OOP brief introduction

LT13a

Procedural Programming

- Related programming statements are group into subroutines (e.g. functions)
- Related data items are grouped into record data structure

  e.g. student info can be stored in a tuple as a record structure: (name, class, gender)

Concept of OOP

- OOP goes one step further to group together the record data structure and the subroutines that operate on the data items in this data structure

Definitions: Class and Object

- A class is a grouping of data and methods within an entity.
- It is a blueprint or template to create objects which are runtime instantiation of the class.

Examples: Class and Object

- An animal class is a blueprint which animals can be instantiated from.
- It consists of properties/attributes (data) such as name of the animal, type of the animal, number of legs etc. It also consists of a set of methods/behaviors such as ability to eat, move, reproduce etc.
- Each animal object is an entity.
- E.g. Lion is an animal object

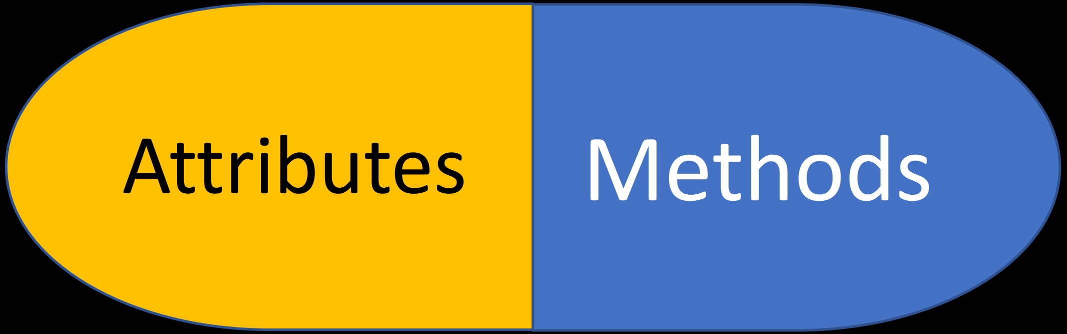
that is a mammal with 4 legs.

It can perform actions such as eat,

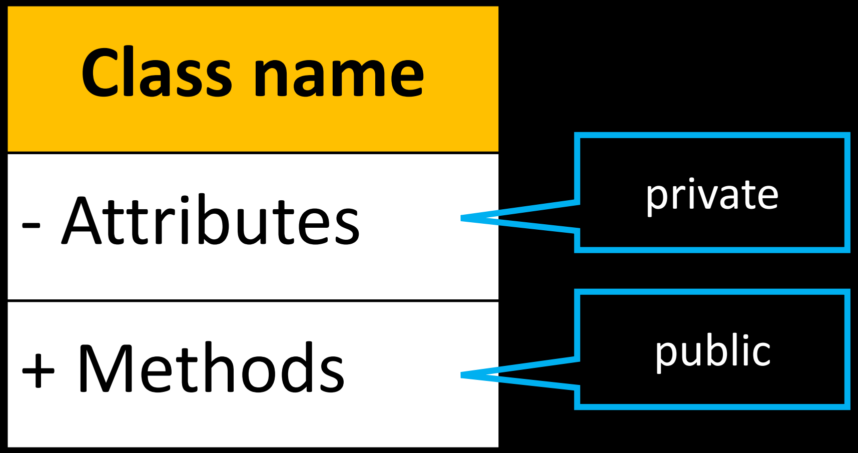
move and reproduce.

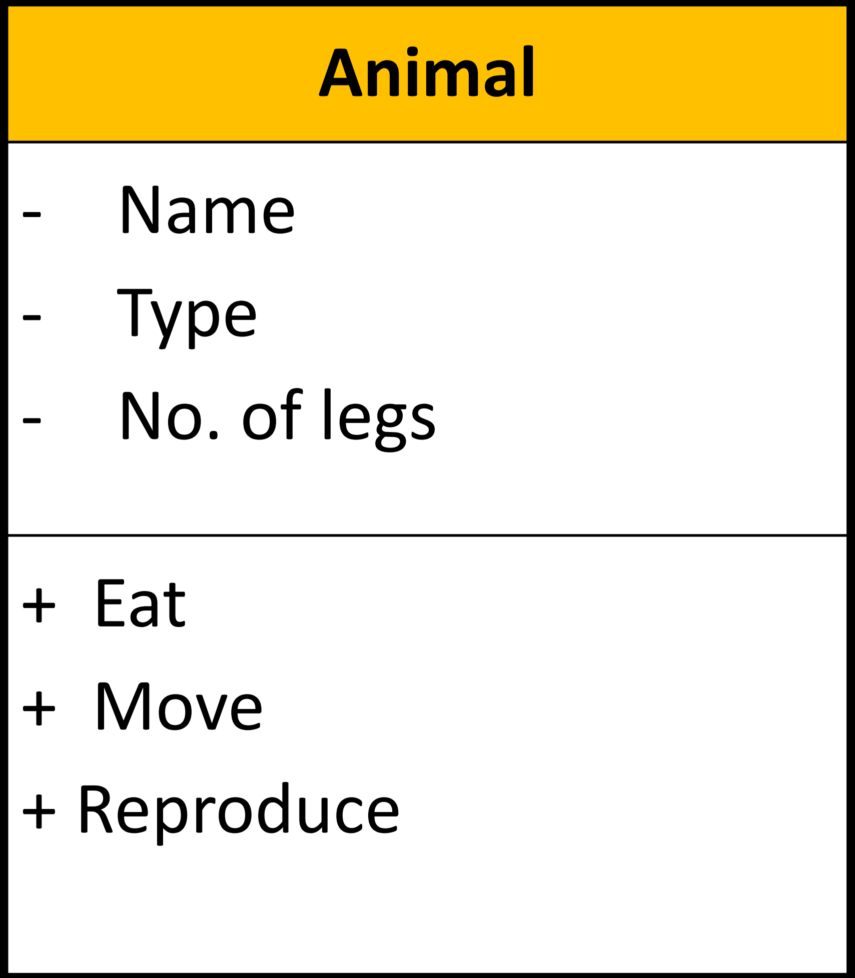
Definition: Encapsulation

- Encapsulation is the act of bundling private data and public methods within a class. (i.e. integrated into a single entity)
- This is different from the procedural ADT way where data and functions are kept separate.
- Private data are only accessible via public methods.



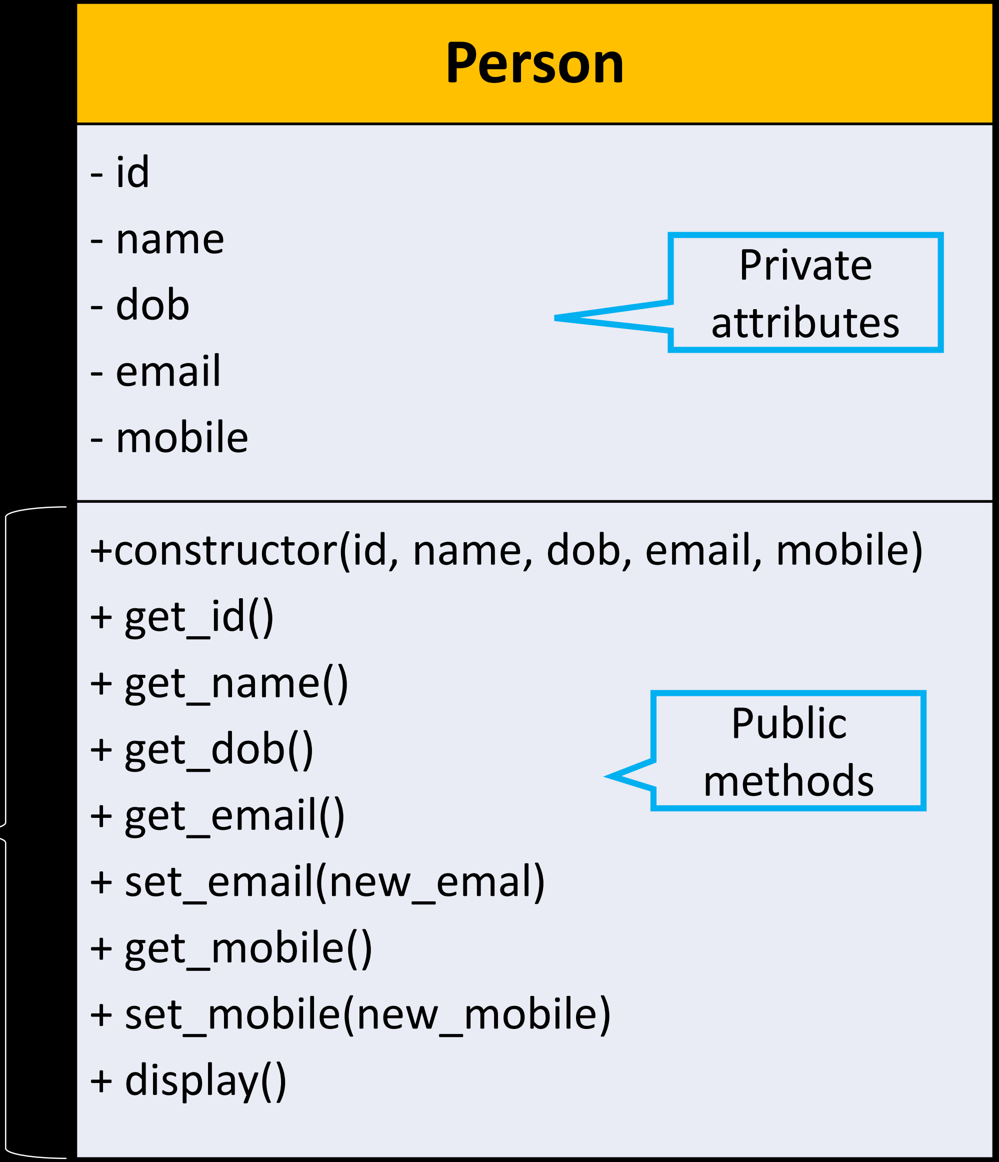
UML Class Diagram





Class Diagram for Person class

Methods:
→Constructor
→Accessors/getters
→Modifiers/setters
→Utilities



Person

- id
- name
- dob
- email
- mobile

+constructor(id, name, dob, email, mobile)
+ get_id()
+ get_name()
+ get_dob()
+ get_email()
+ set_email(new_emal)
+ get_mobile()
+ set_mobile(new_mobile)
+ display()

Initialiser or Constructor Method

A constructor creates an object and allocates storage for the data (attributes).

Two underscores characters before and after init

self is the first parameter in every method

```
class Person:
      def __init__(self, id, n, d, e, m):
         self.pid = id
         self.name = n
         self.dob = d
         self.email = e
         self.mobile = m
```

Accessors

To access the data (attributes), we provide public get methods called accessors.

```
def get_pid(self):
    return self.pid

def get_name(self):
    return self.name

def get_dob(self):
    return self.dob

def get_email(self):
    return self.email

def get_mobile(self):
    return self.mobile
```

Modifiers/Mutators

To modify the data (attributes), we provide public setter methods called modifiers.

```
def set_email(self, new_email):
    self.email = new_email


def set_mobile(self, new_mobile):
    self.mobile = new_mobile
```

Utility

```
def display(self):
    print("ID:", self.pid)
    print("Name:", self.name)
    print("DOB:", self.dob)
    print("Email:", self.email)
    print("Mobile:", self.mobile)
```

Person Class (in code)

```
class Person:
    def __init__(self, id, n, d, e, m):
        self.pid = id
        self.name = n
        self.dob = d
        self.email = e
        self.mobile = m
    def get_pid(self):
        return self.pid
    def get_name(self):
        return self.name
    def get_dob(self):
        return self.dob
    def get_email(self):
        return self.email
    def get_mobile(self):
        return self.mobile
    def set_email(self, new_email):
        self.email = new_email
    def set_mobile(self, new_mobile):
        self.mobile = new_mobile
    def display(self):
        print("ID:", self.pid)
        print("Name:", self.name)
        print("DOB:", self.dob)
        print("Email:", self.email)
        print("Mobile:", self.mobile)
```

Create objects and calling its methods

Create Person object
using constructor: Instantiation

```
p1 = Person(1, "Lim Ah Seng", "1995-01-01", 
"limahseng@hotmail.com", "12345678")
p2 = Person(2, "Tan Ah Lian", "1995-12-31", 
"tanahlian@yahoo.com", "87654321")


print(p1.getName())
p1.set_mobile("88888888")
p1.display()
```

Call object's methods
<Person object> . <method>

Output

```
p1 = Person(1, "Lim Ah Seng", "1995-01-01",
"limahseng@hotmail.com", "12345678")
p2 = Person(2, "Tan Ah Lian", "1995-12-31",
"tanahlian@yahoo.com", "87654321")

print(p1.getName())
p1.set_email("johnny@hotmail.com")
p1.display()
```

```
→ "Lim Ah Seng"
```

No output

```
ID: S1234567A
Name: Lim Ah Seng
DOB: 1995-01-01
Email: johnny@hotmail.com
Mobile: 12345678
```

Summary

- A class is a grouping of data and methods within an entity. It is a blueprint or template to create objects which are runtime instantiation of the class.

- In encapsulation, each class ties together private data and public methods within an integrated entity. Private data are only accessible via public methods.