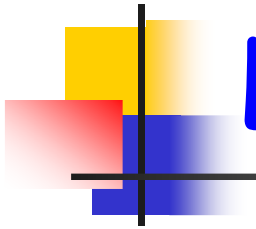
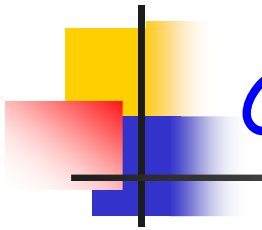


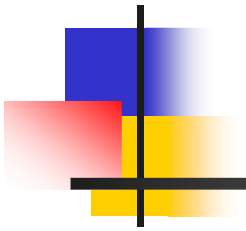
Lecture-3

- C++
  - Concepts of Object oriented Programming
  - Concept of class and Object
  - Constructor and destructor
  - Data and Member functions
  - Data encapsulation
    - public, private and protected members

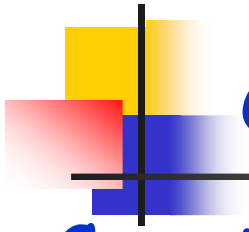


C++ — Philosophically different from C

- High level features of C++
  - Uses concepts of "object oriented programming" (OOP)
  - Everything that works in C works in C++
    - C syntax, operators, structures, control statements, etc. work in C++
    - Reverse is NOT true
- Object Oriented Programming
  - Concept of class/object, methods, inheritance, encapsulation, abstraction, polymorphism
  - Key concepts in this
    - Separation of data and methods



Constructor and Destructor



Constructor and destructor … contd.

Constructor

- A function with the same name as the class
- Called when an object is created
- A class can have more than one constructor

Destructor

- Called when an object is cleaned up (goes out of scope)
- One class can have only one destructor

Examples

```
account x; // constructor code is called
account *y = new account; // constructor code is called
delete (y); // destructor code is called
```

Constructor and destructor

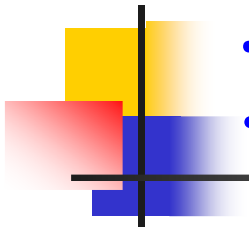
Constructor code

```
account::account( )
{ user_ssn = -1; accountNumber = -1; }

account::account( ) : user_ssn (-1),
                                accountNumber(-1) { }
account::account (int ssn, int acctNum)
{
      user_ssn = ssn;
      accountNumber = acctNum;
}
```

Destructor code

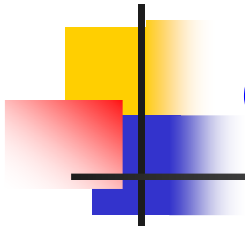
```
account::~account( )
{  // Any memory/resource cleanup, etc. }
```



Initializing member values

```
class Account
  {
     private:
       int balance;
     public:
       Account ( ) : balance (0)
         { }
       Account (int amount) :
   balance (amount) { }
  };
```

```
class checkingAccount : public
    Account
{
  checkingAccount (int amount)
    :  Account (amount) { }
}
```



Class methods

Syntax:

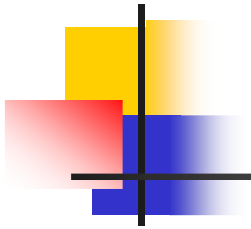
```
<ret_type> class::functionName(args)
{
  // code
}
```

Method code can be present in class definition

- Outside the class definition
- In a separate file

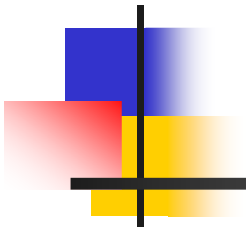
Example

```
void account::withdrawMoney (int amount)
{
  // code
}
```



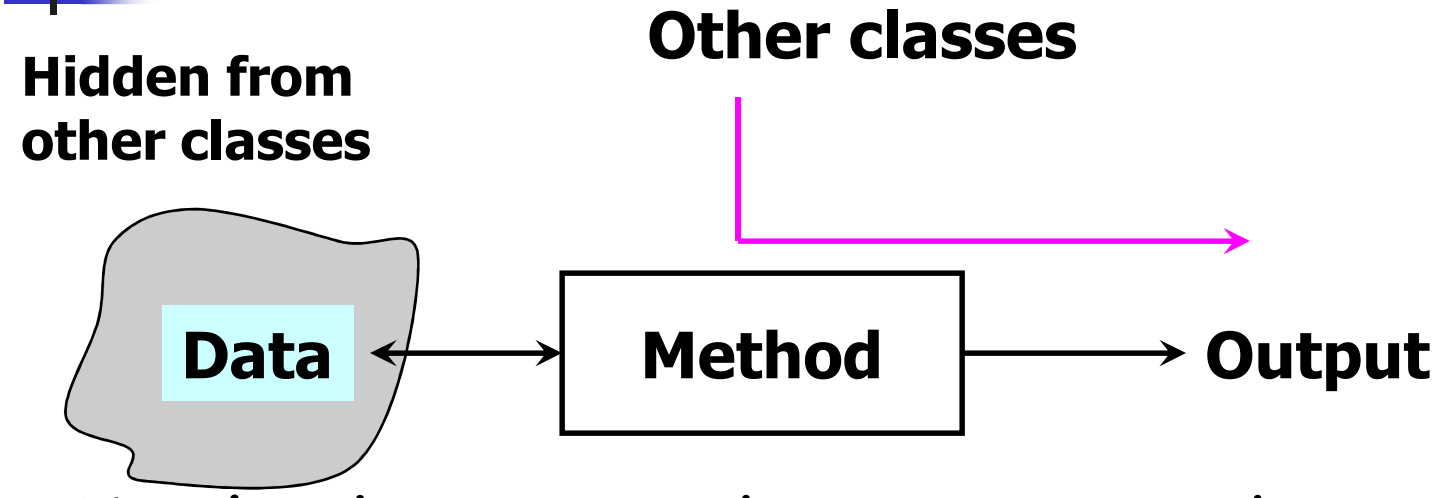
A simple "account" example

```
class account
{
  private:
     int user_SSN;                  // data
     int accountNumber;             // data
  public:
     void withdrawMoney (int amount);   // method
     void depositMoney (int amount);    // method:
     void computeInterest( );           // method
};
account x;  // x is an object of class "account"
```

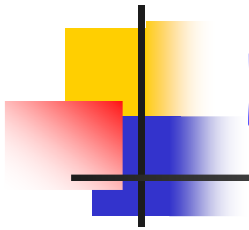


Data encapsulation

Data encapsulation … contd.

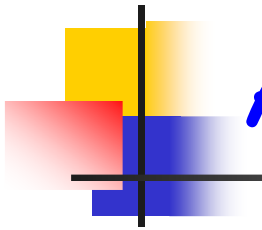


- Methods act on data to provide output.
- User needs to see only method, not data.
- User should not be affected by
  - Implementation details of methods.
  - Changes in implementation of methods.



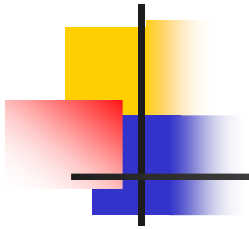
Data encapsulation

- Provide access restrictions to member data and functions
  - From other classes and functions.
- Implemented y using access modifiers
  - **public**, **private** and **protected**
- Other classes, functions need to know what methods are implemented
  - Not how they are implemented



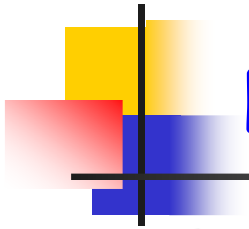
Account example … contd.

- class has both "data" and "methods".
- Attributes and methods are "members" of a class
- An instance of a class is an object.
- A class should typically correspond to some meaningful entity.
- A class uses methods to interact with other classes/functions.
- private members accessible only to the class (and friends)
- public members are accessible to every class and functions



Back to data encapsulation

- How can data be hidden?
  - Only class should have access to data
  - Class methods use data
- Define every class member to be one of
  - public – accessible to every class, function
  - private – accessible only to class and friends
  - protected – accessible only to class, friends and children



Data encapsulation in account example

- In an object of account
  - user_ssn and accountNumber are declared private
    - Accessible only to account objects (and friends)
  - Methods are public
    - Anyone can access them.

- Example

```
void function1 ( ) // function, not defined in Account
 {
        account x;
        x.user_ssn = 123; // Will NOT work
        x.computeInterest ( ); // Will work
}
```