

# What is the purpose of Data Hiding?

- Data hiding is mistaken as data security.
- By hiding data a programmer can avoid mishandling of data by the code outside the class.
- Data is accessible only within a class.
- Only the functions of a class, knows what processing should be done on that data

## Encapsulation vs Abstraction

- A Class is used for Encapsulation as well as Abstraction
- **Encapsulation** means combining related data members and functions together.
- Encapsulation is achieved using a class.
- **Abstraction** means hiding data and showing required functions.
- Abstraction is achieved using encapsulation and data hiding.  
Private and public.

## structure vs class

- C++ supports both structure and class
- **structure** in C++ can have data members and member functions.
- All members of a structure are public by default.
- **class** can contain data members and member functions
- All members of a class are private by default.

# Where the class is stored?

- In C++, class is not stored in main memory
- All member functions of a class are stored in Code section at loading time.
- If an object of a class is created inside a function then memory for all data members will be created in a stack of a function.

# What is “this” pointer?

- It represents current object.
- It is useful in avoiding variable name conflict.

## Example for current object.

```
class Test{
    private:
        int value;
    public:
        void fun(int x){
            this->value=x;
        }
};

int main(){
    Test t1,t1; // both t1 and t2 are having “value”
    t1.fun(10); // “this” inside fun means “value” of t1.
    t2.fun(20); // “this” inside fun means “value” of t2.
}
```

## Example for variable name conflict.

```
class Test{
    private:
```

```

        int value;
    public:
        void fun(int value){
            // here parameter name and data member name is same
            "value" to access data member "this" is used.
            this->value=value;
        }
};

```

## Create object in Stack or Heap

### Stack:

**Rectangle r1;** // Valid

**Rectangle r1();** // invalid, don't give empty brackets.

### Heap:

**Rectangle \*p;** // pointer, it is created in stack.

**p=new Rectangle();** // object is created in heap. Empty () can be given.

## When to Create Object in Heap?

- If you want the object to be access in more than one functions of a program then create an object in heap.
- It can be accessed using pointer.
- Any function in a program can access it, if it is having a pointer to it.

## Pointer size

Every pointer takes **8 bytes** of memory in latest compiler.

Size of pointer is not dependent on its datatype.

Note: I have assumed that pointer takes 2 bytes, to make explanation easy

## **'->' vs '.'**

**Stack:** if an object is created in stack, use '.'

**Rectangle r1;**

**r1.area();**

**Heap:** if an object is created in heap then use '->'

**Rectangle \*p;**

**p=new Rectangle();**

**p->area();**