
Week 2: Introduction to OOP Pillars

— By, Mahrukh Khan —

Member functions defined inside class

```
class Rectangle {  
    int width, height;  
public:  
    void set_values (int,int);  
    void print()  
    {  
        cout<<"Width ="<width<<"Height ="<height;  
    }  
};
```

Member functions defined outside class

- :: Scope Resolution Operator is used with a class name

SYNTAX

```
Returned_Type Class_Name::Function_Name(Parameter_List)  
{  
    Function_Body_Statements  
}
```

```
void Rectangle::set_values (int x, int y) {  
    width = x;  
    height = y;  
}
```

Member Function Calling

Syntax:

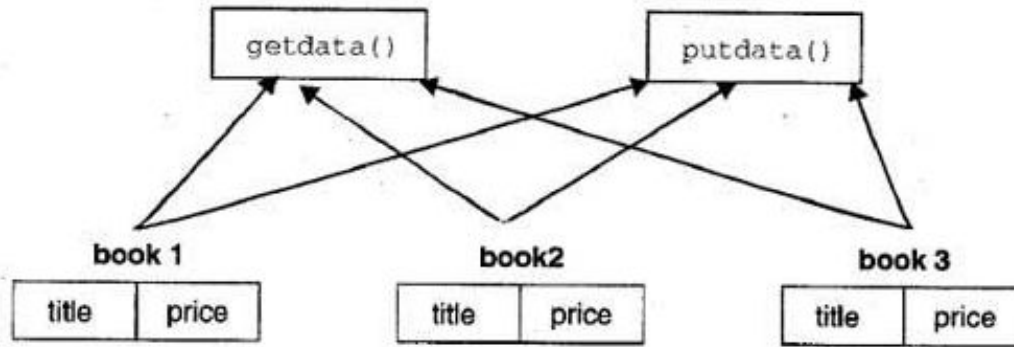
object_name.function_name ;

```
re int main ()  
{  
    Rectangle rect1;  
    rect.set_values (3,4);  
    cout << "area: " << rect.area();  
    return 0;  
}
```

Difference between Structures and Classes

STRUCTURES	CLASSES
structure by default have all its member as "public"	Visibilty can be decided
Instance of 'structure' is called 'structure variable'.	Instance of a 'class' is called 'object'.
Struct are individual elements with specific properties, so they cannot be reused.	Classes form the basic framework, they can be reused
Structures cannout utilize inheritance.	Class can be further inherited to form a subclass.

Memory Allocation for Objects



```

class Whole
{
public:
    int num1,num2;
public :
    void add()
    {
        cout<<num1+num2<<endl;
    }
    void sub()
    {
        cout<<num1-num2<<endl;
    }
    void mul();
    void div();
    void modd();
};

void Whole::mul()
{
    cout<<num1*num2<<endl;
}

void Whole::div()
{
    cout<<num1/num2<<endl;
}

void Whole::modd()
{
    cout<<num1%num2<<endl;
}

```

```

int main()
{
    Whole obj1;
    obj1.num1=25;
    obj1.num2=5;
    obj1.add();
    obj1.sub();
    obj1.mul();
    obj1.div();
    obj1.modd();
}

```

```

C:\Users\Administrator\Desktop\CP-Spring2019\Lessons\
30
20
125
5
0
-----
Process exited after 0.01487 seconds with
Press any key to continue . . .

```

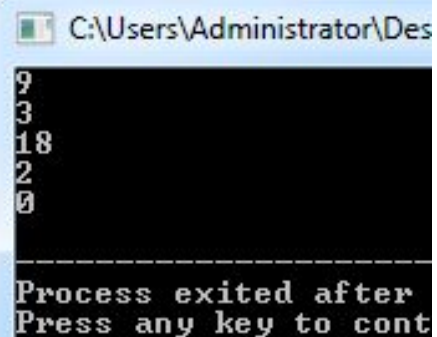
```
2 using namespace std;
3 class Whole
4 {
5     private:
6     int num1,num2;
7     public :
8     void add()
9     {
10         cout<<num1+num2<<endl;
11     }
12     void sub()
13     {
14         cout<<num1-num2<<endl;
15     }
16     void mul();
17     void div();
18     void modd();
19 };
20 void Whole::mul()
21 {
22     cout<<num1*num2<<endl;
23 }
```

Compiler (5) Resources Compile Log Debug Find Results Close

Col	File	Message
	C:\Users\Administrator\Desktop\CP-Spring2019\Less...	In function 'int main()':
7	C:\Users\Administrator\Desktop\CP-Spring2019\Lessons...	[Error] 'int Whole::num1' is private
7	C:\Users\Administrator\Desktop\CP-Spring2019\Lessons...	[Error] within this context
12	C:\Users\Administrator\Desktop\CP-Spring2019\Lessons...	[Error] 'int Whole::num2' is private
7	C:\Users\Administrator\Desktop\CP-Spring2019\Lessons...	[Error] within this context


```
void setNum1(int n1)
{
    num1= n1;
}
void setNum2(int n2)
{
    num2=n2;
}
```

```
int main()
{
    Whole obj1;
    obj1.setNum1(6);
    obj1.setNum2(3);
    obj1.add();
    obj1.sub();
    obj1.mul();
    obj1.div();
    obj1.modd();
}
```



```
C:\Users\Administrator\Desktop
9
3
18
2
0
-----
Process exited after
Press any key to cont
```

Interface

- define and standardize the ways in which things such as people and systems interact with one another.
- describes what services a class's clients can use and how to request those services, but not how the class carries out the services.
- A class's public interface consists of the class's public member functions (e.g. Interface of a phone).

- Screen

- Keypad

- Mic

- Ear piece

- Input Number

- Place Call

- Disconnect Call

- Add number to address book

- Remove number

- Update number

Separation of Interface and Implementation

- Means change in implementation does not affect object interface
- This is achieved via principles of information hiding and encapsulation
- E.g.
 - A driver can drive a car independent of engine type (petrol, diesel), Because interface does not change with the implementation.
 - E.g. A driver can apply brakes independent of brakes type (simple, disk).

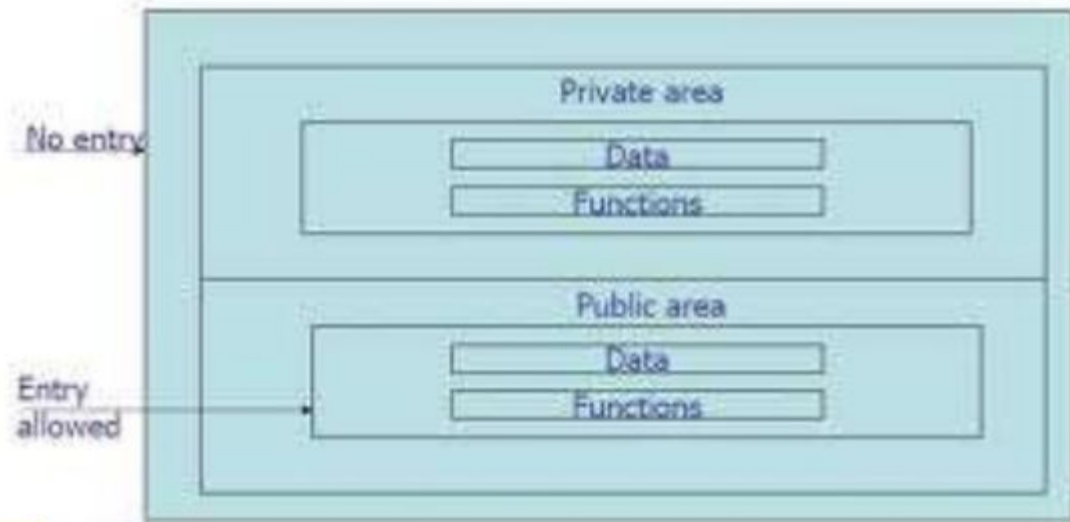
OOP Pillars

- Encapsulation
- Inheritance
- Abstraction
- Polymorphism

Encapsulation

- Information hiding
- Binding together of data and functions in one place
- protects the integrity of the data
- prevents it from being needlessly altered

Encapsulation



Inheritance

Different kinds of objects often have a certain amount in common with each other.

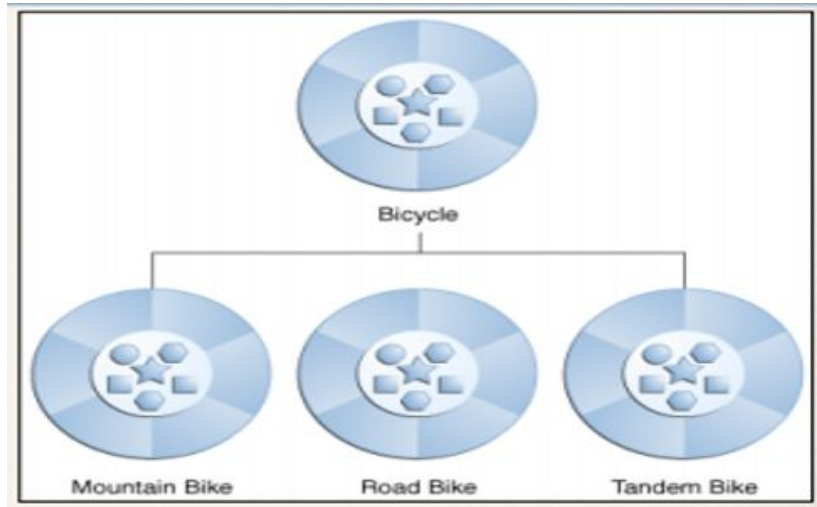
- ♦ Mountain bikes, road bikes and tandem bikes all share characteristics of bicycles.

Each kind also defines additional features that make them different.

- ♦ Tandem bicycles have two seats and two sets of handlebars.
- ♦ Road bikes have drop handlebars.
- ♦ Mountain bikes have an additional chain ring.

Inheritance

Object Oriented Programming allows class to inherit commonly used state and behaviour from other classes.



Inheritance

```
class Person {  
    String name;  
    int age;  
    void birthday () {  
        age = age + 1;  
    }  
}
```

```
class Employee  
    extends Person {  
    double salary;  
    void pay () { ...}  
}
```

Every **Employee** has **name** and **age** fields and **birthday** method *as well as* a **salary** field and a **pay** method.

Inheritance

- reuse existing code
- reduces the time
- derived class/Superclass / Parent Class
- Base class/ Subclass/ Child class
- Example: Windows operating system.

Polymorphism

- different forms at different times (poly + morphos)
- Polymorphism is the ability of two different objects to respond to the same request message in their own unique way.
- For example, I could train my dog to respond to the command bark and my bird to respond to the command chirp. On the other hand, I could train them to both respond to the command speak. Through polymorphism I know that the dog will respond with a bark and the bird will respond with a chirp.

Example

Create a class called Employee that includes three pieces of information as data members—a first name, a last name and a monthly salary (type int). Your class should have a member function that initializes the three data members. If the monthly salary is not positive, set it to 0. Write a test program that demonstrates class Employee's capabilities. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10 percent raise and display each Employee's yearly salary again.

Example

Define a class to represent a Bank account. Include the following data members.

1. Name of the depositor
2. Account number.
3. Type of account.
4. Balance amount in the account.
5. Rate of interest

Also provide Member Functions:-

1. To deposit amount.
2. To withdraw amount after checking for minimum balance.
3. To display all the details of an account holder.
4. Display rate of interest.

Identifying classes

Think of building the system from parts, similar to constructing a machine.

Each part is an object which has its own attributes and capabilities and interacts with other parts to solve the problem.

Identify classes of objects that can be reused.

Think in terms of objects and their interactions.

At a high level, think of an object as a thing-in-its-own-right, not of the internal structure needed to make the object work.

Identifying classes

- use nouns, pronouns, noun phrases to identify objects and classes
- singular → object, plural → class
- not all nouns are really going to relate to objects
- identify individual or group "things" in the system/problem
- people
- places
- things
- organizations
- concepts
- events

Assignment # 1

Identify classes, class attributes and their member functions in following scenarios.

- University
- Library
- Bank