Object Oriented Programming

- **ADT: Abstract Data Type**
- **Encapsulation**
- **Class & Object**
  - Methods (Function Members)
  - Fields/Attributes/Data Members

Traditional Program

```
struct {
    int id ;
    char name[40];
    long salary ;
    float tax ;
} Employee[120];

void main() {
    EditBio();
    Salary();
    Report(Data);
}
```

void EditBio() {
    ...
}

void Report(Data) {
    ...
}

void Salary() {
    ...
}

```
ADT: Abstract Data Type

- **Programmer-defined data type that can be manipulated like system-defined data types**

- **2 parts:** Data and Operations
  - **Attribute/Field**
    - an element of ADT data
  - **Method/Function**
    - an implementation of ADT operation

OOP: Object Oriented Programming

```
Object A

Operation

Operation

Data

message

Object B

operation

operation

operation

operation
```
Encapsulation

- The process shielding an object's data from direct outside access.
- Objects have no direct access to the data in other objects.
- Only the operations that are included in an object directly access the object's own data.
- Data is said to be hidden because access is only possible indirectly.
- In the example, the attribute `accountId` can only be accessed via the operation `getId()`, and attribute `balance` can be accessed via the operation `deposit()` or `withdraw()`.

Examples

- **Bank Account**
  - **Data**
    - เลขที่บัญชี (accountId)
    - ชื่อเจ้าของบัญชี (name)
    - ยอดเงินในบัญชี (balance)
  - **Operation**
    - ฝาก (deposit)
    - ถอน (withdraw)
    - ถามยอด (get balance)
    - ถามเลขที่บัญชี (get account-id)
    - แสดงบัญชี (to string)
Class & Object

- A class is a template or blueprint for creating an object.
- A class defines what types of data are included in the object and specifies the operations the object performs.
- Object is created from a class.
- Creating an object from a class is known as instantiating an object.

Examples

Class

Objects/Instances

Data:
- สี, ประตู, ถังน้ำมัน, เครื่องยนต์, ตрос, พวงมาลัย
- ความจุถังน้ำมัน: 45, 60, 71 ลิตร

Operation:
- เดินหน้า, เลี้ยวซ้าย, หยุด, เปลี่ยนเกียร์
ADT & Object Orientation

**ADT : class**

- Outside the Class
- Inside the class
- Private methods
- Private data
- Public interface methods

---

Methods Declarations

- **void <identifier> () {}**
- **void <identifier> (<args_list>) {}**
- **Type <identifier> () {**
  - ...
  - return <Type> ;
  **}**
- **Type <identifier> (<args_list>) {**
  - ...
  - return <Type> ;
  **}**

```c
flush void flhus() {
  ...
}
```
```c
print void print(int x) {
  ...
}
```
```c
random double random () {
  ...
  return ??? ;
}
```
```c
read int read() {
  ...
  return ??? ;
}
```
### Summary

- **A class is a "pattern" of objects**
- **A class specifies attributes and operation of objects of that class.**
- **Object is created from a class.**
- **Attribute:**
  - an element of object data
- **Method:**
  - an implementation of object behavior
- **Message:**
  - a request to an object to invoke one of its methods

### Object Oriented Programming: Using Java

#### Building a class

```java
class BankAccount {
    private int accountId;
    private double balance;

    public void deposit(int amt) {
        balance = amt;
    }

    public void withdraw(int amt) {
        balance = balance - amt;
    }

    public int getId() {
        return accountId;
    }

    public double getBalance() {
        return balance;
    }

    public String toString() {
        return "Id: " + accountId + ", Balance = " + balance;
    }
}
```
Object Oriented Programming: Using Java

Creating Objects

class TestBankAccount {
    public static void main(String args[]) {
        BankAccount baSomchai = new BankAccount();
        BankAccount baSomsri = new BankAccount();
        baSomsri.deposit(500);
        //baSomchai.balance = 189000;
        System.out.println(baSomsri.toString());
    }
}

Methods Overloading

Multiple methods declare in the same class with the same name and difference number of type of arguments. For example:

class OverloadDemo {
    void print(int x) …
    void print(double x) …
    void print(String x) …
    void setStudent(int id) …
    void setStudent(int id, String name) …
}
Constructor

- The method name must exactly match the class name.
- There must not be a return type declared for the method.

```java
class BankAccount {
    private int accountId;
    private double balance;

    public BankAccount() {}  

    public BankAccount(int id){
        accountId = id
    }
    public BankAccount(int id, double bl){
        accountId = id
        balance = bl;
    }
}
```

Example:
```
BankAccount ba1 = new BankAccount();
BankAccount ba2 = new BankAccount(214578);
BankAccount ba3 = new BankAccount(214578, 450);
```