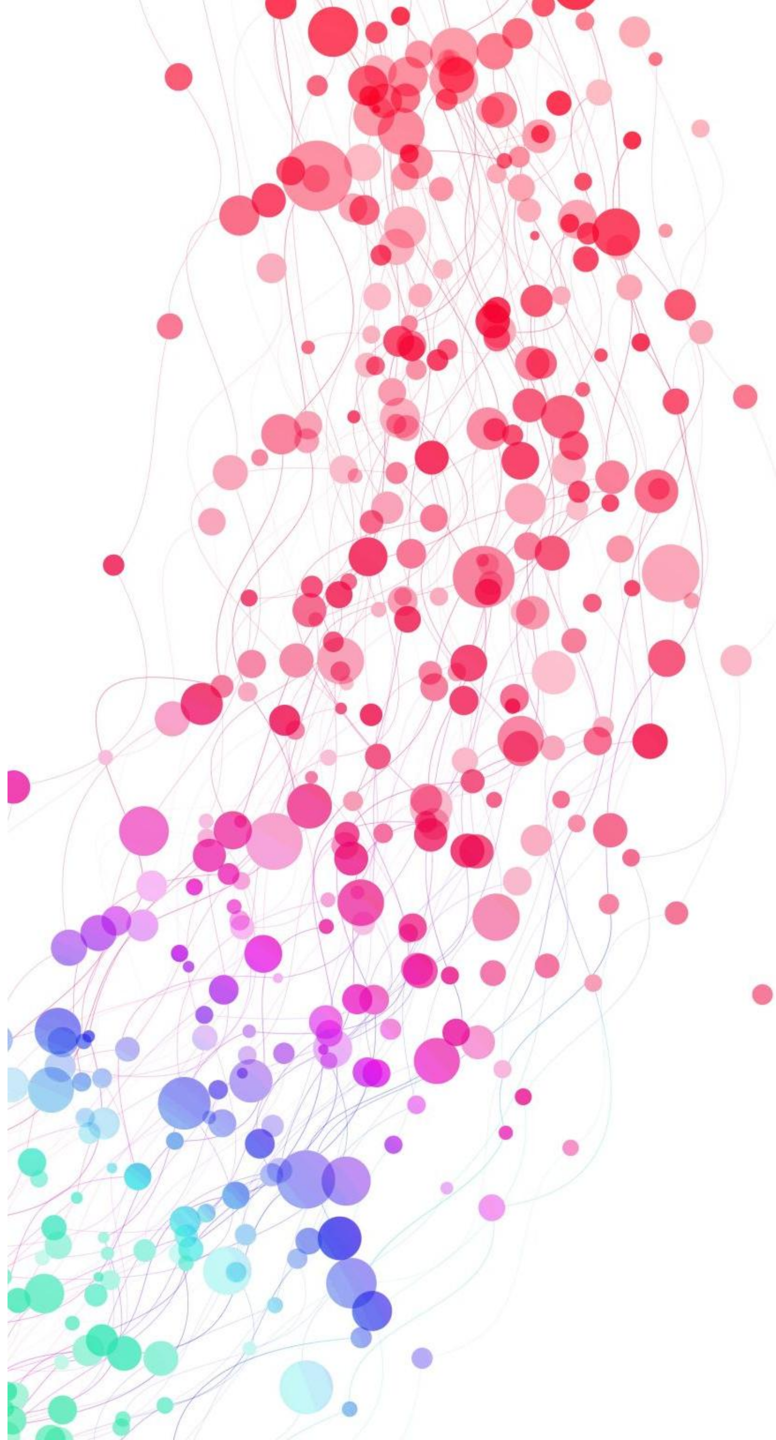
LECTURE 1

Computer programming II



```
#include<iostream>
using namespace std;
int main() {
      Statment1;
      Statment2;
      Statment3;
      Statment4;
      Statment5;
      Statment6;
      Statment7;
      Statment8;
      Statment9;
      Statment10;
      Statment11;
      Statment12;
      Statment13;
      Statment14;
      Statment15;
      Statment16;
}
```

```
#include<iostream>
using namespace std;

void fun1(){
      Statment1;
      Statment2;
      Statment3;
      Statment4;
}
void fun2(){
      Statment5;
      Statment6;
      Statment7;
}
..........
..........
int main() {
      fun1();
      fun2();
}
```

```
#include<iostream>
using namespace std;
class Testclass1 {
      private:
      // data item
      public:
      // function member
};
class Testclass2 {
      private:
      // data item
      public:
      // function member
};
.....
int main() {
      Testclass1 Tc1;
      Testclass2 Tc2;
}
```

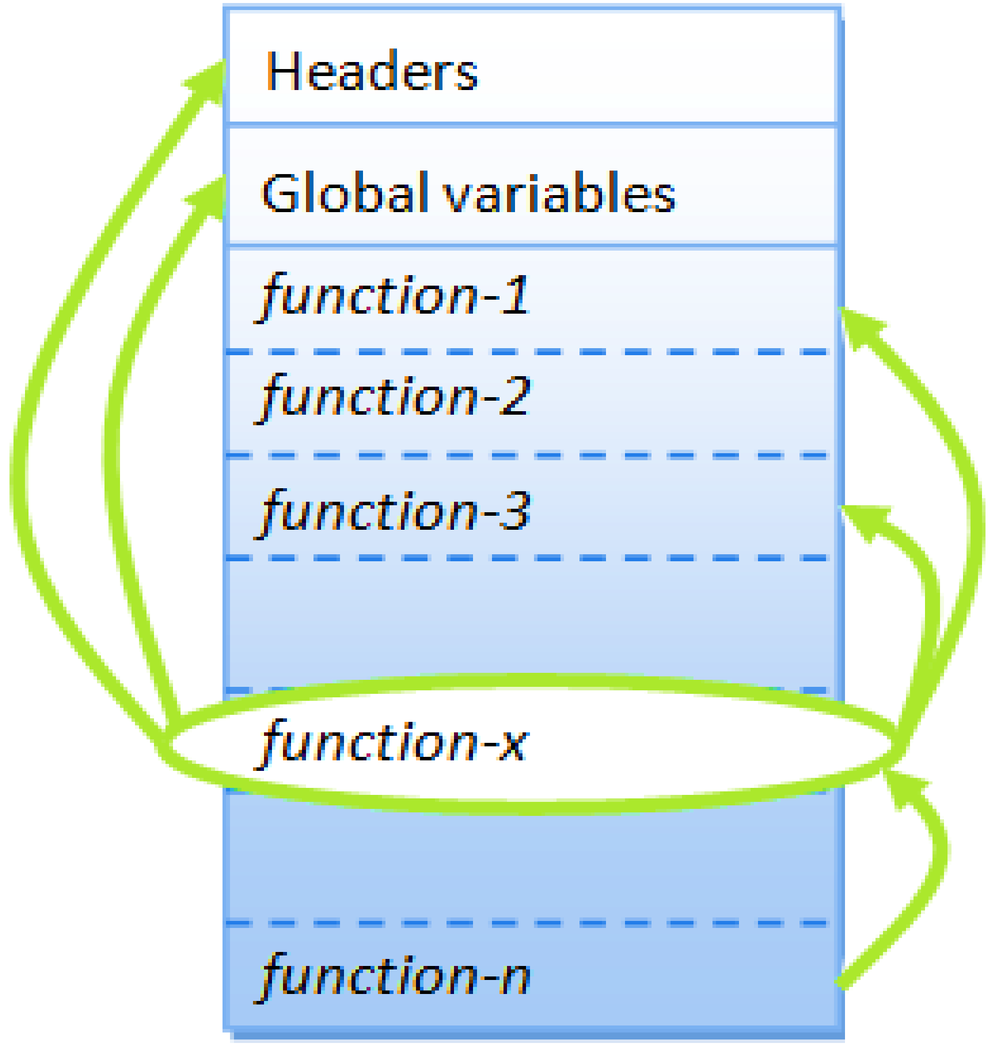
Programming

- There are two common programming methods:

❑ **Procedural programming** is centered on creating procedures or functions.

❑ **Object-Oriented Programming (OOP)** is centered around the object.

Procedural programming



A function (in C) is not well-encapsulated

Procedural programming

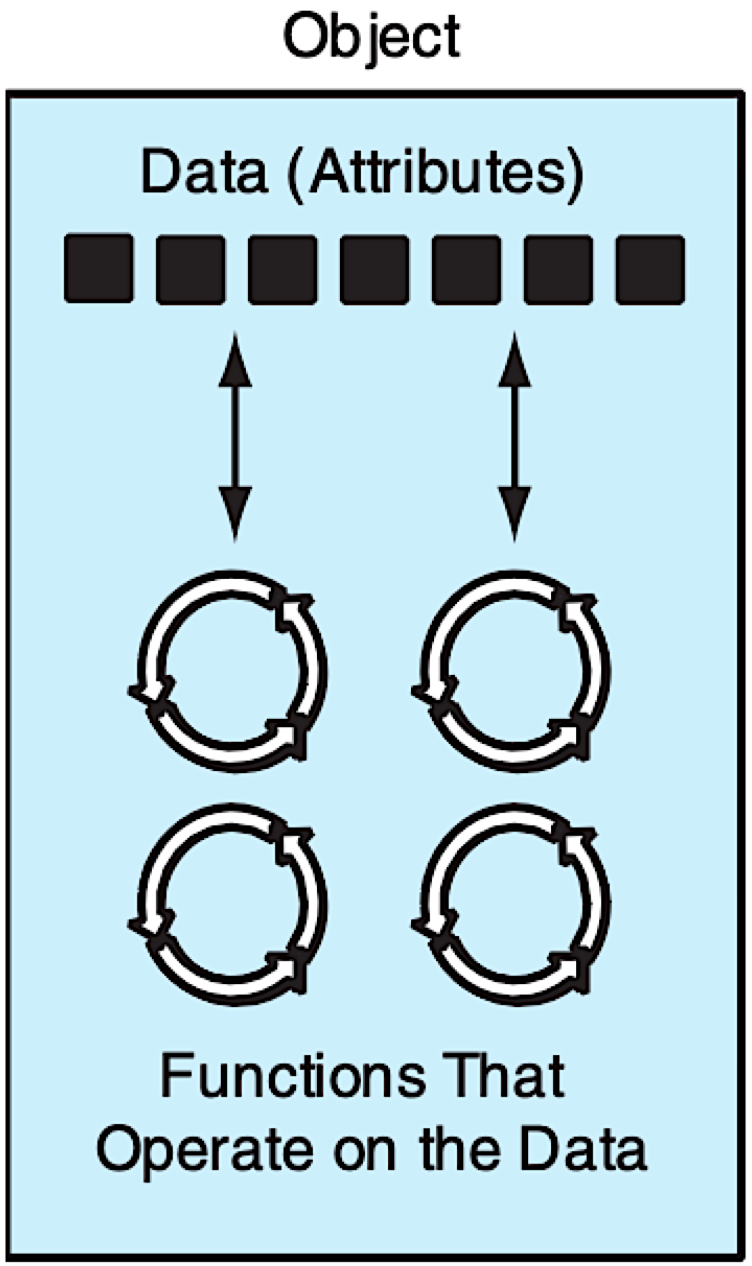
- ❑ As programs become larger and more complex, the separation of a program’s data and the code that operates on the data can lead to problems.
- ❑ When the structure of the data changes, the code that operates on the data must also change to accept the new format.

Procedural programming

- **Smaller programs** - A program in a procedural language is a list of instructions.
- **Larger programs** are divided in to smaller programs known as functions.
- Each **function** has a **clearly defined purpose and a clearly defined interface to the other functions** in the program.
- **Data is Global** and shared by almost all the functions.

Object-oriented programming

- An object is a software entity that contains both data and functions.
- The data that are contained in an object are known as the object's attributes or properties.
- The procedures that an object performs are called member functions.



Reasons for OOP

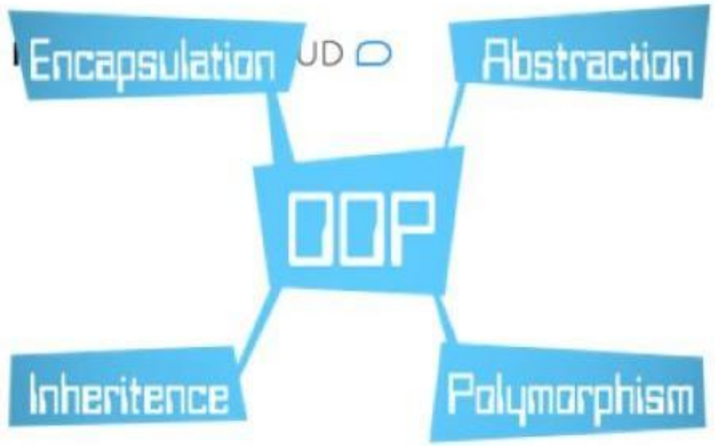
- Simplify programming
- Data hiding:
    - Implementation details hidden within classes themselves
- Object reuse
    - Class objects included as members of other classes

Object-oriented programming

- Languages that are based on classes are know as *Object-Oriented, such as:*
    - C++
    - Modula-3
    - Ada 95
    - Java

Object-oriented programming concepts

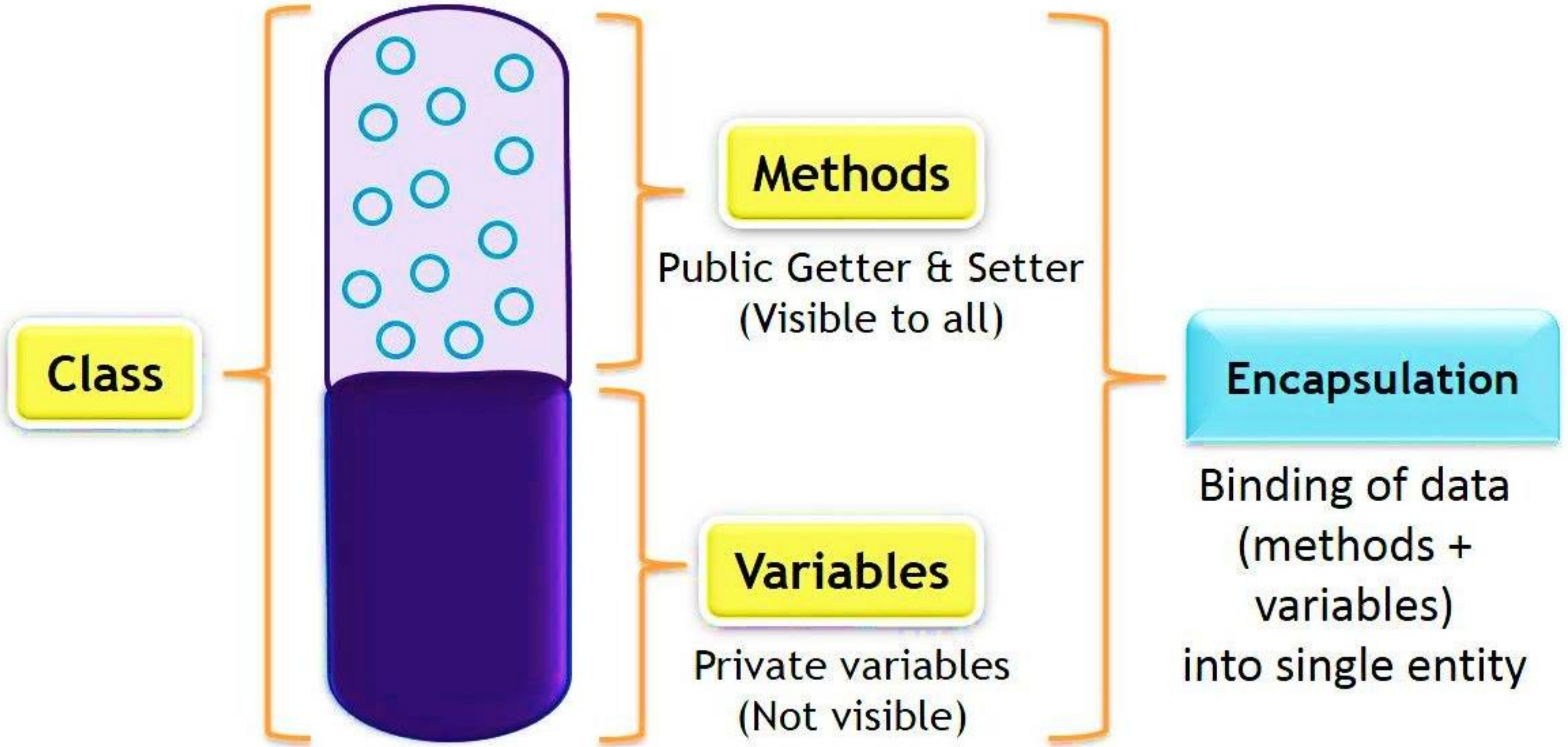
- Classes and objects
- Encapsulation
- Privacy
- Data hiding
- Inheritance
- Polymorphism
- Abstraction



OOP Principles

Encapsulation	Abstraction	Inheritance	Polymorphism
When an object only exposes the selected information.	Hides complex details to reduce complexity.	Entities can inherit attributes from other entities.	Entities can have more than one form.

Encapsulation



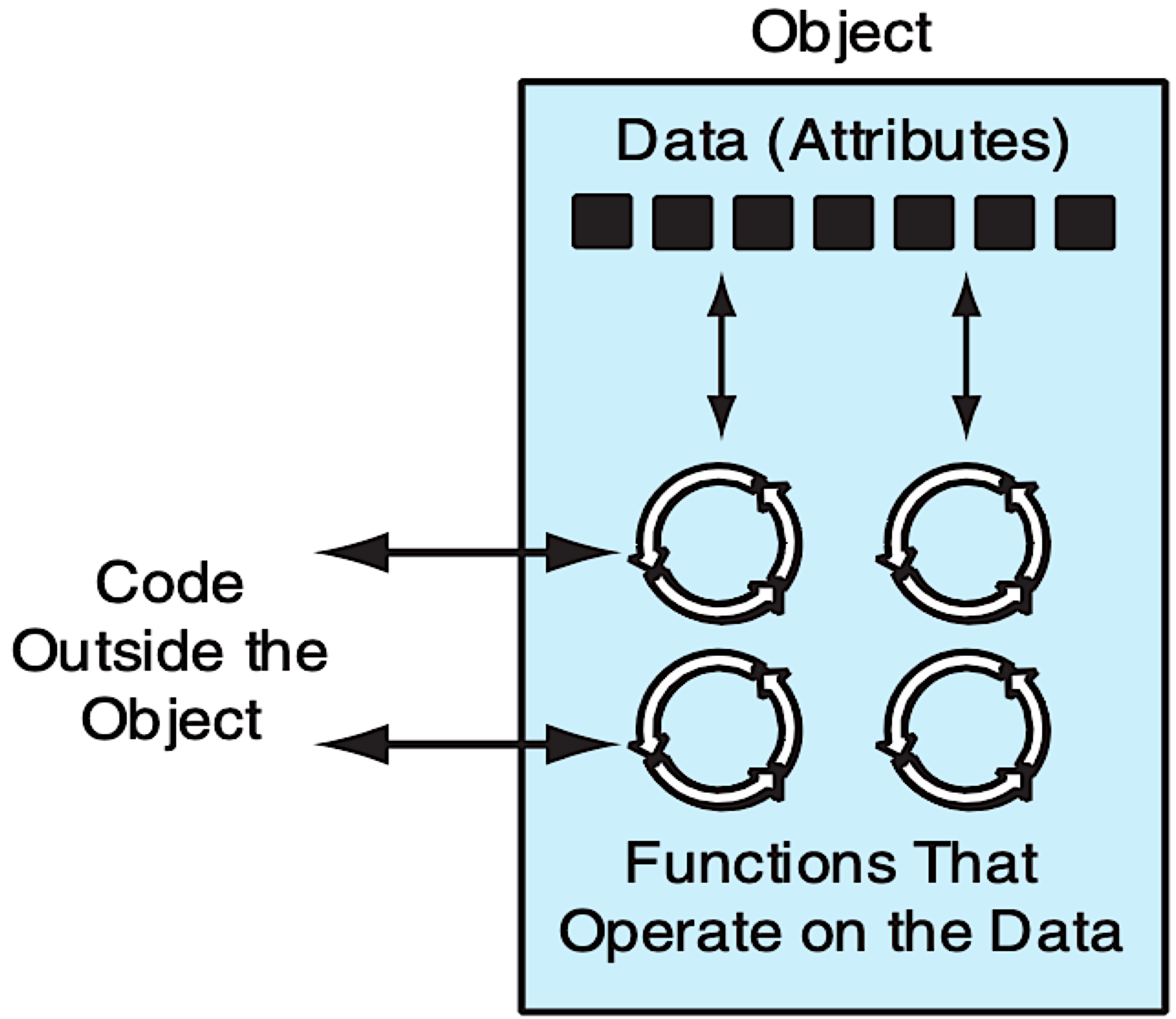
Encapsulation

- ❑ **Encapsulation** refers to the combining of data and code into a single object, called ***class***.
- ❑ **Data hiding** refers to an object’s ability to hide its data from code that is outside the object.
- ❑ Only the object’s member functions may directly access and make changes to the object’s data.

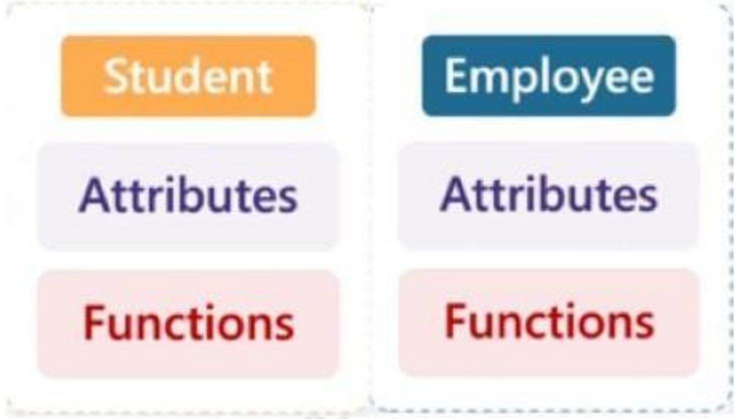
Why data hiding?

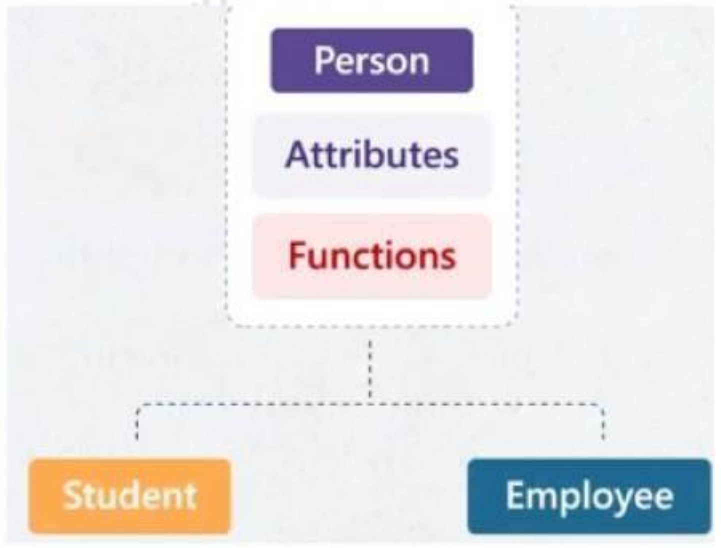
- ❑ An object typically hides its data, but allows outside code to access its member functions.
- ❑ When an object's internal data are hidden from outside code, and access to that data is restricted to the object's member functions, the data are protected from accidental corruption.

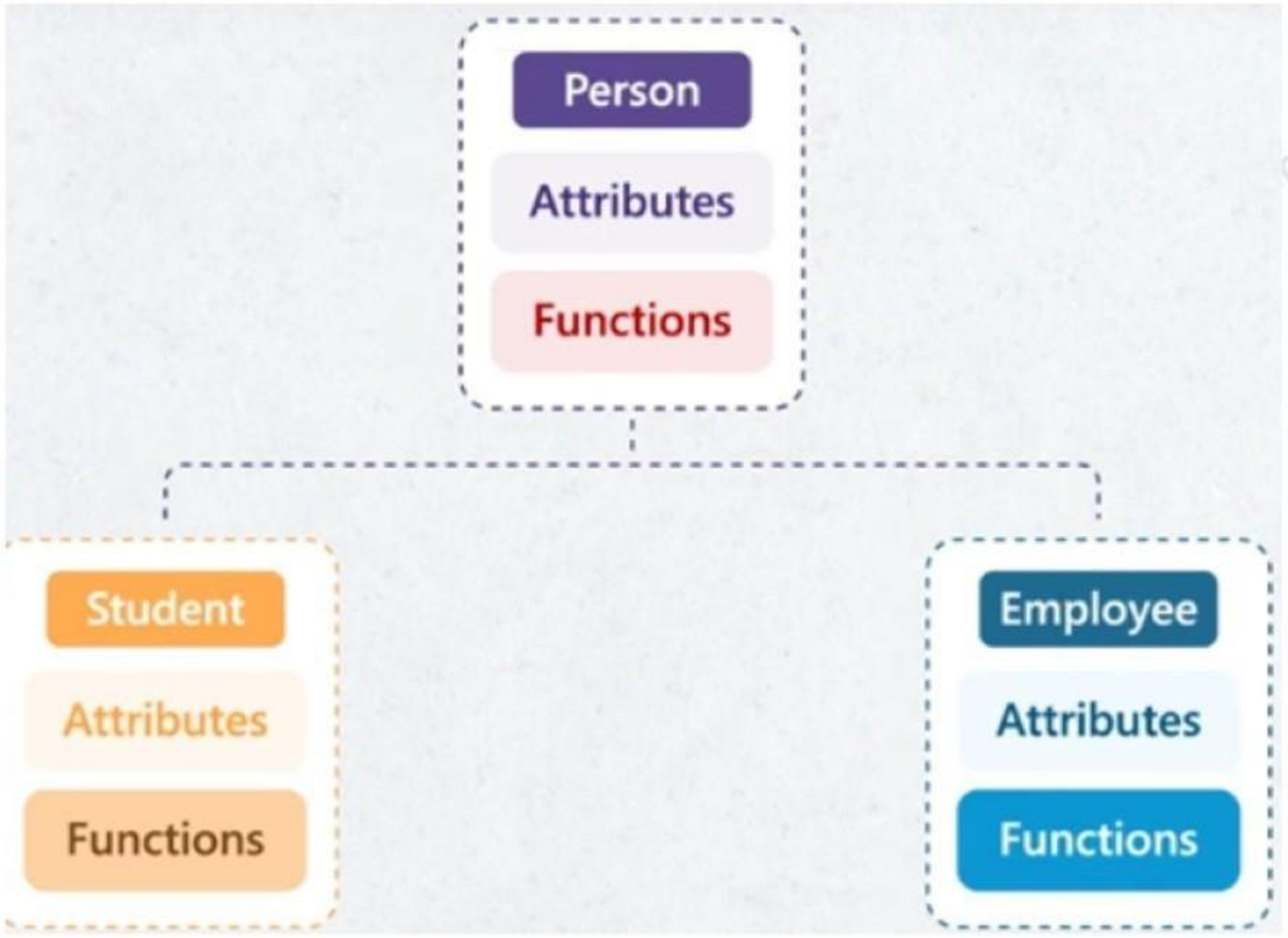
Data hiding


Object
Data (Attributes)
Code Outside the Object
Functions That Operate on the Data

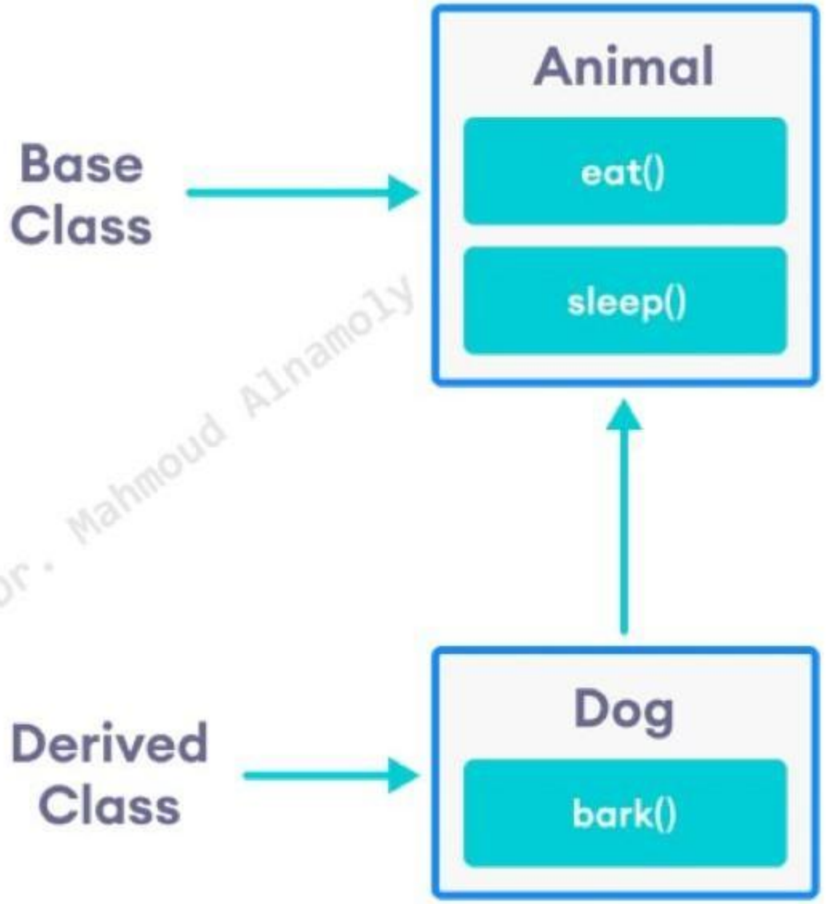
Inheritance

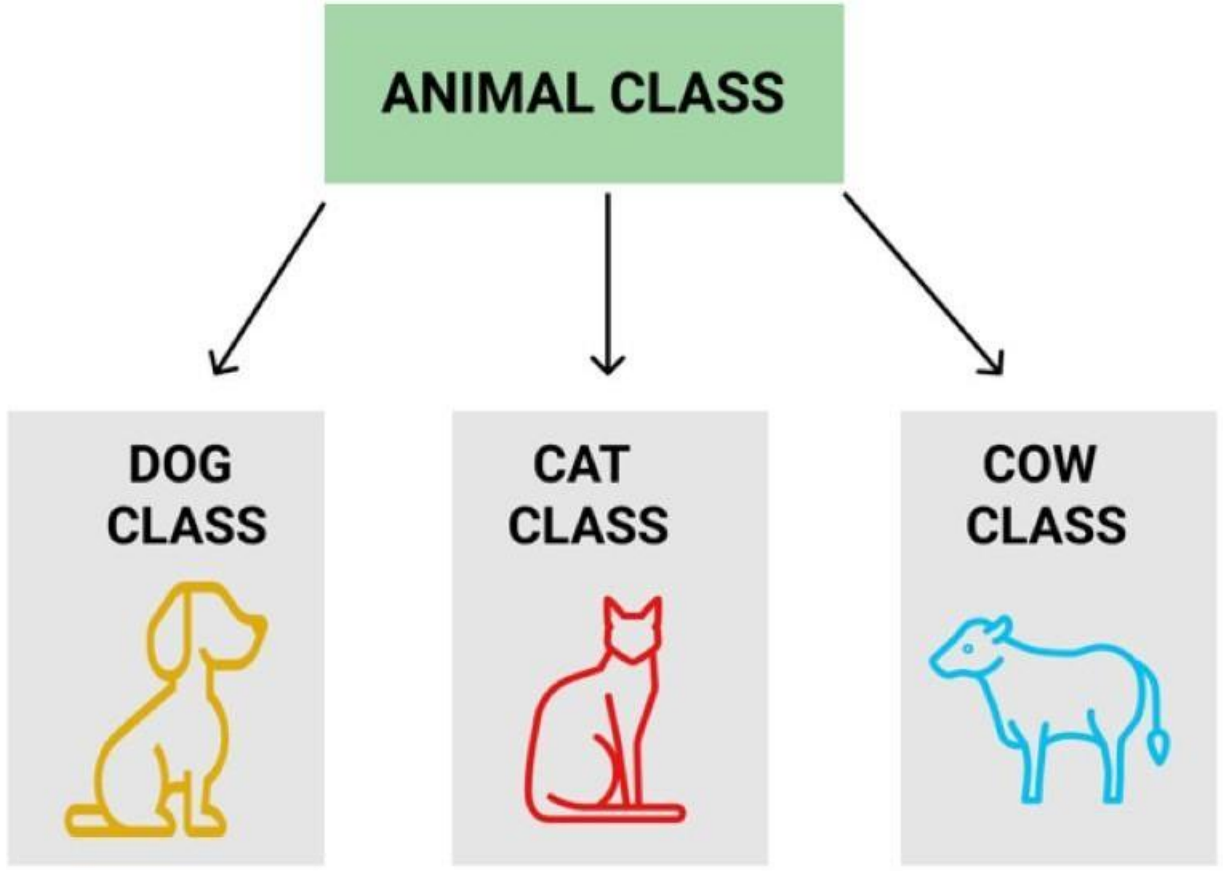






Inheritance

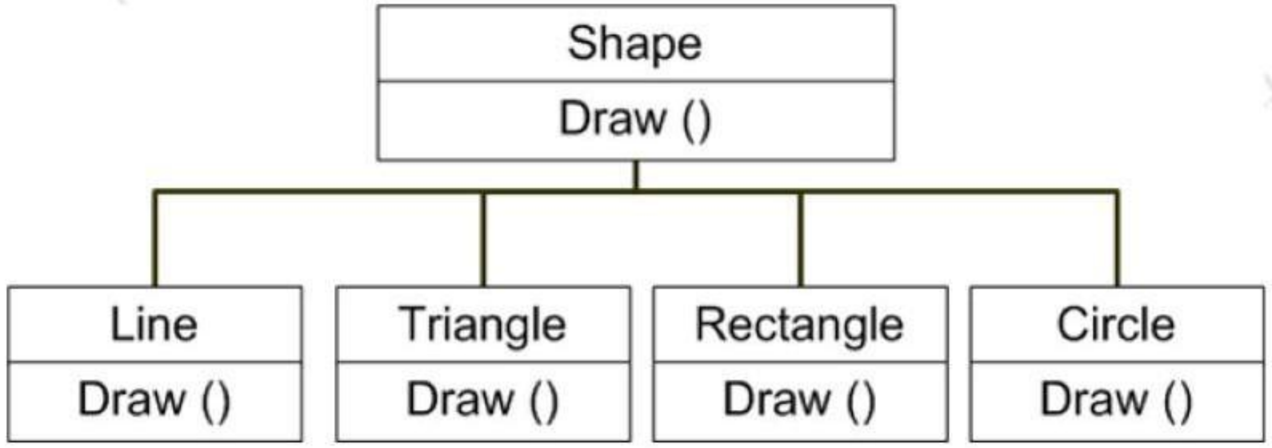


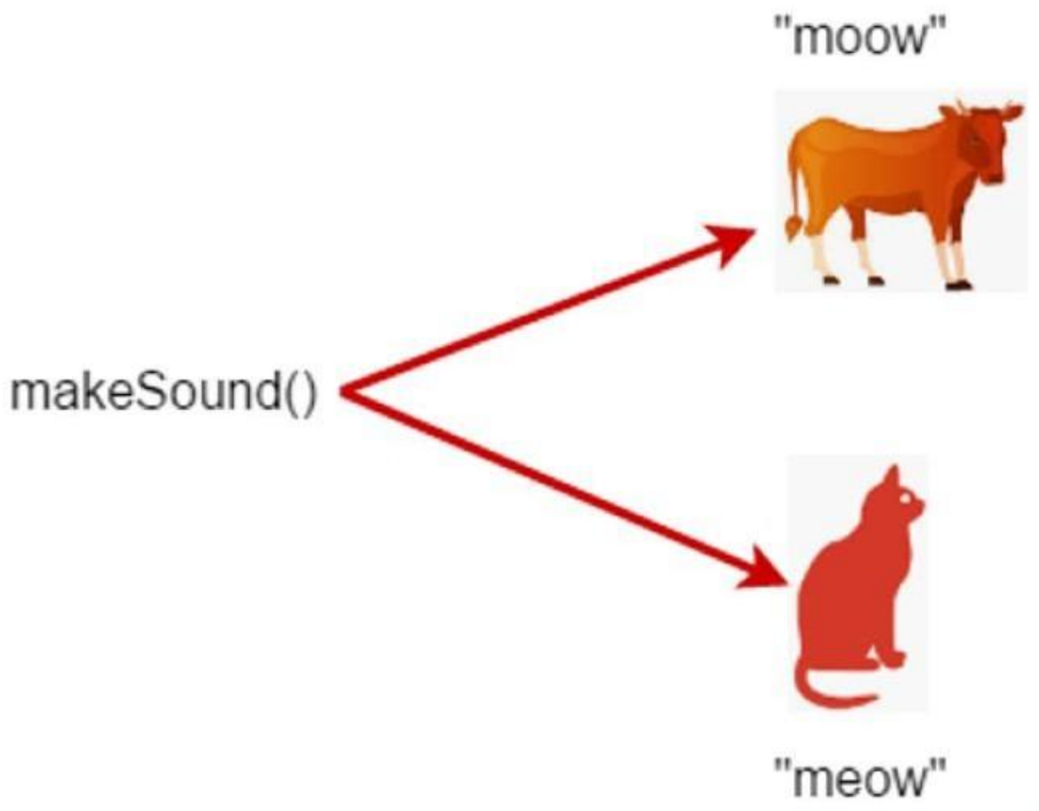


Polymorphism

Polymorphism means the ability to take more than one form. An operation may exhibit different instance. The behaviour depends upon the **type of data** used in the operation.

Without polymorphism you need to make Makesound() function for cow & another maksound() With a different body for cat





Benefit of using oop

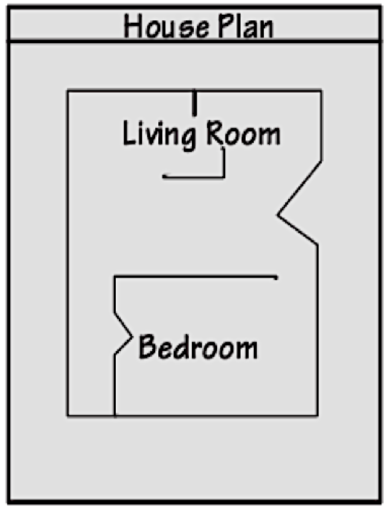
- ❑ Code can be reuse by using **inheritance**.
- ❑ Data can be hiding from outside world by using **encapsulation**.
- ❑ Operators or functions can be overloaded by using **polymorphism**, so same functions or operators can be used for multitasking.

Class in C++

- A *class* is code that species the attributes and member functions that a particular type of object may have.
- Classes are user-defined types, containing:
  - Data (data members or member variables or attributes)
  - Functions (member functions or methods or behaviors)
- A class is a blueprint that objects may be created from it.

Class in C++

Blueprint that describes a house.



Instances of the house described by the blueprint.

Class in C++

```
class class_name
{
  private:
     ….
  public:
     ….
};
```

Any valid identifier

Class body
(data member + member functions)

C++ imposes the following rules in identifiers :

1. It contains (a-z, A-Z), digits (0-9) and underscore (_).
2. It Can't start with a digit. 1stnumber number1
3. It Can't use these characters (whitespace and special characters (+, -, *, /, @, !, %, ^, &, (,), #).
4. Identifiers are case-sensitive. Number ≠ number
5. A variable name must not be any reserved word or keyword e.g. char, float etc.

Class in C++

A class definition begins with the keyword *class*.

Forgetting the semicolon at the end of a class definition is a syntax error.

The body of the class is contained within a set of braces, {   } ;

Access specifiers

- Access specifier determines the access level of the members of the class. Ex. *private, public*
- The default access specifier inside the class is private.
- The data members of a class are declared in the *private:* section of the class
- The member functions of a class are declared in the *public:* section.

Access specifiers

- Member access specifiers
  - public:
    - The member can be accessed outside the class directly.
  - private:
    - The member is accessible only to member functions of class.

- class: The keyword used to define a class.
- ClassName: The name of the class.
- { }: The curly braces enclose the class members, including data members and member functions.
- Data members: Variables declared within the class that represent the data associated with objects of that class.
- Member functions: Functions declared within the class that define the behavior of objects of that class.
- public: The access specifier that determines which members are accessible outside the class.
- private: The member is accessible only to member functions of class.

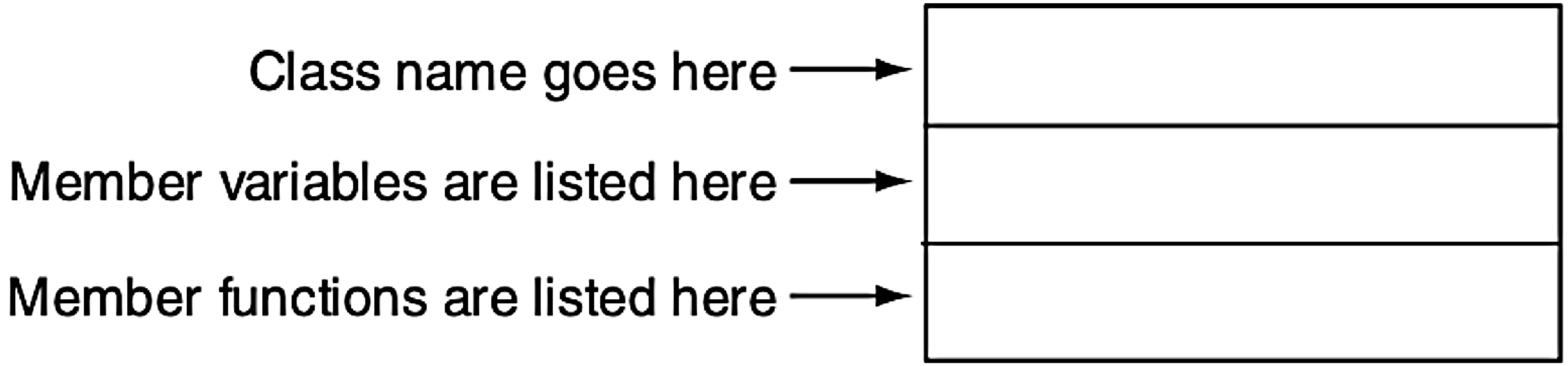
Phase 1 :-

Design Phase

“UML classes Digrams”

Unified Modelling Language

- Unified Modeling Language (UML) provides a standard method for graphically depicting an object-oriented system.


Class name goes here
Member variables are listed here
Member functions are listed here

- ✓ **Design:** specifying the structure of how a software system will be written and function, without actually writing the complete implementation

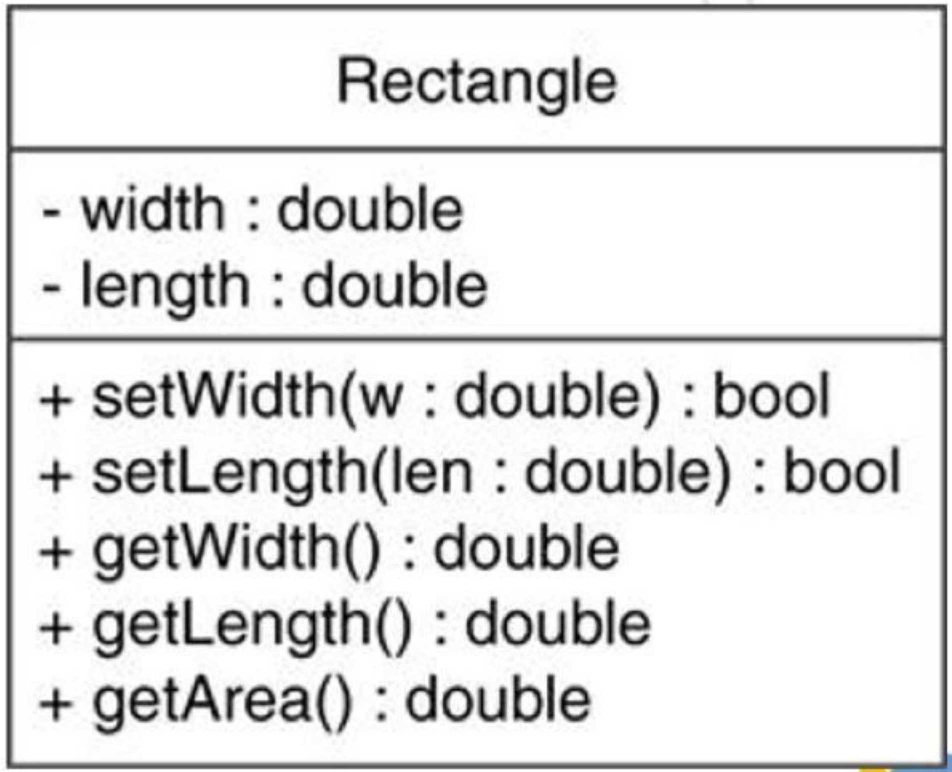
- ✓ A transition from "what" the system must do, to "how" the system will do it
  - ✓ What classes will we need to implement a system that meets our requirements?
  - ✓ What fields and methods will each class have?
  - ✓ How will the classes interact with each other?

UML : Unified Modeling Language

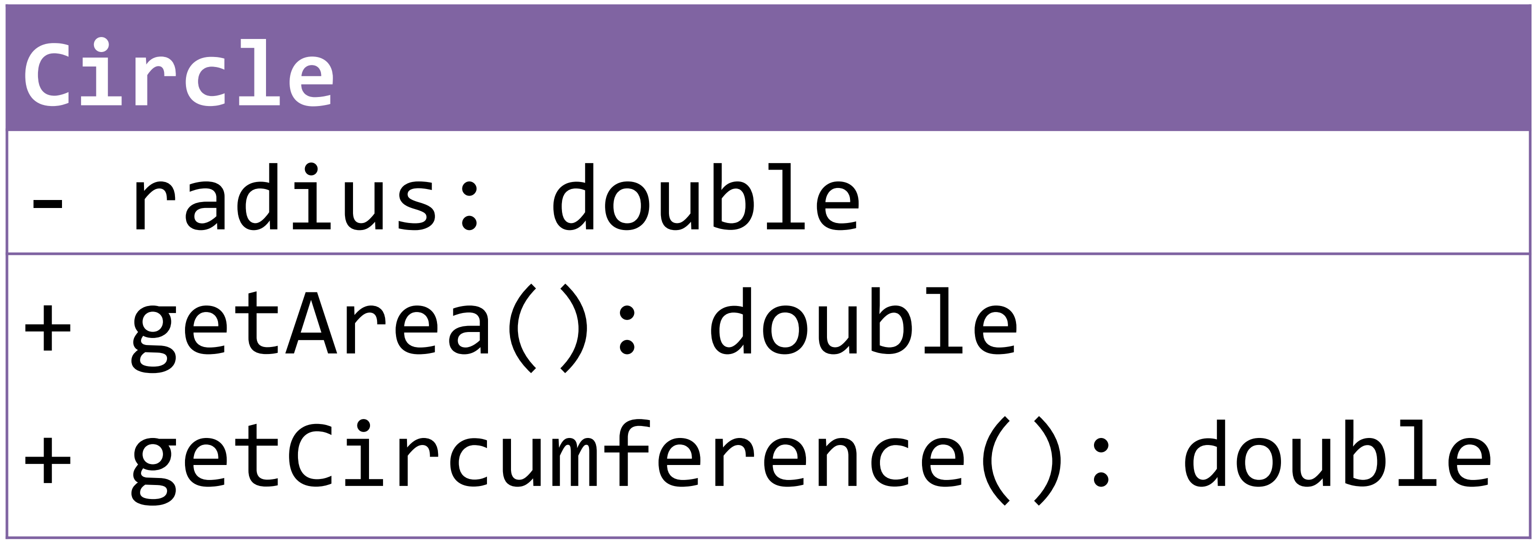
The UML provides a set of standard diagrams for graphically depicting object-oriented systems.

UML: pictures of an OO system

- ✓ programming languages are not abstract enough for OO design
- ✓ UML is an open standard; lots of companies use it



Example: Circle class



Circle class in C++

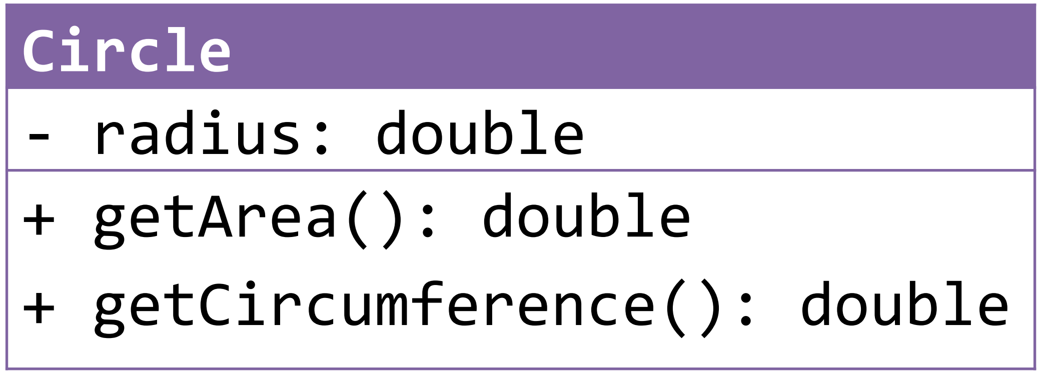
```
class Circle
{
   private:
         double radius;
   public:
        double getArea()
        {
           return radius * radius * 3.14;
        }
        double getCircumference()
        {
           return 2 * radius  * 3.14;
        }
};
```

Creating an object of a class

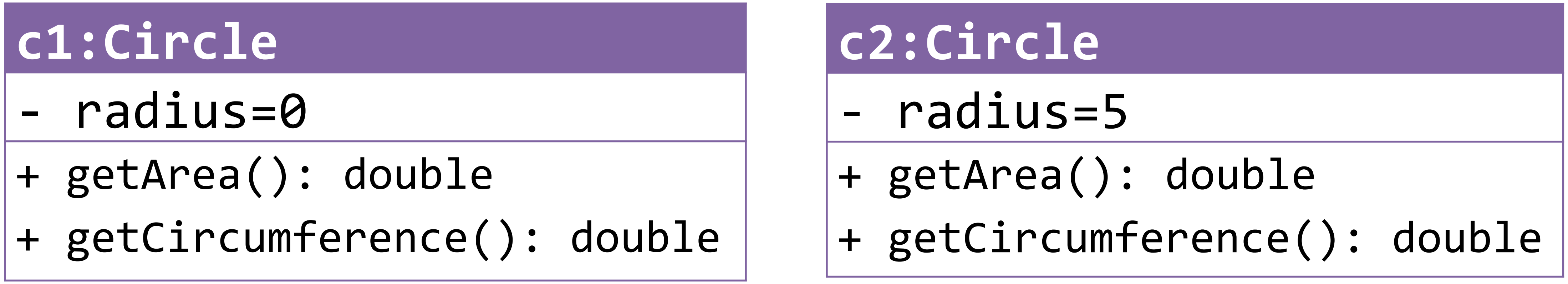
```
ClassName objectName;
```

- `ClassName` is the name of a class
- `objectName` is the name we are giving the object
- Defining a class object is called the *instantiation* of a class

Creating objects



Instances=objects



Creating an object of a Class

- Declaring a variable of a class type creates an object.
- You can have many variables of the same type (class).
- Once an object is created, a new memory location is created for it to store its data members and code.
- Class objects must be defined after the class is declared.
- You can instantiate many objects from a class type.

Ex:
```
Circle c1;
Circle c2;
Circle c3;
```

Accessing Class Members

- Operators to access class members
  - Dot member selection operator (**.**)
    - Object
  - Arrow member selection operator (**->**)
    - Pointers

Circle class in C++

```
class Circle
{
   private:
         double radius;
   public:
       double getArea()
       {
          return radius * radius * 3.14;
       }
       double getCircumference()
       {
          return 2 * radius  * 3.14;
       }
};
```

private data member

public member functions

```
#include <iostream>
using namespace std;
class Circle{
  public:
    double radius;
    double getArea() {
      return radius * radius * 3.14;
    }
    double getCircumference(){
      return 2 * radius  * 3.14;
    }
};
int main(){
   Circle c1;
   c1.radius=0;
   cout<<"Radius: "<<c1.radius<<endl;
   cout<<"Area: "<<c1.getArea()<<endl;
   return 0;
}
```

public data member + public member functions

Create an object c1 of type Circle

- **Accessing members using dot .**
- **main function can access the public radius value**

```
Radius: 0
Area: 0
```

```
#include <iostream>
using namespace std;
class Circle{
public:
   double radius;
   double getArea() {
      return radius * radius * 3.14;
   }
   double getCircumference(){
      return 2 * radius  * 3.14;
   }
};
int main(){
   Circle c1;
   c1.radius=5;
   cout<<"Radius: "<<c1.radius<<endl;
   cout<<"Area: "<<c1.getArea()<<endl;
   return 0;
}
```

public data member +
public member function

main function is an outside code of the class circle that can change the public data member.

```
Radius: 5
Area: 78.5
```

```
#include <iostream>
using namespace std;
class Circle
{
      double radius;
   public:
      double getArea()
      {
         return radius * radius * 3.14;
      }
      double getCircumference()
      {
         return 2 * radius  * 3.14;
      }
};
int main(){
   Circle c1;
   cout<<"Radius: "<<c1.radius<<endl;
   cout<<"Area: "<<c1.getArea()<<endl;
   return 0;
}
```

If the access specifier isn't determined, the default access specifier inside class is private (radius is private).

ERROR: syntax error as radius is private