# Lab 1
# Simple Object Oriented

**Learning outcome:**

At the end of this lab, student should be able:

1. be able to analyze the problem and construct a simple object-oriented program.
2. be able to design a UML diagram.

**Dateline:** Week 2 – Before ends of week 2. Upload to putrablast.

** Copy or other forms of cheating is forbidden. The standard penalty for the first offence is to award 0 to all parties concerned.

---

1. Draw a UML class diagram and create a class named of **Identification.** The class contains:
   - Three private instance variables: name (string), e-mail (string) and gender (char of either 'm' or 'f').
   - One constructor to initialize the name, e-mail and gender with the given values.
   - Methods getters/setters: **getName(), getEmail,** s**etEmail()** and **getGender().**
   - A **toString()** method that returns "name(gender) at email", e.g. "Nor Fazlida (f) at fazlida@upm.edu.my".

   Construct a program named **TestIdentification** that demonstrates that each method works correctly. Try changing the email of the person e.g. "Nor Fazlida (f) at fazlida@gmail.com" Print all the data fields in the **Identification**.

2. Create a class named **Student** that contains the following:
   - idStudent. The idStudent is an int variable that holds the student's matric number.
   - name. The name field references a String object holds the student's name.
   - major. The major field references a String object holds the student's major.
   - classification. The classification field references a String object holds the student's classification level (bongsu, kecil, muda, sulung)
   - Constructor should accept the student's matric number, name, major and classification as arguments. These values should be assigned to the object's idStudent, name, major, and classification fields.
   - Constructor should accept the student's matric number, name, major and classification as arguments. These values should be assigned to the object's idStudent, name, and major. The classification fields should be assigned an empty string (" ").
   - A no-arg constructor that assigns empty strings (" ") to the name, major, and classification fields, and 0 in the idStudent field.

Write appropriate mutator/setter methods that store values in these fields and getter/accessor methods that get values stored in these fields. Once you have written the class, write a separate program that creates four element array of Student object to hold the following data:

| Id Student | Name | Major | Classification |
|---|---|---|---|
| 160932 | Ghazali Ali | Science (Math) | Muda |
| 167432 | Abdul Rahman | Engineering | Muda |
| 174123 | Abdul Majid | Forestry | Kecil |
| 158911 | Hoo Yee An | Computer Science | Bongsu |

3. Construct the UML and create a class named **Fan** that contains:
   - Three contants named **SLOW, MEDIUM** and **FAST** with values 1, 2, and 3 to denote the fan speed.
   - A private **int** data field name **speed** that specifies the speed of the fan (default speed is slow)
   - A private **boolean** data field named **on** that specifies whether the fan is on (default is false)
   - A private **double** data field named **radius** that specifies the radius of the fan (default is 5)
   - A **string** data field named **color** that specifies the color of the fan (default is blue)
   - The accessor and mutator methods for all data fields
   - A no arg constructor that creates a default fan
   - A method named **toString()** that returns a string description for the fan.
     - If the fan is on, the method returns the fan speed, color and radius in one combine string.
     - If the fan is not on, the method returns the fan color and radius along with the string “fan is off” in one combined string.

   Write a test program that creates two Fan objects. Assign the first object with maximum speed, radius 10 and color yellow, while the second object assign medium speed, radius 5, and color blue. Display the objects by invoking toString method.