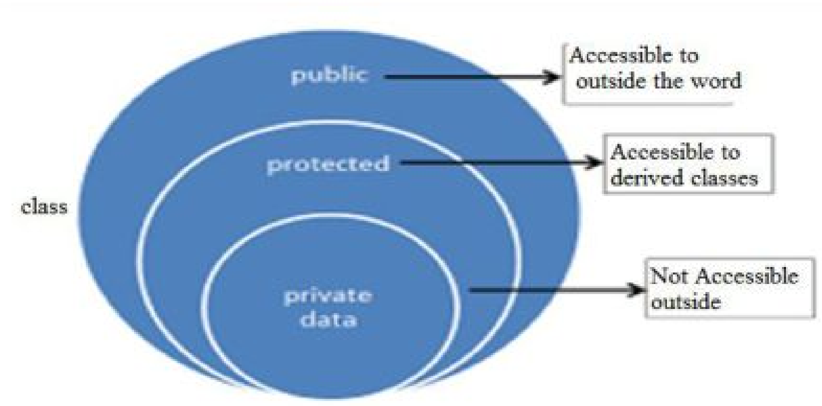
Major concept of the OOP(S)

- ***Encapsulation***
- **Data shadowing**
- **Function overloading**
- Data Hiding
- Function Overriding
- Function Hiding
- Up casting

Encapsulation

- Encapsulation is the *mechanism that binds together code and the data it manipulates.* Other way to think about encapsulation is, ***it provides protective shield that prevents the data from being accessed by the code outside this shield.***
- Technically in encapsulation, **the variables or data of a class is hidden from any other class and can be accessed only through any member function of own class in which they are declared.**



Data shadowing:

- ***If a parent class & a child class both are having static data member with the same name***, this concept is called ***Data Shadowing***

```
#include<iostream>
using namespace std;
class Base
{
        public:
                static int i;
};
int Base::i= 10;

class Derive : private  Base
{
        public:
```

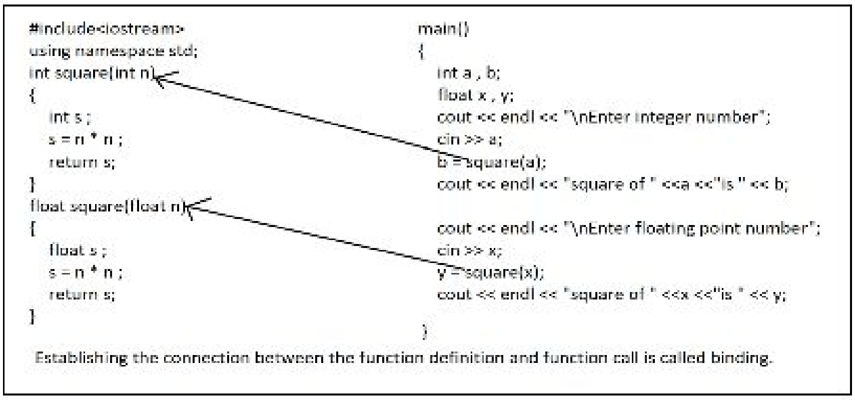
```
                        static int i ;
};
int Derive::i = 20 ;
main()
{
        cout << endl << Derive::i;
        //cout << endl << Derive::Base::i;

}
```

Function Overloading

- ***C++ allows us to create multiple functions with same name.***
- The purpose of ***same name reduces the calling complexity of Naming***(by remembering only name)
- We can create ***multiple function with same name but their argument must different***
  - **Type of parameter**
    - **int function (int);**
    - **float function (float);**
  - **Order of parameter**
    - **void function(int , float);**
    - **void function(float, int);**
  - **Number of parameter**
    - **int function (int , int , int);**
    - **int function (int , int);**
- ***This overall concept is called function overloading.***

e.g.

```
#include<iostream>
using namespace std;
int squareint(int n)
{
        int s ;
        s = n * n ;
        return s;
}
float squarefloat(float n)
{
        float s ;
        s = n * n ;
        return s;
}
```



```
#include<iostream>                          main()
using namespace std;                         {
int square(int n)                               int a , b;
{                                               float x , y;
   int s ;                                      cout << endl << "\nEnter integer number";
   s = n * n ;                                  cin >> a;
   return s;                                    b = square(a);
}                                               cout << endl << "square of " <<a <<"is " << b;
float square(float n)
{                                               cout << endl << "\nEnter floating point number";
   float s ;                                    cin >> x;
   s = n * n ;                                  y = square(x);
   return s;                                    cout << endl << "square of " <<x <<"is " << y;
}
                                             }
```

Establishing the connection between the function definition and function call is called binding.

- When function name is same but their parameter are different ion called ***function overloading.***
- In Function overloading, ***return type of the function does not play any role.***

- Function overloading required ***binding***.
  - ***Establishing the connection between the function call and the function definition is known as binding.***
  - Binding are two type :
    - **Compile Time binding /static polymorphism/ false polymorphism**
      - When this binding done by the compiler itself known ***compile time binding***.
    - **Run Time binding /dynamic polymorphism/true polymorphism**
      - When this binding done by the runtime/execution time is known ***run time binding.***
- ***Name mangling***
  - When we do the function overloading in program compiler uses the concept of ***name mangling.***

IQ: is there is any role, of access specifier in function overloading?

Ans: No

IQ: is there is any role, of return type in function overloading?

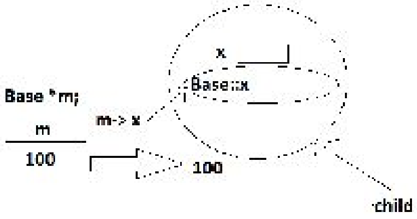
Ans: no

Data Hiding

- ***If a parent class & a child class both are having non static data member with the same name***, this concept is called ***Data Hiding.***

e.g.

```
#include<iostream>
using namespace std;
class Base
{
        public:
                int x ;
                Base()
                {
                        this->x = 10 ;
                }

};
class Child : public Base
{
        public:
                int x  ;
                Child()
                {
                        this->x = 20;
                }
```



```
    void show()
    {
        Base *m = this;
        cout << endl << m->x;
        cout << endl << this->x;
    }
};

main()
{
    Child c ;
    c.show();
}
```

- Base class pointer not only store the address of base class object as well as store the address any object which directly or indirectly inherit from the base class
- Using the base class pointer, we can only access those members they are direct members of the base class.

- ***Function Overriding***
    - ***If a parent class and a child class both are having same Signature of function*** then this concept is called **function overriding.**

```
#include<iostream>
using namespace std;
class Base
{
    public:
        void show(){
            cout << endl << "Base";
        }
};
class Child : public  Base{
    public:
        void show()
        {
            cout << endl << "Child";
        }
};
main()
{
    Child c;
    c.show();
}
```

- ***Function Overriding V/S Function Overloading***

  - To achieve Overriding *Signature of functions* are same in both base and the derive class where as in Overloading function name is same but parameter must be different.
  - return type play important role in overriding where in overloading does not play any role in overloading.
  - Overriding is example of Run Time Polymorphism where Overloading is the example of Compile Time Polymorphism.
  - Objective of overriding is to redefine the member of the base class by suppressing whereas Overloading is for the Programming calling convenience.
  - Need of Inheritance for Overriding whereas no need for Overloading.
  - Overriding is only of non-static member function whereas overloading is for both type of member functions.

- ***Function hiding***
  - ***If parent & child both having the same static function.*** **This concept is called *function hiding.***

    e.g.

```
#include<iostream>
using namespace std;
class base
{
        public:
                static void show()
                {
                        cout << endl << "static show of base";
                }
};
class derive : public base
{
        public:
                static void show()
                {
                        cout << endl << "static show of child ";
                }
};

main()
{
        derive::show();
        derive::base::show();
}
```

- ***Up-Casting***
  - Address of a child class object can be put into parent class pointer variable such concept is called Up-Casting.

```
Base *b = new Child();  //up casting
```

- Note
    - **As we say base class pointer variable stores the address any class object which directly or indirectly inherit from the base class**
    - **Using the base class pointer variable we can access only those members which direct member of base class.**