We are Not Alternate of VU recorded Lectures we are add on for "LEARNING HUNGERS“ Students so that they can apply VU provided concepts in practical as per software industry need

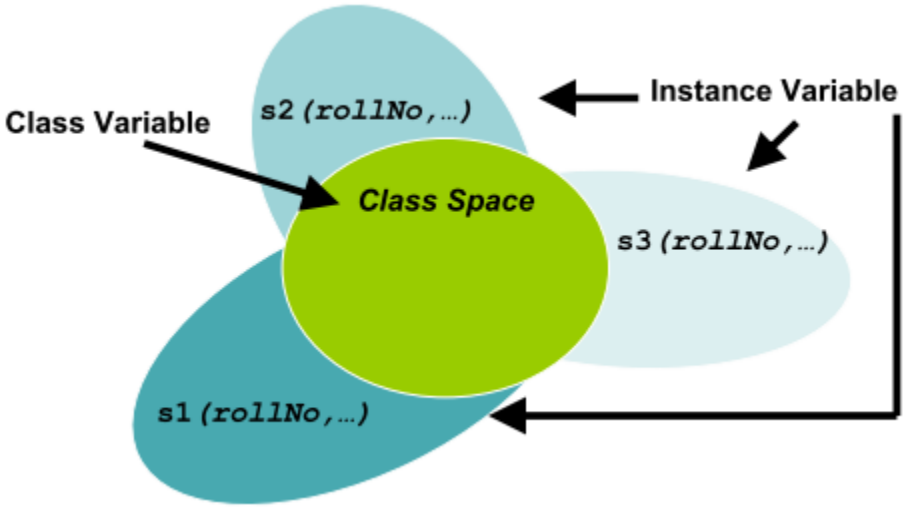
Object Oriented Programming (C++, Java and C#) (CS304)

Objectives

- Static Keyword
- Constructor with Child class and Parent class
- No Default Constructor of Parent Class
- Calling Parent Class Constructor with Child Class
- Member Initialization List
- Base Initialization
- Constant with objects (const keyword)

Static Keyword

- Static variables of a class are such variables which are independent of class objects.
- Lifetime of static variable is throughout the program life, if static variables are not explicitly initialized then they are initialized to 0 of appropriate type.
- Class vs. Instance Variable
  - Memory for static variables in allocated in class space
  - Memory for Instance variables separate for each object



Life of Static Data Member

- They are created even when there is no object of a class
- • They remain in memory even when all Objects of a class are destroyed

C++ Static Keyword Cont…

- Static data member is declared inside the class
- But they are defined outside the class
- Can be accessed through class name and object as well

```
class Student{
private:
static int noOfStudents;
public:
  …
};
int Student::noOfStudents = 0;
```

C# Static Keyword Cont…

- Static data member is declared inside the class
- Can Only be access through Class Name

```
public class Student{
private static int noOfStudents;
public:
  …
}
I
```

Java Static Keyword Cont…

- Static data member is declared inside the class
- Can be accessed through class name and object as well

```
public class Student{
private static int noOfStudents;
public:
  …
}
I
```

C++ Static Member Functions

- The function that needs access to the members of a class, yet does not need to be invoked by a particular object, is called static member function.
  - They are used to access static data members
  - Access mechanism for static member functions is same as that of static data members
  - They cannot access any non-static members

```
class Student{
  static int noOfStudents;
  int rollNo;
public:
  static int getTotalStudent(){
    return noOfStudents;
  }
};
```

```
int main(){
  int i = Student::getTotalStudents();
   Student std ;
Std.getTotalStudents();
          return 0;
}
```

C# Static Member Functions

- The function that needs access to the members of a class, yet does not need to be invoked by a particular object, is called static member function.
  - They are used to access static data members
  - Access mechanism for static member functions is same as that of static data members in C#
  - They cannot access any non-static members

```
class Student{
   public static void Print()
      {

      }
}
```

```
class Test{
    static void Main()
      {
            Employee.Print();
      }
}
```

Java Static Member Functions

- The function that needs access to the members of a class, yet does not need to be invoked by a particular object, is called static member function.
  - They are used to access static data members
  - Access mechanism for static member functions is same as that of static data members in Java
  - They cannot access any non-static members

```
class Student{
   public static void Print()
      {

      }
}
```

```
class Test{
    static void Main()
      {
            Employee.Print();
           Employee emp = new Employee( );
           emp.Print();
      }
}
```

C++ Accessing Non Static

```
class Student{
 private:
  int noOfStudents;
  int rollNo;
public:
            static int getTotalStudents(){
    return noOfStudents;
 /*Error: There is no instance of Student, noOfStudents cannot be accessed*/

  }
};

int main(){
Student st;
  int i = st.getTotalStudents();
            return 0;
}
```

C# Accessing Non Static

```
class Student{
  private int noOfStudents;
  private int rollNo;
public static int getTotalStudents(){
    return noOfStudents;
 /*Error: There is no instance of Student, rollNo cannot be accessed*/

  }
}
static void Main(string[ ] args){
   Student st = new Student();
  int i = st.getTotalStudents();

}
```

Java Accessing Non Static

```
class Student{
  private int noOfStudents;
  private int rollNo;
public static int getTotalStudents(){
    return noOfStudents;
 /*Error: There is no instance of Student, rollNo cannot be accessed*/

  }
}
public static void Main(string[ ] args){
   Student st = new Student();
  int i = st.getTotalStudents();

}
```

“this” Pointer and static member functions

- this pointer is passed implicitly to member functions
- this pointer is not passed to static member functions
- Reason is static member functions cannot access non static data members
- Same behavior in C++, C#, and Java

Objectives

- Static Keyword
- Constructor with Child class and Parent class
- Calling Parent Class Constructor with Child Class
- No Default Constructor of Parent Class
- Member Initialization List
- Base Initialization
- Constant with objects (const keyword)

Constructor with Child class and Parent class

- The anonymous object of base class must be initialized using constructor of base class
- When a derived class object is created the constructor of base class is executed before the constructor of derived class
- If default constructor of base class does not exist then the compiler will try to generate a default constructor for base class and execute it before executing constructor of derived class.
- If the user has given only an overloaded constructor for base class, the compiler will not generate default constructor for base class

Objectives

- Static Keyword
- Constructor with Child class and Parent class
- Calling Parent Class Constructor with Child Class
- No Default Constructor of Parent Class
- Member Initialization List
- Base Initialization
- Constant with objects (const keyword)

C++ : Calling Parent Class Constructor with Child Class

- Parent Constructor can be called from child as below

```
ContractualEmployee():Employee(5){
 cout<<"ContractualEmployee Class Constructor "<<endl;
 }
```

C#: Calling Parent Class Constructor with Child Class

- Parent Constructor can be called from child as below

```
ContractualEmployee():base(5){


 }
```

Java: Calling Parent Class Constructor with Child Class

- Parent Constructor can be called from child as below

```
ContractualEmployee() {
        super(5)  ;
 }
```

Objectives

- Static Keyword
- Constructor with Child class and Parent class
- Calling Parent Class Constructor with Child Class
- No Default Constructor of Parent Class
- Member Initialization List
- Base Initialization
- Constant with objects (const keyword)

C++ No Default Constructor of Parent Class

- It’s a compile time error if No Default Constructor exist in base class and Child class having it.
- In this case you must provide Constructor as Member Initializer

```
ContractualEmployee():Employee(5){
 cout<<"ContractualEmployee Class Constructor "<<endl;
 }
```

C# No Default Constructor of Parent Class

- It’s a compile time error if No Default Constructor exist in base class and Child class having it.
- In this case you must provide Constructor as Member Initializer

```
ContractualEmployee():base(5){


 }
```

Java No Default Constructor of Parent Class

- It’s a compile time error if No Default Constructor exist in base class and Child class having it.
- In this case you must provide Constructor as Member Initializer

```
ContractualEmployee() {
    super(5) ;
 }
```

Objectives

- Static Keyword
- Constructor with Child class and Parent class
- Calling Parent Class Constructor with Child Class
- No Default Constructor of Parent Class
- Member Initialization List
- Base Initialization
- Constant with objects (const keyword)

C++ Member Initialization List

- Member initialization list is used where we cannot modify the state of data members in the member functions of the class including constructor.
  - A member initializer list is a mechanism to initialize data members
  - It is given after closing parenthesis of parameter list of constructor

```
class Student{
  const int rollNo;
  char *name;
  float GPA;
public:
  Student(int aRollNo) : rollNo(aRollNo), name(Null), GPA(0.0){ //
initialization
    …
  }
…
};
```

C# Member Initialization List

- You can use object initializers to initialize type objects in a declarative manner without explicitly invoking a constructor for the type.

```
// Declare a StudentName by using the constructor that has two parameters.
    StudentName student1 = new StudentName("Craig", "Playstead");

// Make the same declaration by using an object initializer and sending
    // arguments for the first and last names. The default constructor is
    // invoked in processing this declaration, not the constructor that has
    // two parameters.
    StudentName student2 = new StudentName
    {
        FirstName = "Craig",
        LastName = "Playstead",
    };
```

Java Member Initialization List

- Doesn’t Support

Objectives

- Static Keyword
- Constructor with Child class and Parent class
- No Default Constructor of Parent Class
- Calling Parent Class Constructor with Child Class
- Member Initialization List
- Base Initialization
- Constant with objects (const keyword)

Base Initialization

- The child can only call constructor of its direct base class to perform its initialization using its constructor initialization list.
- The child cannot call the constructor of any of its indirect base classes to perform their initialization using its constructor initialization list

```
class GrandParent{
  int gpData;
public:
  GrandParent() : gpData(0){...}
  GrandParent(int i) : gpData(i){...}
  void Print() const;
};
class Parent1: public GrandParent{
  int pData;
public:
  Parent1() : GrandParent(), pData(0) {…}
};
class Child1 : public Parent1 {
public:
  Child1() : Parent1()  {...}
  Child1(int i) : GrandParent (i) //Error: Child1 can not call its
indirect base class GrandParent Constructor from its constructor
initialization list.
  {...}
  void Print() const;
};
```

Objectives

- Static Keyword
- Constructor with Child class and Parent class
- No Default Constructor of Parent Class
- Calling Parent Class Constructor with Child Class
- Member Initialization List
- Base Initialization
- Constant with objects (const keyword)

Constant with objects (const keyword)

- const ensures that an object is immutable when accessed through a special kind of pointer called a const-pointer
- There are functions that are meant to be read only
- There must exist a mechanism to detect error if such functions accidentally change the data member
- Keyword **`const`** is placed at the end of the parameter list

Declaration:

```
class ClassName{
 ReturnVal Function() const;
};
```

Definition:

```
ReturnVal ClassName::Function() const{
 …
}
```

Example (const)

```
class Student{
public:
 int getRollNo() const  {
   return rollNo;
 }
};
```

Constant with objects (const keyword)

- Constant member functions cannot modify the state of any object
- They are just ***"read-only"***
- Errors due to typing are also caught at compile time
- Constructors and Destructors cannot be const
- Constructor and destructor are used to modify the object to a well defined state
- Constant member function can't access non-const member functions.

C++ const Example

Change the class Student such that a student is given a roll number when the object is created and cannot be changed afterwards

C++ const Example

```
class Student{
…
 const int rollNo;
public:
 Student(int aNo);
 int getRollNo();
 void setRollNo(int aNo);
…
};
```

C++ const Example Continue….

```
Student::Student(int aRollNo)
{
 rollNo = aRollNo;
 /*error: cannot modify a constant
data member*/
}
```

C++ const Example Continue….

```
void Student::setRollNo(int i)
{
 rollNo = i;
 /*error: cannot modify a constant
data member*/
}
```

C++ const Example Continue….

- **Solution**
  - A member initializer list is a mechanism to initialize data members
  - It is given after closing parenthesis of parameter list of constructor
  - In case of more then one member use comma separated list

C++ const Example Continue….

```
class Student{
  const int rollNo;
  char *name;
  float GPA;
public:
  Student(int aRollNo)
  : rollNo(aRollNo), name(Null), GPA(0.0){
    …
  }
…
};
```

Constant Data Members

- Make all functions that don't change the state of the object constant
- This will enable constant objects to access more member functions