming
Chapter 1: pg.28



HOW WILL YOU SOLVE THIS PROBLEM?

To solve the problem

- ❑ Breaking it down into manageable pieces
- ❑ Design separate pieces that are responsible for certain parts of the solution.

2 popular programming design methods:

1. Procedural Programming
2. Object-Oriented Programming

PROCEDURAL VS. OBJECT-ORIENTED

What Are The Differences??

PROCEDURAL

- Emphasis of program is on procedure – on HOW to accomplish a task.
- Top-down design; Step-wise refinement
- Examples: C, COBOL, Fortran, LISP, Perl, HTML, VBScript

OBJECT-ORIENTED

- Emphasis of program is on object – understanding the objects involved in a problem and how they interact
- Bottom-Up design; Reusable libraries
- Example: C++, Visual Basic.NET and JAVA

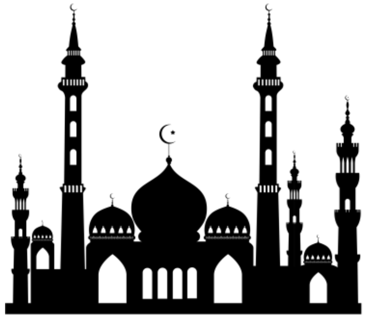
What Are The Differences??

INTRODUCTION TO OBJECT & CLASS

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

Class
Definition of objects that share structure, properties and behaviors

Skyscrapers
Class



Mosques
Class

Berries
Class

Instance
Concrete Object, Created from a certain class

KL Tower
Instance of ***Skyscraper***

Beth Mosque
Instance of ***Mosque***

Strawberry
Instance of ***Berry***

WHAT IS AN OBJECT?

- ❑ Concept : View the world as OBJECT.
- ❑ A real world object is a thing, both tangible (television) and intangible (bank account)
- ❑ Objects are described by using two features:
  1. Attributes
  2. Behaviors

ATTRIBUTE

- Describe Characteristic of the object
- Ex: Color, NumberofWheel, TopSpeed

BEHAVIORS

- An action that an object is capable of performing
- Ex: goForward, TurnLeft, Stop

OBJECT : EXAMPLE

STATE

Color	: Red
NumberOfDoors	: 4
TopSpeed	: 240kmph
Model	: Proton

BEHAVIORS

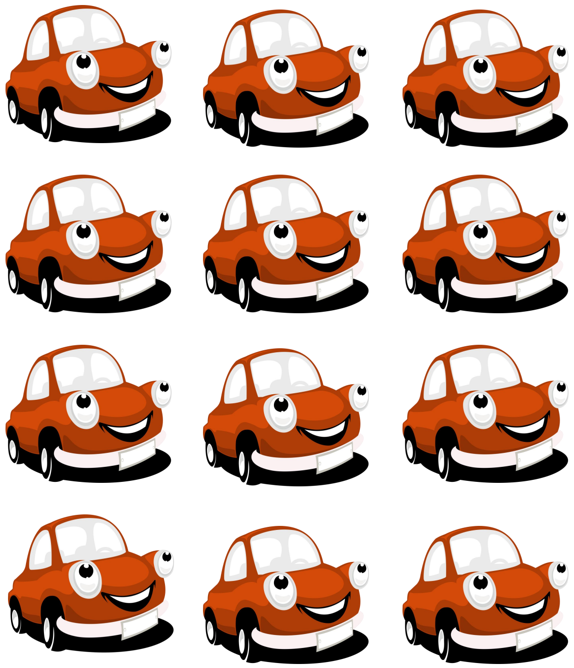
goForward
TurnLeft
Stop
TurnRight

TASK:

Working in pair, think of One Object and list out the State and It's Behaviors

WHAT IS A CLASS?

- ❑ A class is a template that defines attributes and methods of a real-world object
- ❑ It is the blueprint of an object
- ❑ It contains the codes that specifies the state and behaviors of a particular type of object
- ❑ Each object that is created from a class is called an instance of the class
- ❑ Multiple objects can be created from the same class during program execution & stored in the main memory



CLASS : Car

4 Design Principles

1. Abstraction

2. Encapsulation

3. Inheritance

4. Polymorphism

1. ABSTRACTION

"A multimillionaire woman told her son that he would inherit the family fortune if he promised to continue working after she passed on. The woman dies. The money starts pouring in, and somehow the son forgets about his promise. However, a clause in her will forces him back to work."

Jim Keogh & Mario Giannini
OOP Demystified- A self teaching guide
Pg.87

A way a programmer of a super class (mom) forces a programmer of a subclass (son) to define a behavior.

- The concept of exposing only the required essential characteristics and behavior with respect to a context
- And hides its complexity

ADVANTAGE

✓ Every user will get his own view of the data according to his requirements and will not get confused with unnecessary data

EXAMPLE:

```
public abstract class Bank
{
private int accno;
private String name;
void display_to_clerk ( )
{
System.out.println (“Account No. = “ + accno);
System.out.println (“Name = “ + name);
}
void display_to_manager ( )
{
System.out.println (“Account No. = ” + accno);
System.out.println (“Name = “ + name) ;
System.out.println (“Loan = “ + loan);
System.out.println (“Balance = “ + balance);
} }
```

2. ENCAPSULATION

- The condition of being **enclosed**.
- Enclosing "attributes" and "methods" **within a class**

- This feature keeps the data safe from outside interference and misuse.
- Led to a concept of **data hiding**

- Access to attributes and methods of a class is control by using **Access Modifiers** within a class definition

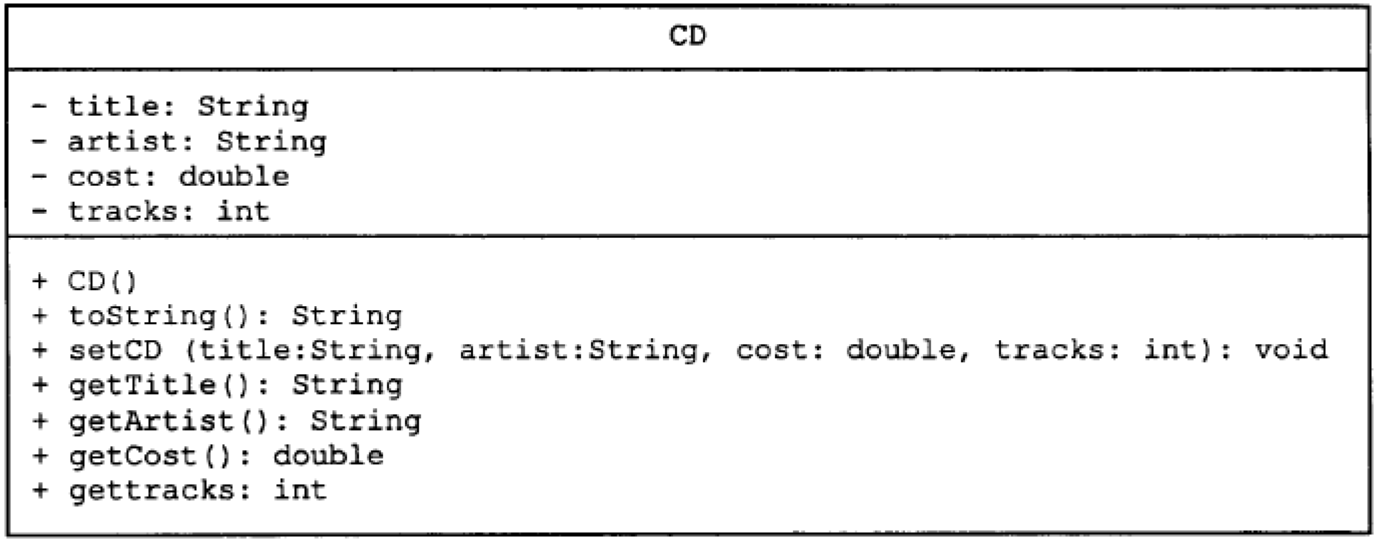
- **Access Modifiers** is a keyword that tells what part of the program can access attributes and methods that are members of a class.

- `Private` access assigned to the data – Keep data inaccessible to outside class

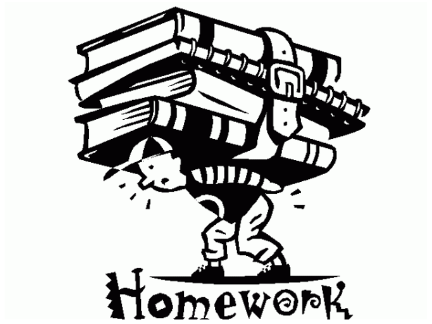
- `Public` access – To make methods/data accessible to outside classes

Accessibility Modifier

- UML Class Diagram :
  - Minus sign (-) used for items that are `private`
  - Plus sign (+) used for items that are `public`



```
CD
- title: String
- artist: String
- cost: double
- tracks: int
+ CD()
+ toString(): String
+ setCD (title:String, artist:String, cost: double, tracks: int): void
+ getTitle(): String
+ getArtist(): String
+ getCost(): double
+ gettracks: int
```



ABSTRACTION VS. ENCAPSULATION

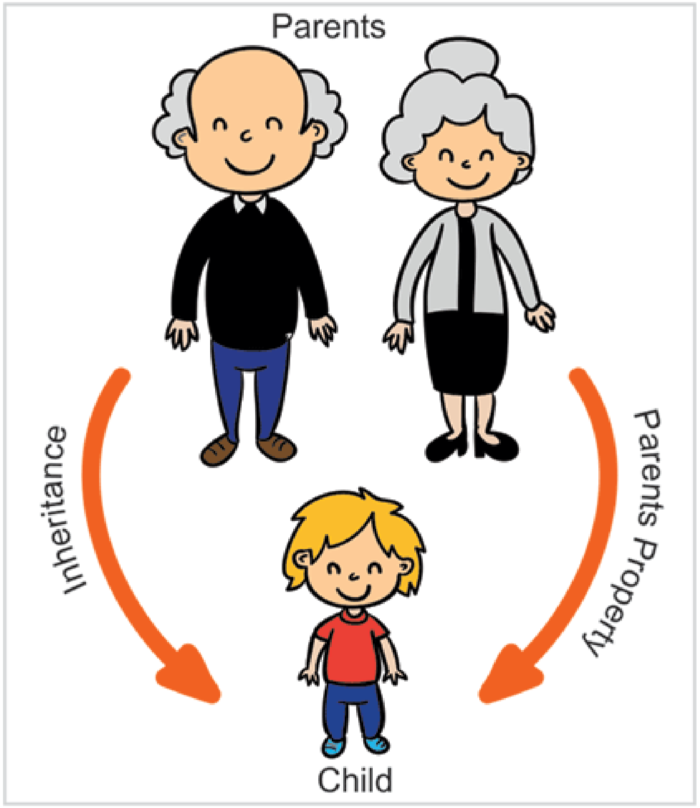
What Are The Differences??

3. INHERITANCE

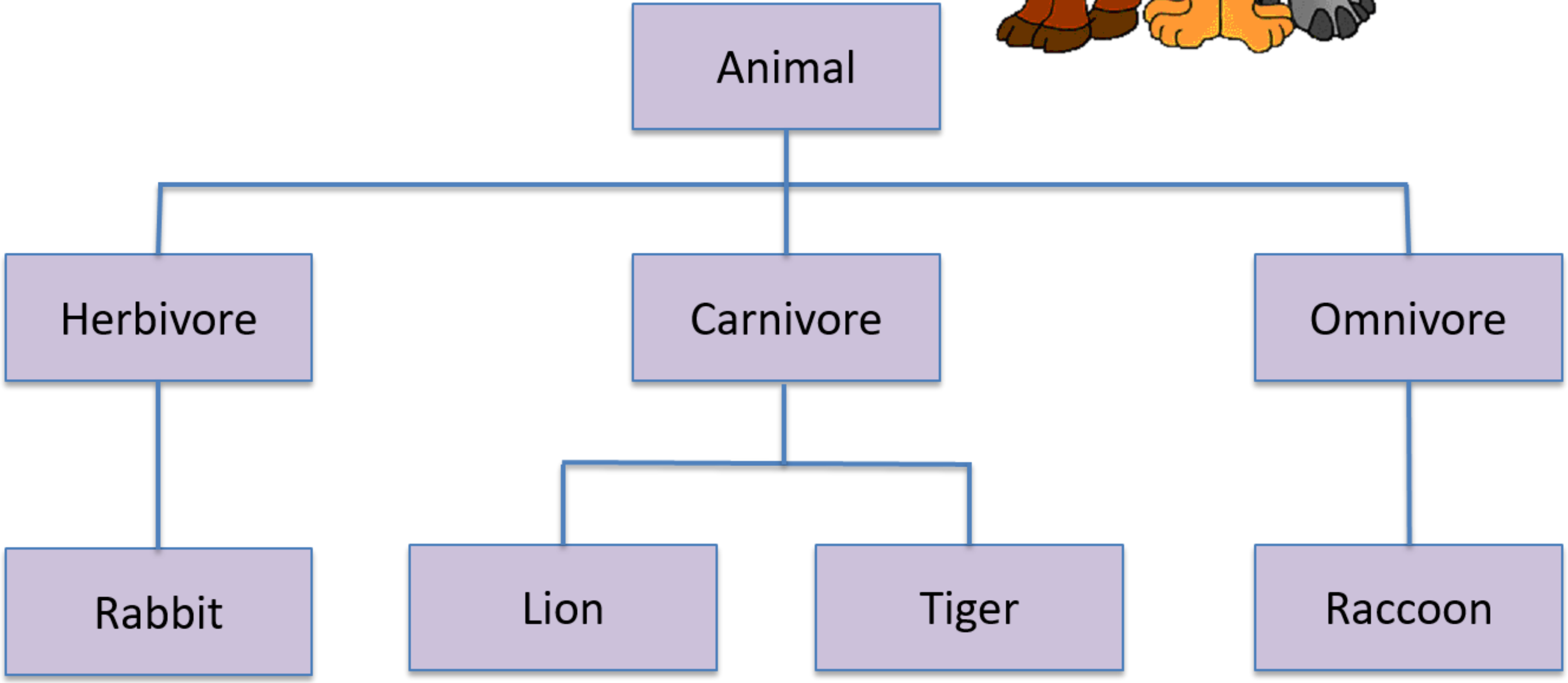
- It enables objects to inherit attributes and behaviors from other objects
- Classes with properties in common can be grouped so that their common properties are only defined once
- **Superclass Vs. Subclass**

ADVANTAGE

- Reduce the amount of new code that must be designed, written and tested each time a new program is develop



INHERITANCE : EXAMPLE



INHERITANCE : EXAMPLE

Parent Class (Superclass)

Feature 1

Feature 2

Feature of Parent Class

Child Class (Subclass) – inherited from Parent Class

Feature 1

Feature 2

Feature of Parent Class accessible to child class because of inheritance

Feature 3

Feature defined in Child Class

4. POLYMORPHISM

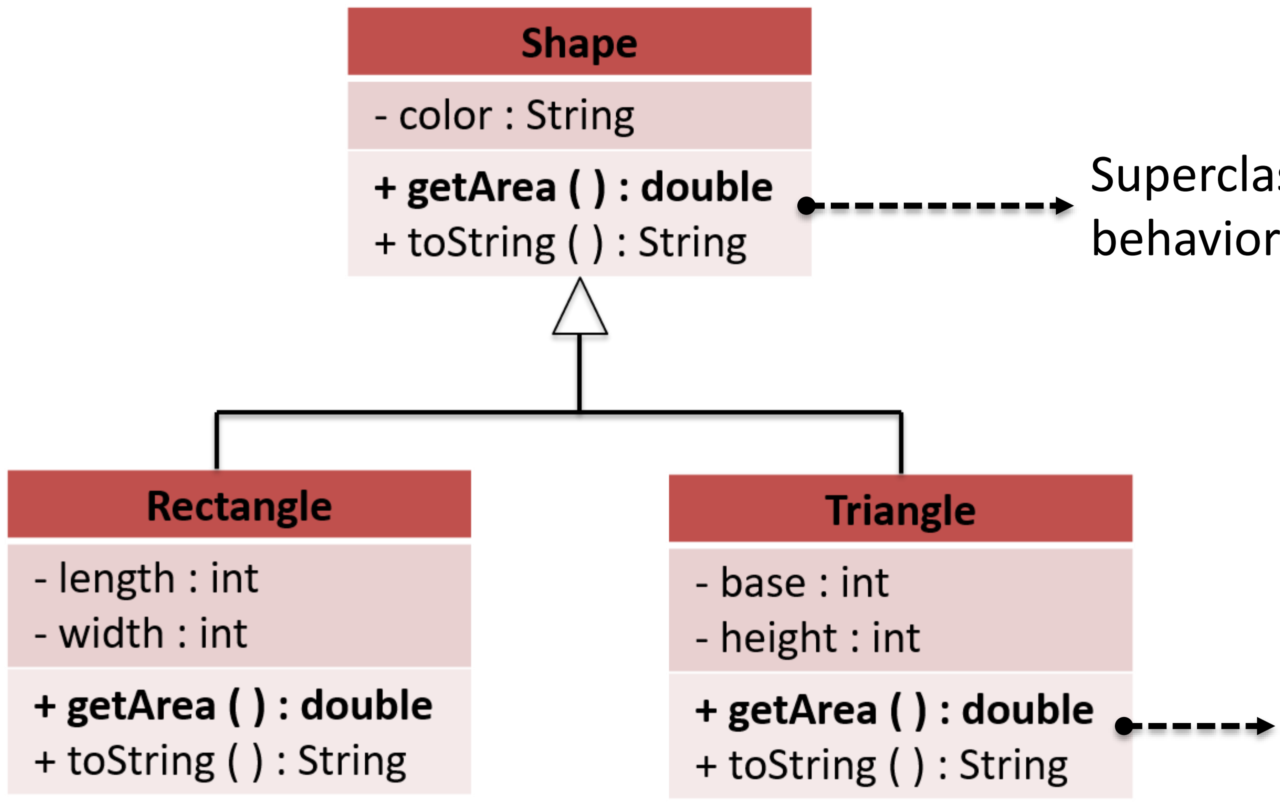
- One **thing** has the ability to appear in many **shapes**.

- Programming terms:
  - **"thing"** – The name of a method
  - **"shape"** – The behavior performed by the method

- The name of a method can have many behaviors.

- Allows programmer to create methods with the SAME name BUT with DIFFERENT implementation in DIFFERENT classes (that are related through inheritance)

POLYMORPHISM : EXAMPLE



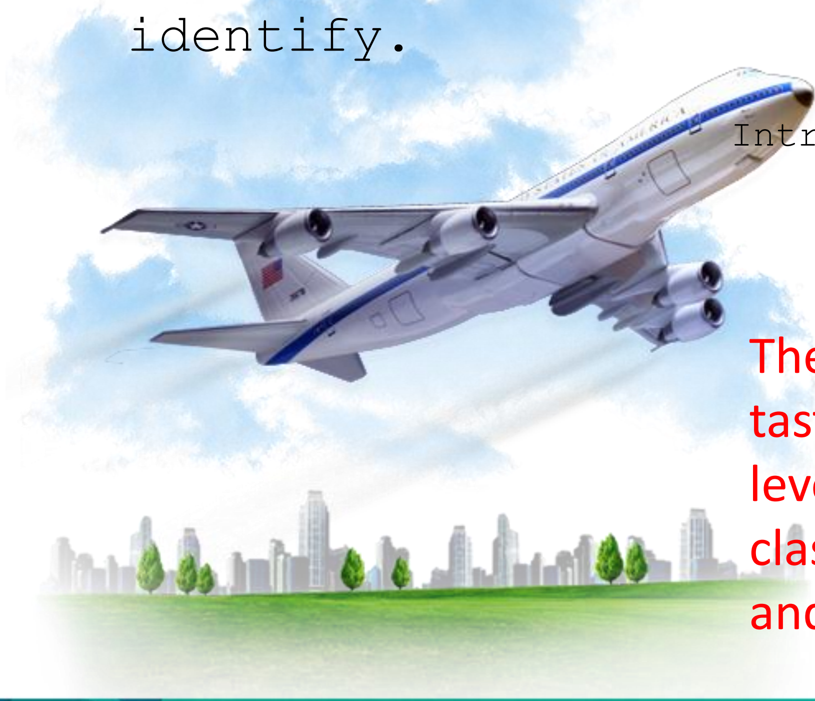
Superclass defines the expected behaviors of all subclasses

Subclasses provide the actual implementations

EXERCISE

"Imagine you are given the task of designing an airline reservation system that keeps track of flights for a commuter airline. List the classes you think would be necessary for designing such a system. Describe the data values and methods you would associate with each class you identify."

C.Thomas Wu
Introduction to Object-Oriented Programming



The OBJECTIVE of this Exercise is to give you a taste of thinking about a program at a very high level. Try to identify about a half dozen or so classes, for each class, describe several methods and data members

Author Information

Dr. Nor Saradatul Akmar Binti Zulkifli

Senior Lecturer
Faculty of Computer Systems & Software Engineering
Universiti Malaysia Pahang