Module 24

Sourangshu
Bhattacharya

Objectives &
Outline

Example –
Phone
Hierarchy

Summary

Module 24: Programming in C++

Inheritance: Part 4 (Example – Phone Hierarchy)

Sourangshu Bhattacharya

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

sourangshu@cse.iitkgp.ac.in

Slides taken from NPTEL course on Programming in C++

by **Prof. Partha Pratim Das**

- Model a hierarchy of phones using inheritance

- ISA Relationship
- Inheritance in C++
  - Semantics
  - Data Members and Object Layout
  - Member Functions
    - Overriding
    - Overloading
  - protected Access
  - Constructor & Destructor
  - Object Lifetime
- Example – Phone Hierarchy
- Inheritance in C++ (private)
  - Implemented-As Semantics

- Let us model a hierarchy of phones comprising:
  - Land line Phone
  - Mobile Phone
  - Smart Phone
- We model Helper classes
- We model each phone separately
- We model the phone hierarchy

Helper Classes

Module 24

Sourangshu
Bhattacharya

Objectives &
Outline

Example –
Phone
Hierarchy

Summary

Class	Description
class PhoneNumber	12-digit phone number
class Name	Subscriber Name (as `string`)
class Photo	Image & Subscriber Name as alt text
class RingTone	Audio & ring tone name
class Contact	`PhoneNumber`, `Name`, and `Photo` (optional) of a contact
class AddressBook	List of contacts

Module 24

Sourangshu
Bhattacharya

Objectives &
Outline

Example –
Phone
Hierarchy

Summary

Land line Phone Model

- Landline Phone
  - Call: By dial / keyboard
  - Answer

```cpp
class LandlinePhone {
    PhoneNumber number_;
    Name subscriber_;
    RingTone rTone_;

public:
    LandlinePhone(const char *num,
        const char *subs);

    void Call(const PhoneNumber *p);

    void Answer();

    friend ostream& operator<<(ostream& os,
        const LandlinePhone& p);
};
```

Module 24

Sourangshu
Bhattacharya

Objectives &
Outline

Example –
Phone
Hierarchy

Summary

Mobile Phone Model

- Mobile Phone
  - Call: By keyboard –
    shows number
    - By Number
    - By Name
  - Answer
  - Redial
  - Set Ring Tone
  - Add Contact
    - Number
    - Name

```cpp
class MobilePhone {
    PhoneNumber number_;
    Name subscriber_;
    RingTone rTone_;
    AddressBook aBook_;
    PhoneNumber *lastDial_;
    void SetLastDialed(const PhoneNumber& p);
    void ShowNumber();

public:
    MobilePhone(const char *num,
        const char *subs);

    void Call(PhoneNumber *p);
    void Call(const Name& n);

    void Answer();

    void ReDial();
    void SetRingTone(RingTone::RINGTONE r);
    void AddContact(const char *num = 0,
        const char *subs = 0);

    friend ostream& operator<<(ostream& os,
        const MobilePhone& p);
};
```

Module 24

Sourangshu
Bhattacharya

Objectives &
Outline

Example –
Phone
Hierarchy

Summary

Smart Phone Model

- Smart Phone
    - Call: By touchscreen – shows number & photo
        - By Number
        - By Name
    - Answer
    - Redial
    - Set Ring Tone
    - Add Contact
        - Number
        - Name
        - Photo

```cpp
class SmartPhone {
    PhoneNumber number_;
    Name subscriber_;
    RingTone rTone_;
    AddressBook aBook_;
    PhoneNumber *lastDial_;
    void SetLastDialed(const PhoneNumber& p);
    void ShowNumber();
    unsigned int size_;
    void DisplayPhoto();

public:
    SmartPhone(const char *num,
        const char *subs);

    void Call(PhoneNumber *p);
    void Call(const Name& n);

    void Answer();

    void ReDial();
    void SetRingTone(RingTone::RINGTONE r);
    void AddContact(const char *num = 0,
        const char *subs = 0);

    friend ostream& operator<<(ostream& os,
        const MobilePhone& p);
};
```

- Landline Phone
  - Call: By dial / keyboard
  - Answer

- Mobile Phone
  - Call: By keyboard –
    shows number
    - By Number
    - By Name
  - Answer
  - Redial
  - Set Ring Tone
  - Add Contact
    - Number
    - Name

- Smart Phone
  - Call: By touchscreen –
    shows number & photo
    - By Number
    - By Name
  - Answer
  - Redial
  - Set Ring Tone
  - Add Contact
    - Number
    - Name
    - Photo

- There exists a substantial overlap between the functionality of the phones
- A mobile phone is more capable than a land line phone and can perform (almost) all its functions
- A smart phone is more capable than a mobile phone and can perform (almost) all its functions
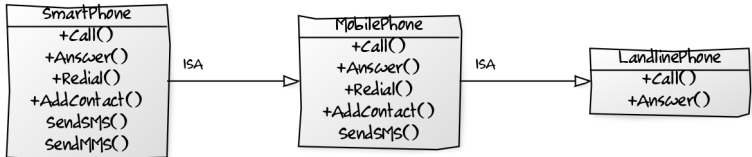- **These phones belong to a Specialization / Generalization hierarchy**

- MobilePhone **ISA** LandlinePhone
    - LandlinePhone is *generalization*
    - MobilePhone is *specialization*
    - MobilePhone inherits the properties of LandlinePhone
- SmartPhone **ISA** MobilePhone
    - MobilePhone is *generalization*
    - SmartPhone is *specialization*
    - SmartPhone inherits the properties of MobilePhone
- **ISA** is *transitive*

```cpp
class LandlinePhone {
    PhoneNumber number_;
    Name subscriber_;
    RingTone rTone_;



public:
    LandlinePhone(const char *num,
        const char *subs);
    void Call(const PhoneNumber *p);


    void Answer();




    friend ostream& operator<<(ostream& os,
        const LandlinePhone& p);
};
```

```cpp
class MobilePhone {
    PhoneNumber number_;
    Name subscriber_;
    RingTone rTone_;
    AddressBook aBook_;
    PhoneNumber *lastDial_;
    void SetLastDialed(const PhoneNumber& p);
    void ShowNumber();
public:
    MobilePhone(const char *num,
        const char *subs);
    void Call(PhoneNumber *p);
    void Call(const Name& n);
    void ReDial();
    void Answer();
    void SetRingTone(RingTone::RINGTONE r);
    void AddContact(const char *num = 0,
        const char *subs = 0);

    friend ostream& operator<<(ostream& os,
        const MobilePhone& p);
};
```
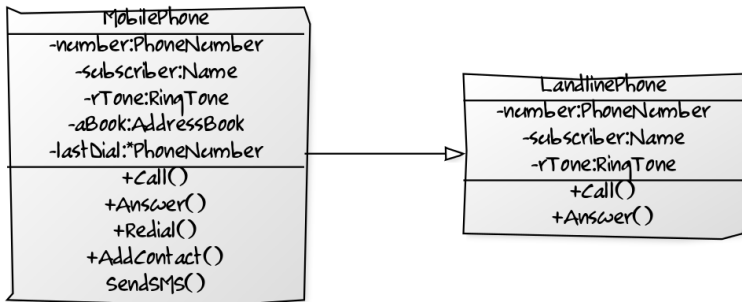
MobilePhone
-number:PhoneNumber
-subscriber:Name
-rTone:RingTone
-aBook:AddressBook
-lastDial:*PhoneNumber
+Call()
+Answer()
+Redial()
+AddContact()
SendSMS()

LandlinePhone
-number:PhoneNumber
-subscriber:Name
-rTone:RingTone
+Call()
+Answer()

Base Class

```cpp
class LandlinePhone {
protected:
    PhoneNumber number_;
    Name subscriber_;
    RingTone rTone_;




public:
    LandlinePhone(const char *num,
        const char *subs) :
        number_(num), subscriber_(subs),
        rTone_() {}

    void Call(const PhoneNumber *p);

    void Answer();





    friend ostream& operator<<(ostream& os,
        const LandlinePhone& p);
};
```

Derived Class

```cpp
class MobilePhone : public LandlinePhone {
protected:
    //PhoneNumber number_;
    //Name subscriber_;
    //RingTone rTone_;
    AddressBook aBook_;
    PhoneNumber *lastDial_;
    void SetLastDialed(const PhoneNumber& p);
    void ShowNumber();
public:
    MobilePhone(const char *num,
        const char *subs) :
        LandlinePhone(num, subs), // Base ctor
        lastDial_(0) {}

    void Call(const PhoneNumber *p); // Override
    void Call(const Name& n);        // Overload
    //void Answer();
    void ReDial();
    void SetRingTone(RingTone::RINGTONE r);
    void AddContact(const char *num = 0,
        const char *subs = 0);

    friend ostream& operator<< (ostream& os,
        const MobilePhone& p);
};
```

Module 24

Sourangshu
Bhattacharya

Objectives &
Outline

Example –
Phone
Hierarchy

Summary

Base Class

```cpp
class MobilePhone : public LandlinePhone {
protected:
    //PhoneNumber number_;
    //Name subscriber_;
    //RingTone rTone_;
    AddressBook aBook_;
    PhoneNumber *lastDial_;
    void SetLastDialed(const PhoneNumber& p);
    void ShowNumber();



public:
    MobilePhone(const char *num,
        const char *subs) :
        LandlinePhone(num, subs), // Base ctor
        lastDial_(0) {}


    void Call(const PhoneNumber *p); // Override
    void Call(const Name& n);       // Overload
    //void Answer();
    void ReDial();
    void SetRingTone(RingTone::RINGTONE r);
    void AddContact(const char *num = 0,
        const char *subs = 0);

    friend ostream& operator<< (ostream& os,
        const MobilePhone& p);
};
```

Derived Class

```cpp
class SmartPhone : public MobilePhone {
protected:
    //PhoneNumber number_;
    //Name subscriber_;
    //RingTone rTone_;
    //AddressBook aBook_;
    //PhoneNumber *lastDial_;
    //void SetLastDialed(const PhoneNumber& p);
    //void ShowNumber();
    unsigned int size_;
    void DisplayPhoto()
public:
    SmartPhone(const char *num,
        const char *subs) :
        MobilePhone(num, subs), // Base ctor
        lastDial_(0) {}


    void Call(const PhoneNumber *p); // Override
    void Call(const Name& n);       // Override
    //void Answer();
    void ReDial();                  // Override
    //void SetRingTone(RingTone::RINGTONE r);
    //void AddContact(const char *num = 0,
        //const char *subs = 0);

    friend ostream& operator<< (ostream& os,
        const SmartPhone& p);
};
```

Module 24

Sourangshu
Bhattacharya

Objectives &
Outline

Example –
Phone
Hierarchy

Summary

Phone Hierarchy

```cpp
class Phone {
public:
    virtual void Call(const PhoneNumber *p)
        = 0;
    virtual void Answer() = 0;
    virtual void ReDial() = 0;
};
class LandlinePhone: public Phone {
    void ReDial()
    { cout << "Not implemented" << endl; }

protected:
    PhoneNumber number_;
    Name subscriber_;
    RingTone rTone_;
public:
    LandlinePhone(const char *num,
        const char *subs) :
        number_(num), subscriber_(subs),
        rTone_() {}
    void Call(const PhoneNumber *p);
    void Answer();
    friend ostream& operator<<(ostream& os,
        const LandlinePhone& p);
};
```

```cpp
class MobilePhone : public LandlinePhone {
protected:
    AddressBook aBook_;
    PhoneNumber *lastDial_;
    void SetLastDialed(const PhoneNumber& p);
    void ShowNumber();
public:
    MobilePhone(const char *num,
        const char *subs) :
        LandlinePhone(num, subs), // Base ctor
        lastDial_(0) {}
    void Call(const PhoneNumber *p); // Override
    void Call(const Name& n);        // Overload
    void ReDial();
    friend ostream& operator<< (ostream& os,
        const MobilePhone& p);
};
class SmartPhone : public MobilePhone {
protected: unsigned int size_;
    void DisplayPhoto()
public:
    SmartPhone(const char *num,
        const char *subs) :
        MobilePhone(num, subs), // Base ctor
        lastDial_(0) {}
    void Call(const PhoneNumber *p); // Override
    void Call(const Name& n);        // Override
    void ReDial();                   // Override
    friend ostream& operator<< (ostream& os,
        const SmartPhone& p);
};
```
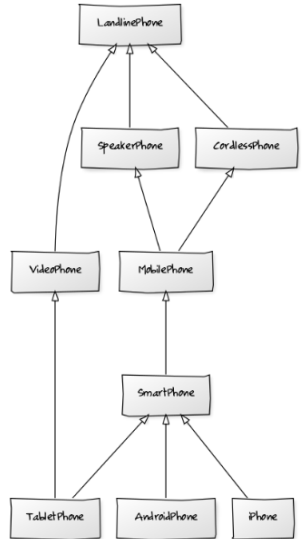
- Using the Phone Hierarchy as an example analyzed the design process with inheritance