INHERITANCE

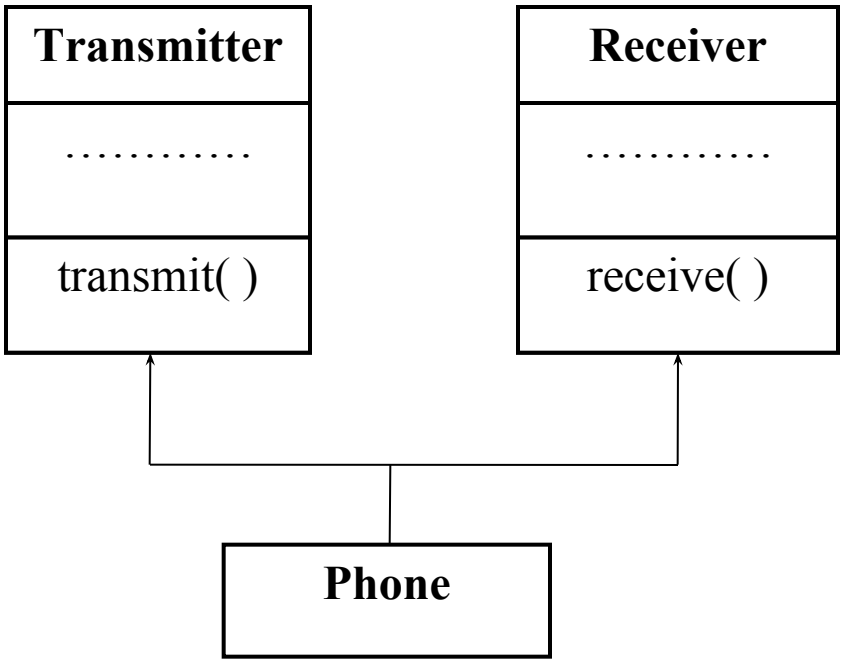
Multi-level and Multiple Inheritance

by Khuram Shahzad

Multiple Inheritance
Introduction

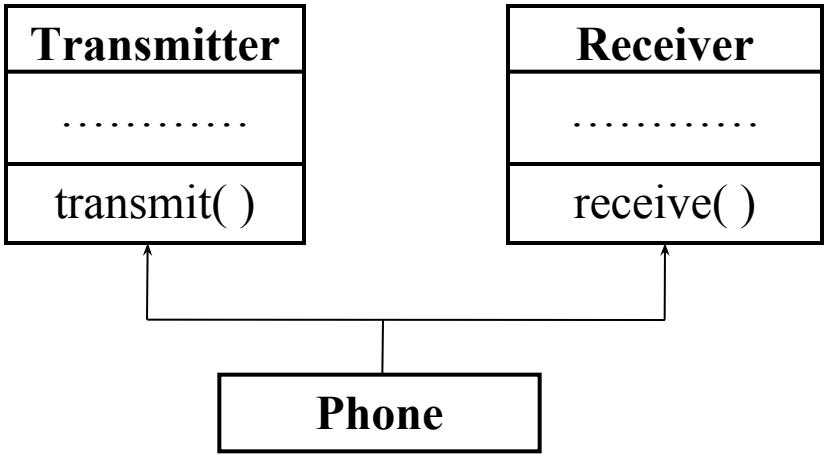
- A class can inherit more than one classes
- Inherited class would have properties of all its base classes
- Base classes kept unchanged by this process

Multiple Inheritance
Example



Here ***Phone*** class is inherited from both ***Transmitter*** and ***Receiver*** classes

Multiple Inheritance
Example Syntax of C++




```
class Transmitter{ ……    };

class Receiver{    ……      };


class Phone : public Transmitter, public Receiver{
    ……
};
```

Multiple Inheritance
Example Program

```
#include <iostream.h>

class Transmitter{
 public:
    void transmit(){cout<<"Transmitting"<<endl;}
};

class Receiver{
 public:
    void receive(){cout<<"Receiving"<<endl;}
};

class Phone : public Transmitter, public Receiver{

};

void main(){
    Phone myPhone;
    myPhone.transmit();// Transmitting
    myPhone.receive(); // Receiving
}
```

Advantages of Multiple Inheritance

- Features of more than one classes can be used into a single class
- Code duplication can be avoided

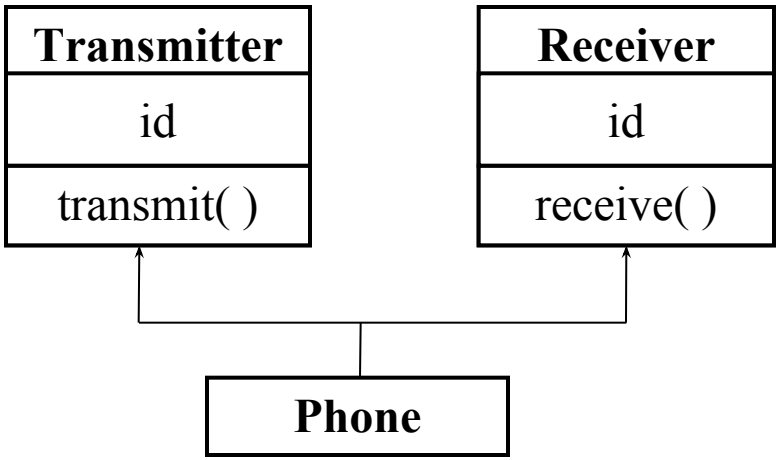
Problems in Multiple Inheritance

- Ambiguity
- Diamond shape problem

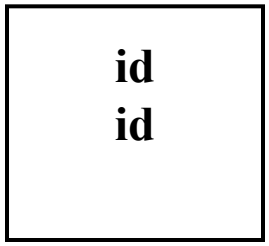
Ambiguity in Multiple Inheritance

If a same name member is coming from more than one base classes then it can create ambiguity when using it in child class.

Ambiguity in Multiple Inheritance
Example




```
Phone myPhone;
```



myPhone;

- **Problems in Multiple Inheritance**

Ambiguity in Multiple Inheritance

Example Program

```
class Transmitter{
 protected:
    int id;
 public:
    void transmit(){cout<<"Transmitting"<<endl;}
};

class Receiver{
 protected:
    int id;
 public:
    void receive(){cout<<"Receiving"<<endl;}
};

class Phone : public Transmitter, public Receiver{
 public:
    void printID(){
        cout<<"Printing ID in phone class: "<<id<<endl;
    }
};
```

Error: id is ambiguous

- **Problems in Multiple Inheritance**

Disambiguation

Solution of Ambiguity in Multiple Inheritance

```
class Transmitter{
 protected:
    int id;
 public:
    void transmit(){cout<<"Transmitting"<<endl;}
};
class Receiver{
 protected:
    int id;
 public:
    void receive(){cout<<"Receiving"<<endl;}
};
class Phone : public Transmitter, public Receiver{
 public:
   void printID(){
     cout<<"Printing ID in phone class: "<<endl
     <<"ID from Transmitter: "<<Transmitter::id<<endl
    <<"ID from Receiver:    "<<Receiver::id<<endl;
  }
};
```

Scope resolution operator also called **Disambiguation operator**

- **Problems in Multiple Inheritance**

Ambiguity in Multiple Inheritance

Example Program 2

```
class Transmitter{
 protected:
    int id;
 public:
    void transmit(){cout<<"Transmitting"<<endl;}
    void printID(){cout<<"Tranmitter ID: "<<id<<endl;}
};
class Receiver{
 protected:
    int id;
 public:
    void receive(){cout<<"Receiving"<<endl;}
    void printID(){cout<<"Receiver ID: "<<id<<endl;}
};
class Phone : public Transmitter, public Receiver{

};
void main(){
    Phone myPhone;
    myPhone.printID();
}
```

Error: printID is ambiguous

- **Problems in Multiple Inheritance**

Disambiguation
Example Program 2

```
class Transmitter{
 protected:
    int id;
 public:
    void transmit(){cout<<"Transmitting"<<endl;}
    void printID(){cout<<"Tranmitter ID: "<<id<<endl;}
};
class Receiver{
 protected:
    int id;
 public:
    void receive(){cout<<"Receiving"<<endl;}
    void printID(){cout<<"Receiver ID: "<<id<<endl;}
};
class Phone : public Transmitter, public Receiver{
};
void main(){
    Phone myPhone;
    myPhone.Transmitter::printID();  //printID( ) of Transmitter class
    myPhone.Receiver::printID();     //printID( ) of Receiver class
}
```

Disambiguation

- **Problems in Multiple Inheritance**