IIT Bombay

Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Advanced Operations with Inheritance

Recap

- Redefining member functions of the base class
- Access methods of base class using derived classes
- Constructors for derived classes
- Destructors
- Inheritance of assignment operators

Overview of This Lecture

- Objects of base and derived classes
- Objects of classes with pointers and references
- Inheritance
    - Multiple
    - Diamond

Acknowledgment

- Much of this lecture is motivated by the treatment in

  An Introduction to Programming Through C++

  by Abhiram G. Ranade

  McGraw Hill Education 2014

  Some examples used in this lecture are from the above book

Object of base and derived class

IIT Bombay

Object of derived class can be assigned to object of base class

```
class base {
  public:
    int id;
    float balance;
};
```

```
class savings : public base {
  public:
    int age;
    long int ATM;
};
```

Object Slicing
(Loss of Information)

- ‘base’ class has ‘id’ and ‘balance’
- ‘savings’ class has ‘age’ and ‘ATM’, along with ‘id’ and ‘balance’ from base class
- Information of ‘age’ and ‘ATM’ is lost in base

Output

```
1, 15000
2, 67890
```

✓

```
2, 67890
2, 67890
```

```
int main() {
  base b;
  b.id = 1;   b.balance = 15000;

  savings s;
  s.id = 2;   s.balance = 67890;

  cout << b.id << ", " << b.balance << endl;

  cout << s.id << ", " << s.balance << endl;

  b = s;  //Assign object of derived to base

  cout << b.id << ", " << b.balance << endl;

  cout << s.id << ", " << s.balance << endl;

  return 0;
}
```

Object of base and derived class

IIT Bombay

Can we assign object of base class to object of derived class ?

```
class base {
  public:
    int id;
    float balance;
};
```

```
class savings : public base {
  public:
    int age;
    long int ATM;
};
```

Compile time error

```
int main() {
  base b;
  b.id = 1;   b.balance = 15000;

  savings s;
  s.id = 2;   s.balance = 67890;

  cout << b.id << ", " << b.balance << endl;

  cout << s.id << ", " << s.balance << endl;

✗ s = b;  //Assign object of base to derived

  cout << b.id << ", " << b.balance << endl;

  cout << s.id << ", " << s.balance << endl;

  return 0;
}
```

Objects of classes with pointers and references

```
class base {
  public:
    int id;
    float balance;
    void print(){
      cout << "base called" ;
    }
};
```

```
class savings : public base {
  public:
    int age;
    long int ATM;
    void print(){
      cout << "savings called";
    }
};
```

```
int main() {
  base b;  savings s;
  b.id = 1;   b.balance = 15000;
  s.id = 2;   s.balance = 67000;   s.age = 20;   s.ATM = 240;
  b = s;
  cout << b.id << ", " << b.balance ;
  b.id = 3;   b.balance = 30000;
  s.id = 4;   s.balance = 40000;
  //s = b;
  base *bptr;
  savings *sptr;
  bptr = &s;
  cout << bptr->id << ", " << bptr->balance;
  //sptr = &b;
  base& bref = s;
  bref.print();
  return 0;
}
```

derived class can be assigned to object of base class, additional data members are sliced off, only members 'id' and 'balance' will be copied

```
2, 67000
```

error, cannot assign an object of superclass to variable of subclass

assigning subclass object to superclass pointer

```
4, 40000
```

error, cannot assign superclass object to subclass pointer

reference of type superclass to objects of subclass

```
base called
```

calls the 'print' in the base class and not in the savings class

Overloading assignment operator

IIT Bombay

```
class base {
  public:
    int id; float balance;
    base(int x):id(x){ }
    base & operator=(base & a){
       id = a.id;
       cout << "base class operator\n" ;
       return *this;
    }
};
```

```
class savings : public base {
   public:
      int age; long int ATM;
      savings(int x, int y):base(x),age(y) { }
      savings & operator=(base &b) {
         base::operator=(b);
         return *this;
      }
};
```

assignment operator assigning base class to savings class object

```
int main() {
  base b1(10);
  savings s1(11,20), s2(12,30);
  s2 = s1;
  cout << s2.id << "," << s2.age << endl;
  b1 = s1;
  cout << b1.id << endl;
  b1.id = 50;
  s2 = b1;
  cout << s2.id << "," << s2.age << endl;
  return 0;
}
```

11,20

11

50,20

Inheriting assignment operator

IIT Bombay

```
class base {
  public:
    int id; float balance;
    base(int x):id(x){ }
    base & operator=(base & a){
      id = a.id;
      cout << "base class operator\n" ;
      return *this;
    }
};
```

```
class savings : public base {
  public:
    int age; long int ATM;
    savings(int x, int y):base(x),age(y) { }
    using base::operator=;
};
```

```
int main() {
  base b1(10);
  savings s1(11,20), s2(12,30);
  s2 = s1;
  cout << s2.id << "," << s2.age << endl;
  b1 = s1;
  cout << b1.id << endl;
  b1.id = 50;
  s2 = b1;
  cout << s2.id << "," << s2.age << endl;
  return 0;
}
```

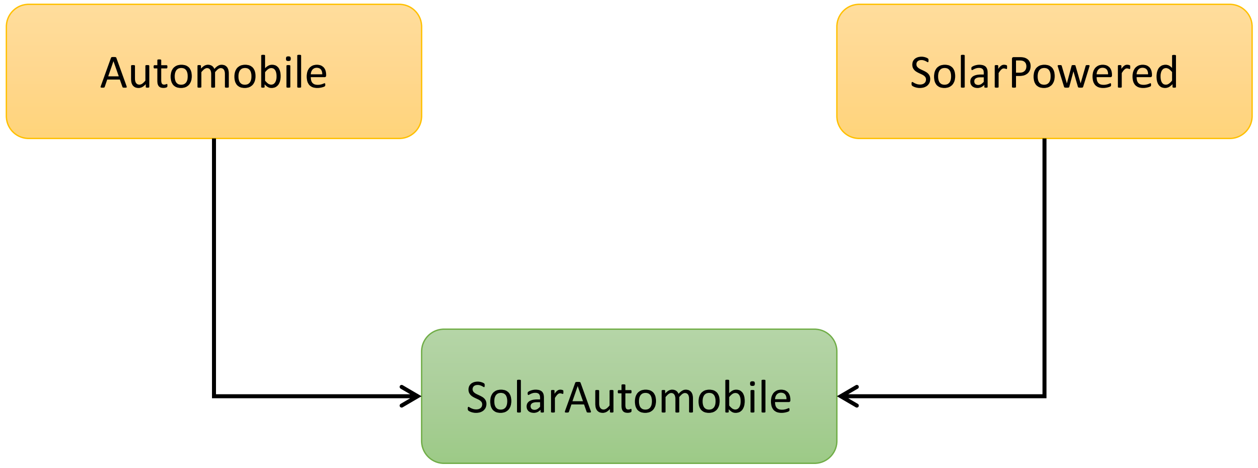
```
11,20
```

```
11
```

```
50,20
```

Multiple Inheritance

```
class Automobile{
    double mileage;
};
```



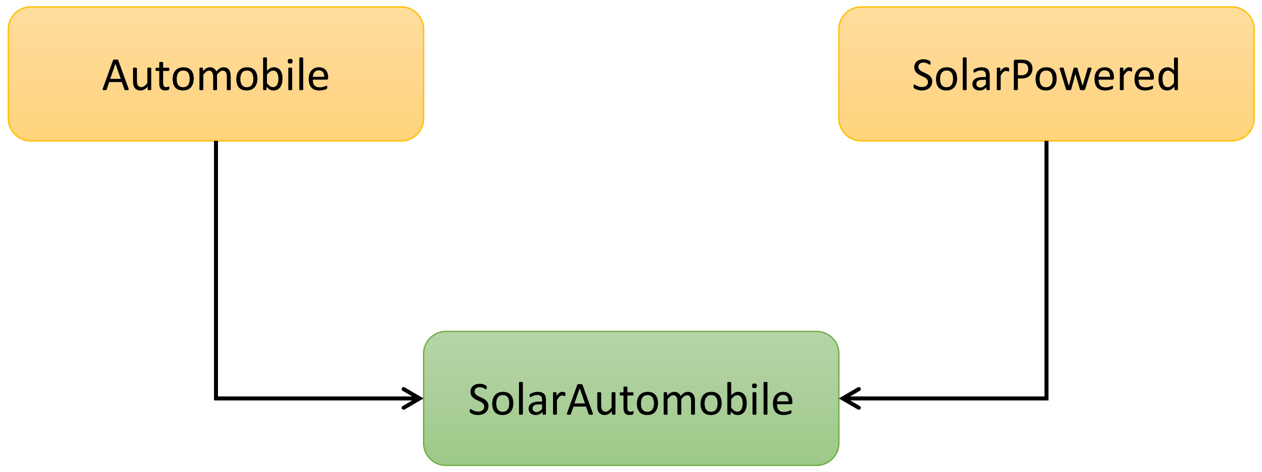
```
class SolarPowered {
    double cellEfficiency;
};
```

```
class SolarAutomobile  : public Automobile, public SolarPowered {
    int noOfSeats;
};
```

Multiple Inheritance

What if there are same member variables/functions in the base classes

```
class Automobile{
    double mileage;
    float price;
};
```



```
class SolarPowered {
    double cellEfficiency;
    float price;
};
```

```
class SolarAutomobile  : public Automobile, public SolarPowered {
    int noOfSeats;
};
```

Multiple Inheritance

IIT Bombay

```
c
```

```
class SolarPowered {
```

```
class Sol
  int no(
  float t
  public:
    Solar
       /
    float
      ret
    }
};
```

```
int main() {
  double mileage = 20.5;
  float automobilePrice = 400000;
  double cellEfficiency = 60.5;
  float solarPrice = 100000;
  int seats = 4;
  SolarAutomobile s(mileage, automobilePrice, cellEfficiency, solarPrice, seats);
  cout << s.getTotalPrice();
  return 0;
}
```

Diamond Inheritance

```
class student{
  int rollNo;
  public :
    student(int a):rollNo(a) { }
    int getRollNo() {
      return rollNo;
    }
};
```

$$\begin{tikzcd}
 & \text{student} & \\
\text{midsem} \arrow[ur] & & \text{endsem} \arrow[ul] \\
 & \text{total} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

```
class endsem: public student{
  float endSemMarks;
  public :
    endsem(int a, float b) :
        student(a), endSemMarks(b) { }
    float getEndSemMarks() {
      return endSemMarks;
    }
};
```

```
class midsem: public student{
  float midSemMarks;
  public :
    midsem(int a, float b) :
       student(a), midSemMarks(b) { }
    float getMidSemMarks() {
      return midSemMarks;
    }
};
```

```
class total : public midsem, public endsem {
  float totalMarks;
  public:
    total(int a, float b, float c) :
      midsem(a,b), endsem(a,c) { }
    float getTotal() {
      return midsem::getMidSemMarks() +
           endsem::getEndSemMarks();
    }
};
```

Diamond Inheritance

```
class student{
  int rollNo;
  public :
    student(int a):rollNo(a) { }
    int getRollNo() {
      return rollNo;
    }
};
```

```
class midsem: public student{
  float midSemMarks;
  public :
    midsem(int a, float b) :
          student(a), midSemMarks(b) { }
    float getMidSemMarks() {
      return midSemMarks;
    }
};
```

```
class endsem: public student{
  float endSemMarks;
  public :
    endsem(int a, float b) :
          student(a), endSemMarks(b) { }
    float getEndSemMarks() {
      return endSemMarks;
    }
};
```

```
class total : public midsem, public endsem {
  float totalMarks;
  public:
    total(int a, float b, float c) :
         midsem(a,b), endsem(a,c) { }
    float getTotal() {
      return midsem::getMidSemMarks() +
           endsem::getEndSemMarks();
    }
};
```

```
int main() {
  int roll = 1001;
  float mMarks = 32.2, eMarks = 43.4;
  total student1(roll, mMarks, eMarks);
  cout << student1.getRollNo();  ✗
  cout << student1.getMidSemMarks();
  cout << student1.getEndSemMarks();
  cout << student1.getTotal() ;
  return 0;
}
```

Call to member function ‘getRollNo’ is ambiguous

Problems

- ‘total’ derives from both, ‘midsem’ and ‘endsem’.
- ‘midsem’ and ‘endsem’ have their own copy of the data members and methods of the student class.
- total object “student1" contains *two* subobjects of ‘*student’* base class.

Virtual Derivation

```
class student{
  int rollNo;
  public :
    student(int a):rollNo(a) { }
    int getRollNo() {
      return rollNo;
    }
};
```

```
class midsem: virtual public student{
  float midSemMarks;
  public :
    midsem(int a, float b) :
          student(a), midSemMarks(b) { }
    float getMidSemMarks() {
      return midSemMarks;
    }
};
```

```
class endsem: virtual public student{
  float endSemMarks;
  public :
    endsem(int a, float b) :
          student(a), endSemMarks(b) { }
    float getEndSemMarks() {
      return endSemMarks;
    }
};
```

```
class total : public midsem, public endsem {
  float totalMarks;
  public:
    total(int a, float b, float c) :
          student(a), midsem(a,b), endsem(a,c) { }
    float getTotal() {
      return midsem::getMidSemMarks() +
             endsem::getEndSemMarks();
    }
};
```

Only one subobject of the ‘student’ class is created for every ‘total’ object.

```
int main() {
  int roll = 1001;
  float mMarks = 32.2, eMarks = 43.4;
  total student1(roll, mMarks, eMarks);
  cout << student1.getRollNo() << endl;
  cout << student1.getMidSemMarks() << endl;
  cout << student1.getEndSemMarks() << endl;
  cout << student1.getTotal() << endl;
  return 0;
}
```

Output

```
1001
32.2
43.4
75.6
```

Summary

- Objects of base and derived classes
- Objects of classes with pointers and references
- Inheritance
  - Multiple
  - Diamond