

MIT WORLD PEACE UNIVERSITY

Computer Networks  
Second Year B.Tech Semester 3  
Academic Year 2022-23

---

---

MODULE 1 - CLASS NOTES

---

---

NOTES

Prepared By

P34. Krishnaraj Thadesar

Batch A2

September 12, 2022

## **Contents**

<b>1 Things to do</b>	<b>2</b>
<b>2 Inheritance</b>	<b>2</b>
2.1 Differenece between overloading and overiding . . . . .	2
2.2 Benefits of using Inheritance . . . . .	2
<b>3 Class Derivation in C++</b>	<b>2</b>
<b>4 Types of Inheritance</b>	<b>3</b>
4.1 What Access modifiers mean when inheriting . . . . .	3
4.2 Constructors and Destructors in Base and Derived classes . . . . .	3
<b>5 Overriding Member Functions</b>	<b>3</b>
<b>6 Virtual Base Class</b>	<b>4</b>
<b>7 Inheritance in Java</b>	<b>4</b>
<b>8 Operator Overloading</b>	<b>4</b>
8.1 Types of Polymorphism . . . . .	5
8.2 Pure Virtual Functions . . . . .	5
8.3 Abstract Class . . . . .	6
8.4 Virtual Destructor . . . . .	6

### 1 Things to do

1. Types of Inheritance
2. Virtual base Classs
3. Polymorphism
4. Vitual functions

### 2 Inheritance

- It is the mechanism by which one class acquires the properties of another class
- Provides a way to create a new class from an existing class
- The new class is a specialized fersion of the existing class
- Inheritance establishes an "is a" relationship or a parent child relationship between classes.
- Allows sharing off the behavior of the parent class into its child classes
- child class can add new behavior or override existing behaviour from parent
- It allows a hierarchy of classes to be built moving from the most general to the most specific class.

#### 2.1 Differnece between overloading and overiding

- Overloading is when you write the same function many times within the same class
- Overriding is when you do that same thing, but in sub classes.

#### 2.2 Benefits of using Inheritance

- Reusablity : Reuse the methods and data of the existing class
- Extendability: Extend the existing class by adding new data and new methods.
- Modifyability: Modify the existang class by overloading its methods with newer implementations, saves memory space, increases reliability, saves the developing process.

### 3 Class Derivation in C++

syntax: `class DerivedClassName : specification BaseClassName`  
like `class child : public parent() // private by default ;`

## **4 Types of Inheritance**

1. Single level Inheritance : You have 1 base class → 1 Child class.
2. Multiple Inheritance : 2 or more Base Classes → 1 Child Class
3. Multi-Level inheritance : 1 Base Class → 1 Child Class → Another Child Class and so on
4. Heirarchical Inheritance : 1 Base Class → 2 or more Child Classes.
5. Hybrid Inheritance : Any legal combination of any of these things.

### **4.1 What Access modifiers mean when inheriting**

1. If you do `class child : private parent;` then every private data member becomes inaccessible, coz anyway thats what should happen, then the protected data members become private, and public data members also become private.
2. If you do `class child : protected parent;` then its the same thing, except you still cant access private variables, but protected and public data members become protected
3. Same with `class child : public parent;` everything remains unchanged. The objects will behave in accordance with the usual laws of objects.

### **4.2 Constructors and Destructors in Base and Derived classes**

1. Derived classes can have their own constructors and destructors
2. When an object of a derived class is created, the base class's constructor is executed first followed by the derived class's constructor
3. In case of multiple inheritances, the base classes are constructed in the order in which they appear in the declaration of the derived class.
4. For destructors, the order is reversed.

## **5 Overriding Member Functions**

- If a base and derived class have member functions with same name, and arguments then method is said to be overridden and it is called as "function overriding" or "method overriding".
- The Child class provides alternative implementation for parent class method specific to a particular subclass type.
- You might need to do this if your child class has something to add to the previous definition. You could still call it from that function.
- If you have multiple functions tho, you could have some ambiguity in your code, and to fix that you could use the scope resolution operator.

### 6 Virtual Base Class

- In hybrid inheritance child class has two direct parents which themselves have a common base class.
- So you can prevent multiple copies of the base class coming into the child class by declaring the base class as virtual when its being inherited.
- So like imagine you have 2 base classes each inheriting the same class. Now imagine a third class that inherits from both of them. So the base, or the grandparent classes methods are copied twice. You can prevent this by declaring them as virtual base classes.

### 7 Inheritance in Java

- It is pretty Much similar to cpp

Syntax:

```
class derived_class extends base_class Name
{
// methods and stuff.
}
```

### 8 Operator Overloading

- Operator overloading is a feature in C++ Programming that allows programmer to redefine the meaning of an existing operator when they operator on class objects.
- It is the ability to tell the compiler thow to perform a cerain operation when its corresponding opeator is used on one or more ariables.
- Closely Related to function overloading.
- Allows existing operators to be redefined or overloaded to have new meaning for a specific class objects.
- Already used the + and - operator when you are adding ints and floats and stuff. They have been overloaded to implicitly convert the operands if they are compatible, but not same.
- Overloading of operators are achieved by creating operaotr function.
- An operator function defined the operations that the overloaded operator can perform relative to the class.
- An operator function is created using the keyword Operator.
- Operator functions can be either members or non members of a class.
- Non member operator functions are always friend functions of the class, coz you need to access all the data members. So you gotta use the friend keyword.

Syntax:

```
Returntype classname::Operator OperatorSymbol (Argument list)
{
// function body.
}
```

### Restrictions on Operator Overloading

- Precedence or Associativity of an operator cannot be changed by overloading. So use parenthesis to force order of overloaded operators in an expression.
- C++ doesn't allow new operators to be created.
- Number of operands an operator takes cannot be changed, unary remain unary and stuff.
- Cannot overload the meaning of operators if all arguments are primitive data types.
- So that means no overloading for operators for built-in types
- You can't change how 2 integers are added.
- You can't change ::, ., .\*
- .

## 8.1 Types of Polymorphism

### 1. Early Binding or Compile Time Polymorphism

- Events occur at the compile time, so the compiler while making the exe file, knows what function to call.
- All information needed to call a function is known at compile time.
- It is therefore efficient.
- It is also fast in execution because of this.
- To invoke it you have to overload your functions or operators.

### 2. Dynamic Binding or Runtime Polymorphism

- Events occur at runtime
- The OS Decided what function to call
- The Information needed to call the function is known at runtime. It is flexible.
- Due to all these factors, it's slow.
- To invoke it, you need to use the virtual keyword, or override your function in java.
- You always need to use pointers for this.

## 8.2 Pure Virtual Functions

1. These are called Abstract functions.
2. They don't do anything.
3. You assign them to 0

### **8.3 Abstract Class**

1. Abstract Class is also known as abstract base class.
2. An Abstract class is a class whose instances or objects cant be made.
3. Objects of subclass can be made if they are not abstract.
4. An Abstract class has atleast one abstract function (Pure virtual function )
5. Abstract class can have normal functions and variables along with a pure virtual function.
6. If even one pure virtual function is not overridden, the derived class will also be abstract.
7. Compiler will refuse to create any objects of the class.
8. You cannot call a constructor in this.

### **8.4 Virtual Destructor**

1. Calling the destructor of base class does not destruct the memory of derived class.
2. This problem can be fixed by making the base class destructor virtual.
3. we can ensure that the derived class destructor is called this way.