Object-Oriented Programming (OOP)
Lecture No. 24

Example

```
class Person{
   char * name;
public:
   Person(char * = NULL);
   const char * GetName() const;
   ~Person();
};
```

Example

```
class Student: public Person{
   char* major;
public:
   Student(char *, char *);
   void Print() const;
   ~Student();
};
```

Example

```
Student::Student(char *_name, char *_maj) :
Person(_name), major(NULL)
{
    if (_maj != NULL) {
        major = new char [strlen(_maj)+1];
        strcpy(major,_maj);
    }
}
```

Example

```
void Student::Print() const{
        cout << "Name: "<< GetName()
                        <<endl;
        cout << "Major: " << major
                                    << endl;
}
```

Example

```
int main(){
    Student sobj1("Ali", "Computer Science");
    {
        Student sobj2 = sobj1;
                //      Student sobj2(sobj1);
        sobj2.Print();
    }
    return 0;
}
```

Example

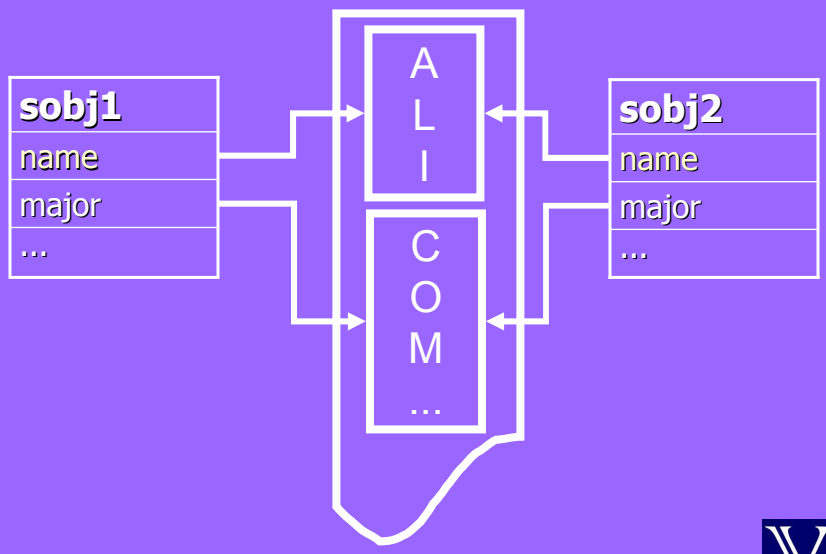
- The output is as follows:

  Name: Ali
  Major: Computer Science

Copy Constructor

- Compiler generates copy constructor for base and derived classes, if needed
- Derived class Copy constructor is invoked which in turn calls the Copy constructor of the base class
- The base part is copied first and then the derived part

Shallow Copy



Example

```
Person::Person(const Person& rhs){
    // Code for deep copy
}

int main(){
    Student sobj1("Ali", "Computer Science");
    Student sobj2 = sobj1;
    sobj2.Print();
    return 0;
}
```

Example

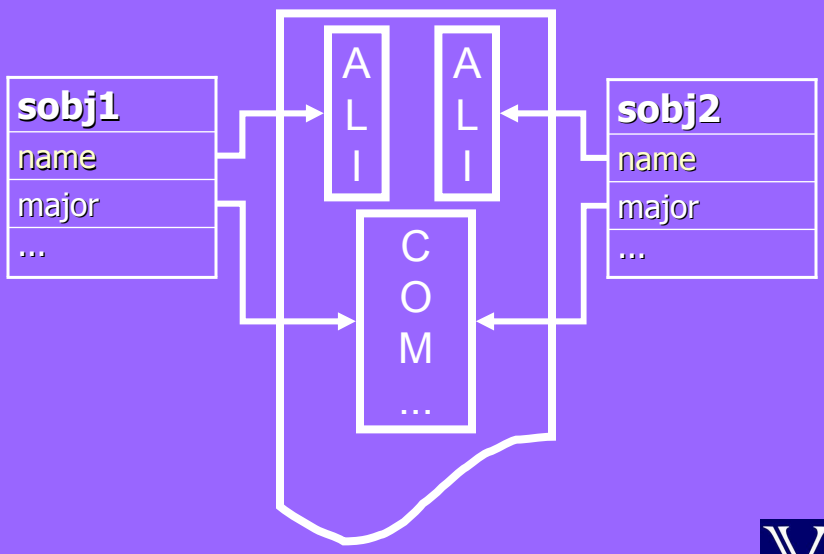
- The output is as follows:

  Name: Ali
  Major: Computer Science

Copy Constructor

- Compiler generates copy constructor for derived class, calls the copy constructor of the base class and then performs the shallow copy of the derived class's data members

Shallow Copy



Example

```
Person::Person(const Person& rhs) {
        // Code for deep copy
}
Student::Student
                        (const Student& rhs) {
        // Code for deep copy
}
```

Example

```
int main(){
   Student sobj1("Ali",
                 "Computer Science");
   Student sobj2 = sobj1;
   sobj2.Print();
   return 0;
}
```

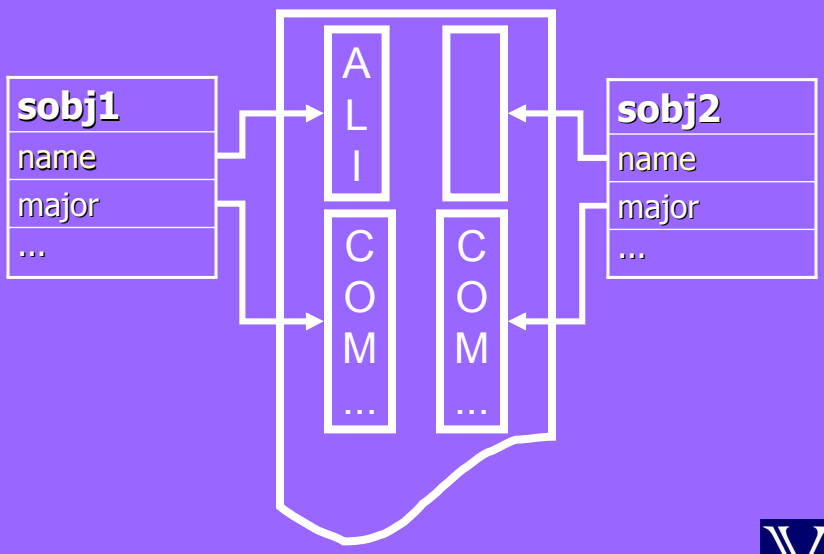
Copy Constructor

- The output will be as follows:

  Name:
  Major: Computer Science

- Name of sobj2 was not copied from sobj1

Copy



Modified Default Constructor

```
Person::Person(char * aName){
   if(aName == NULL)
        cout << "Person Constructor";
   ...
}
int main(){
   Student s ("Ali","Computer Science");
   …
}
```

Copy Constructor

- The output of previous code will be as follows:

```
Person Constructor
Name:
Major: Computer Science
```

Copy Constructor

- Programmer must explicitly call the base class copy constructor from the copy constructor of derived class

Example

```
Person::Person(const Person&
                        prhs) {
   // Code for deep copy
}

Student::Student(const Student
        &srhs) :Person(srhs) {
   // Code for deep copy
}
```

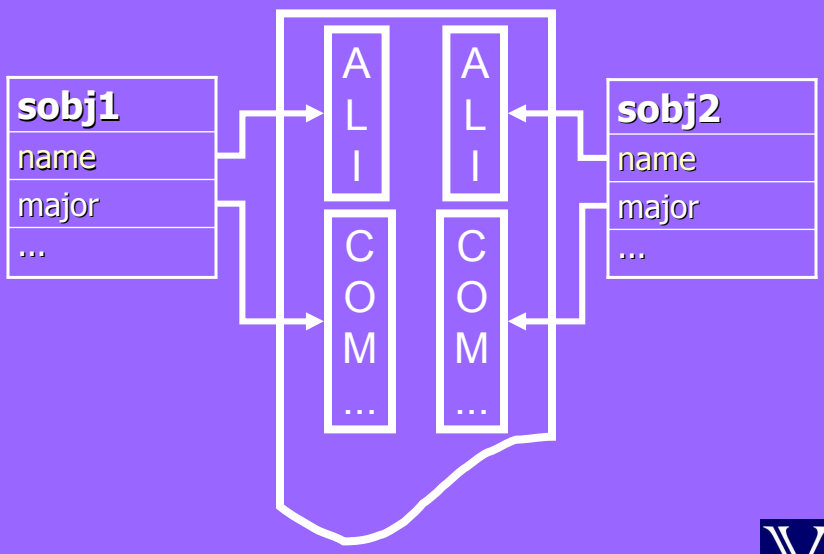
Example

- main function shown previously will give following output

  Name: Ali

  Major: Computer Science

Copy



Copy Constructors

```
3   Person::Person(const Person &rhs) : name(NULL) {
4       //code for deep copy
5   }
    Student::Student(const Student & rhs) :
1       major(NULL),
6   Person(rhs){
2       //code for deep copy
7   }
```

Example

```
int main()
{
   Student sobj1, sboj2("Ali", "CS");
   sobj1 = sobj2;
   return 0;
}
```

Assignment Operator

- Compiler generates copy assignment operator for base and derived classes, if needed
- Derived class copy assignment operator is invoked which in turn calls the assignment operator of the base class
- The base part is assigned first and then the derived part

Assignment Operator

- Programmer has to call operator of base class, if he is writing assignment operator of derived class

Example

```
class Person{
public:
    Person & operator =
                        (const Person & rhs){
        cout << "Person Assignment";
        // Code for deep copy assignment
    }
};
```

Example

```
class Student: Public Person{
public:
    Student & operator = (const Student & rhs){
        cout<< "Student Assignment";
        // Code for deep copy assignment
   }
};
```

Example

```
int main()
{
   Student sobj1, sboj2("Ali", "CS");
   sobj1 = sobj2;
   return 0;
}
```

Example

- The assignment operator of base class is not called
- Output

  Student Assignment

Assignment Operator

- There are two ways of writing assignment operator in derived class
  - Calling assignment operator of base class explicitly
  - Calling assignment operator of base class implicitly

Calling Base Class Member Function

- Base class functions can be explicitly called with reference to base class itself

```
//const char* Person::GetName() {...};
void Student::Print()
{
        cout << GetName();
        cout << Person::GetName();
}
```

Explicitly Calling operator =

```
Person & Person::operator =
                    (const Person & prhs);

Student & Student ::operator =
                    (const Student & srhs){
   Person::operator = (srhs);
   …
   return *this;
}
```

Implicitly Calling operator =

```
Student & Student ::operator =
                        (const Student & srhs) {
   static_cast<Person &>*this=srhs;
   // Person(*this) = srhs;
   // (Person)*this = srhs;
   …
}
```