

# Object Oriented Programming (OOP)

## Lecture No. 12

# Review

- ▶ Constant data members
- ▶ Constant objects
- ▶ Static data members

# Static Data Member

## **Definition**

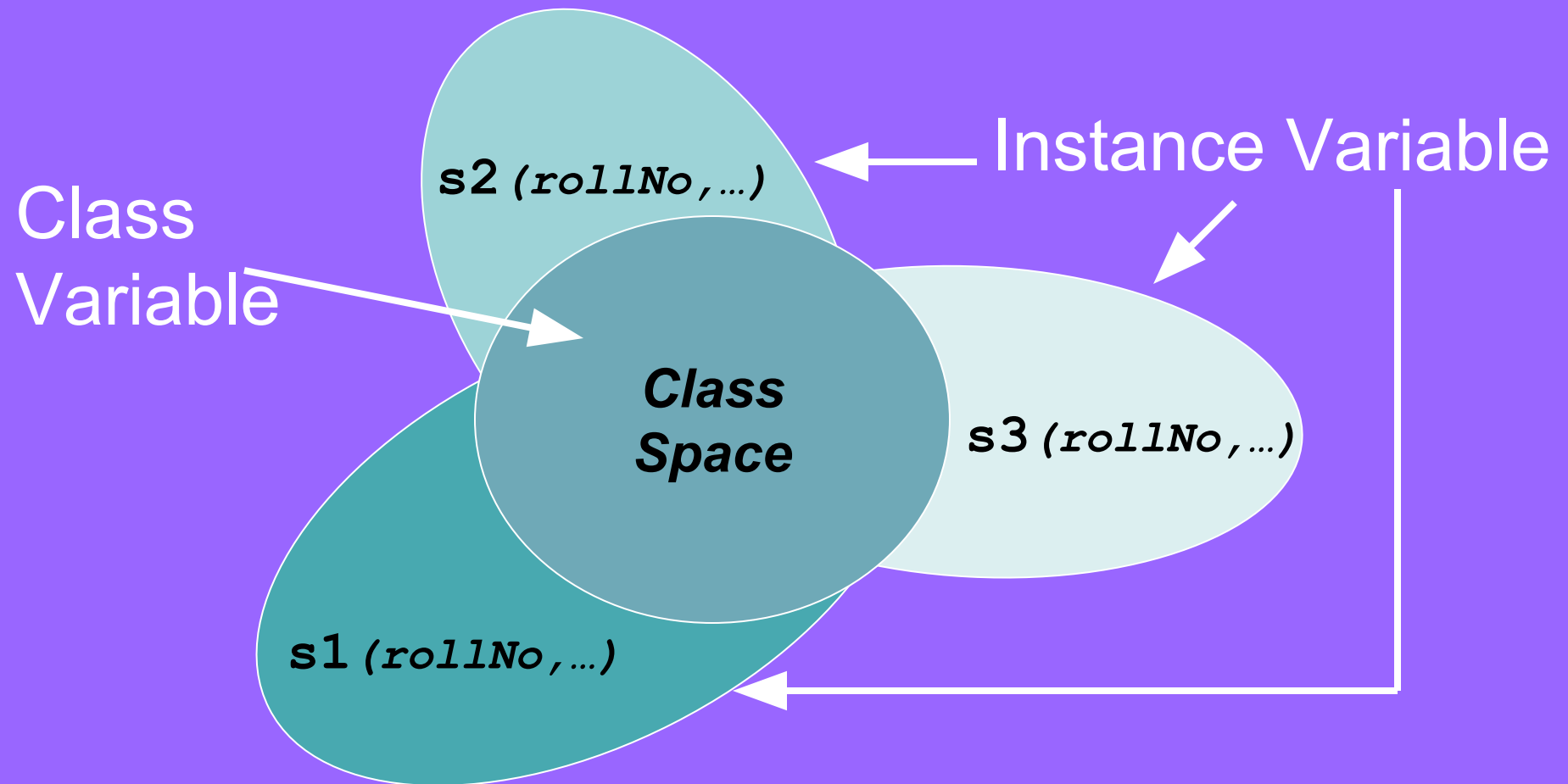
“A variable that is part of a class, yet is not part of an object of that class, is called static data member”

# Static Data Member

- ▶ They are shared by all instances of the class
- ▶ They do not belong to any particular instance of a class

# Class vs. Instance Variable

► `Student s1, s2, s3;`



# Static Data Member (Syntax)

- ▶ Keyword `static` is used to make a data member static

```
class ClassName{  
...  
static DataType VariableName;  
};
```

# Defining Static Data Member

- ▶ Static data member is declared inside the class
- ▶ But they are defined outside the class

# Defining Static Data Member

```
class ClassName{  
...  
static DataType VariableName;  
};  
  
DataType ClassName::VariableName;
```



# Initializing Static Data Member

- ▶ Static data members should be initialized once at file scope
- ▶ They are initialized at the time of definition

# Example

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

int Student::noOfStudents = 0;
/*private static member cannot be accessed outside the
class except for initialization*/
```

# Initializing Static Data Member

- ▶ If static data members are not explicitly initialized at the time of definition then they are initialized to 0

# Example

```
int Student::noOfStudents;
```

is equivalent to

```
int Student::noOfStudents=0;
```

# Accessing Static Data Member

- ▶ To access a static data member there are two ways
  - Access like a normal data member
  - Access using a scope resolution operator '::'

# Example

```
class Student{  
public:  
    static int noOfStudents;  
};  
int Student::noOfStudents;  
int main() {  
    Student aStudent;  
    aStudent.noOfStudents = 1;  
    Student::noOfStudents = 1;  
}
```

# 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

# Example

```
class Student{  
public:  
    static int noOfStudents;  
};  
int Student::noOfStudents;  
int main(){  
    Student::noOfStudents = 1;  
}
```



# Example

```
class Student{
public:
    static int noOfStudents;
};
int Student::noOfStudents;
int main(){
    {
        Student aStudent;
        aStudent.noOfStudents = 1;
    }
    Student::noOfStudents = 1;
}
```

# Uses

- ▶ They can be used to store information that is required by all objects, like global variables

# Example

- ▶ Modify the class Student such that one can know the number of student created in a system

# Example

```
class Student{  
...  
public:  
    static int noOfStudents;  
    Student();  
    ~Student();  
...  
};  
int Student::noOfStudents = 0;
```

# Example

```
Student::Student() {  
    noOfStudents++;  
}  
  
Student::~~Student() {  
    noOfStudents--;  
}
```

# Example

```
int Student::noOfStudents = 0;
int main() {
    cout << Student::noOfStudents << endl;
    Student studentA;
    cout << Student::noOfStudents << endl;
    Student studentB;
    cout << Student::noOfStudents << endl;
}
```

Output:

0

1

2

# Problem

- ▶ noOfStudents is accessible outside the class
- ▶ Bad design as the local data member is kept public

# Static Member Function

## **Definition:**

“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”



# 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

# Example

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

int main(){
    int i = Student::getTotalStudents();
}
```

# Accessing non static data members

```
int Student::getTotalStudents() {  
    return rollNo;  
}  
int main() {  
    int i = Student::getTotalStudents();  
    /*Error: There is no instance of Student,  
    rollNo cannot be accessed*/  
}
```

# this Pointer

- ▶ *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

# Global Variable vs. Static Members

- ▶ Alternative to static member is to use global variable
- ▶ Global variables are accessible to all entities of the program
  - Against information hiding

# Array of Objects

- ▶ Array of objects can only be created if an object can be created without supplying an explicit initializer
- ▶ There must always be a default constructor if we want to create array of objects

# Example

```
class Test{  
public:  
};  
  
int main(){  
    Test array[2]; // OK  
}
```

# Example

```
class Test{  
public:  
    Test();  
};  
int main(){  
    Test array[2]; // OK  
}
```



# Example

```
class Test{  
public:  
    Test(int i);  
};  
int main(){  
    Test array[2]; // Error  
}
```

# Example

```
class Test{  
public:  
    Test(int i);  
}  
int main(){  
    Test array[2] =  
    {Test(0),Test(0)};  
}
```

# Example

```
class Test{  
public:  
    Test(int i);  
}  
int main(){  
    Test a(1),b(2);  
    Test array[2] = {a,b};  
}
```