Object Oriented Programming (OOP)
Lecture No. 11

Engr.Asma Khan
Software Engineering Dept(NED University)

Review

- this Pointer
- Separation of interface and implementation
- Constant member functions

Problem

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

Student Class

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

Modified Student Class

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

Example

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

Example

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

Member Initializer List

- 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

Example

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

Order of Initialization

- Data member are initialized in order they are declared
- Order in member initializer list is not significant at all

Example

```
class ABC{
  int x;
  int y;
  int z;
public:
  ABC();
};
```

Example

```
ABC::ABC():y(10),x(y),z(y)
{
   …
}
/*  x = Junk value
    y = 10
    z = 10 */
```

`const` Objects

- Objects can be declared constant with the use of const keyword
- Constant objects cannot change their state

Example

```
int main()
{
   const Student aStudent;
   return 0;
}
```

Example

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

Example

```
int main(){
   const Student aStudent;
   int a = aStudent.getRollNo();
   //error
}
```

`const` Objects

- `const` objects cannot access "*non const*" member function
- Chances of unintentional modification are eliminated

Example

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

Example

```
int main(){
   const Student aStudent;
   int a = aStudent.getRollNo();
}
```

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

Static Variables

- 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

Example

```
void func1(int i){
   static int staticInt = i;
   cout << staticInt << endl;
}
int main(){
   func1(1);
   func1(2);
}
```

Output:
1
1

Static Data Member

Definition

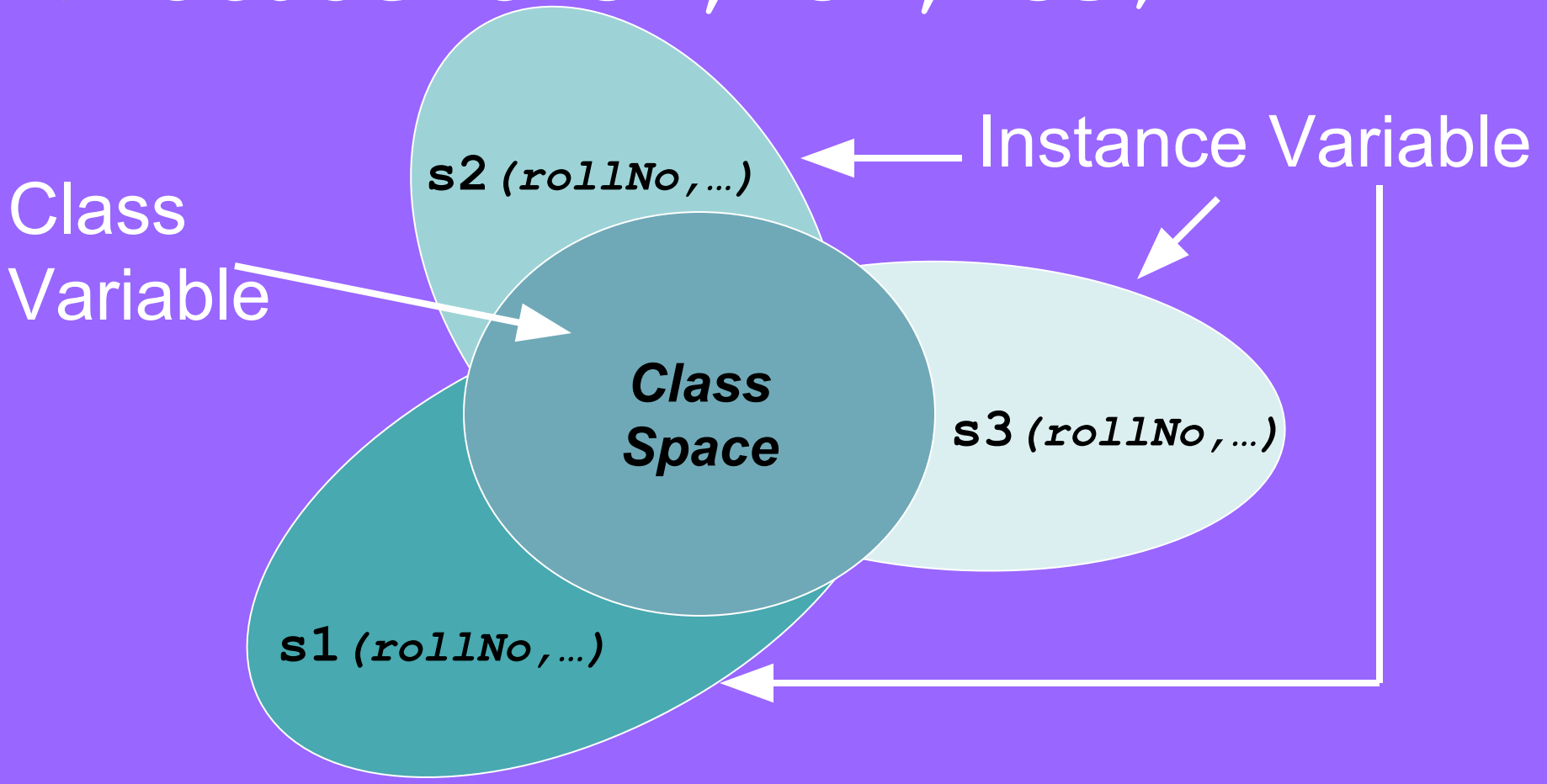
"A variable that is part of a class, yet is not part of an object of that class, is called static data member"

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;
```