Module 24: Programming in C++

Inheritance:

Part 4: Phone Hierarchy

Intructors: Abir Das and Sourangshu Bhattacharya

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

{*abir*, *sourangshu*}*@cse.iitkgp.ac.in*

Slides taken from NPTEL course on Programming in Modern C++

by **Prof. Partha Pratim Das**

Module Objectives

- Model a hierarchy of phones using inheritance

Module Outline

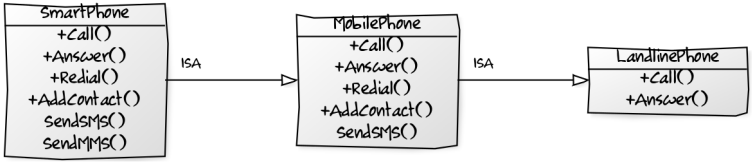
Approach to Modeling Hierarchy

- Identify the **Concepts and their ISA relationships** to define the hierarchy: model with public inheritance
- Identify and model **Helper classes** - lower level UDTs to define components
- Identify the **Interface** of each concept: signatures of public member functions
- Identify the **State Variables** of each concept: types of of private / protected data members (also member functions used for ease of implementation)
- **Refactor** common data members and member functions between specialized and generalized classes to link the classes by inheritance
- **Integrate the hierarchy** with abstract (pure) interface
- Explore **extendability**
- *We illustrate with the phone hierarchy*

Helper Classes

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

Hierarchy of Phones



- `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*

Interfaces of Phones

- **Landline Phone**
  - Call: By dial / keyboard
  - Answer
  - Caller ID (with special attached device)
- **Mobile Phone**
  - Call: By keyboard – shows number
    - By Number
    - By Name
  - Answer
  - Caller ID
  - Redial
  - Set Ring Tone
  - Add Contact
    - Number
    - Name
- **Smart Phone**
  - Call: By touchscreen – shows number & photo
    - By Number
    - By Name
  - Answer
  - Caller ID
  - Redial
  - Set Ring Tone
  - Add Contact
    - Number
    - Name
    - Photo

- There exists a substantial overlap between the functionalities 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**

Interface & State Variable: Landline Phone

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

```
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);
};
```

Interface & State Variable: Mobile Phone

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

```
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);
};
```

Interface & State Variable: Smartphone

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

```
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);
};
```

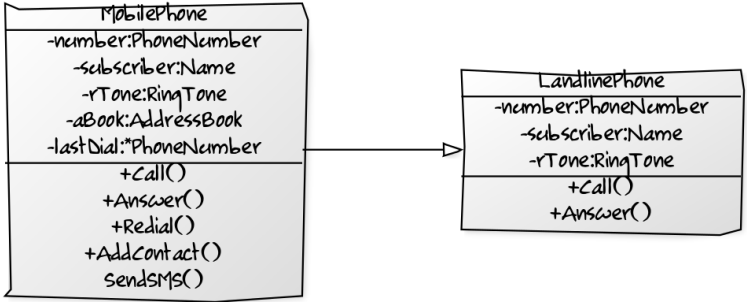
Refactoring

MobilePhone **ISA** LandlinePhone


```
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);
};
```

```
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();                 // Inherited
    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);
};
```

MobilePhone **ISA** LandlinePhone



SmartPhone **ISA** MobilePhone


```
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();                 // Inherited
    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);
};
```

```
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);
};
```

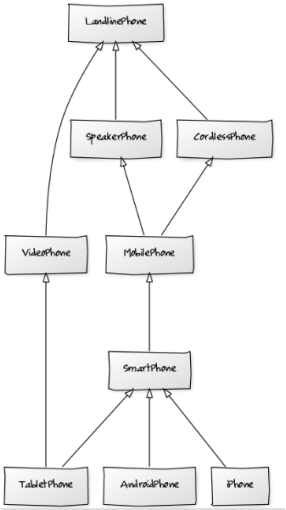
Hierarchy Integration

Hierarchy Integration


```
// Abstract Base Class - A Pure Interface
class Phone { public:
    virtual void Call(const PhoneNumber *p) = 0;
    virtual void Answer() = 0;
    virtual void ReDial() = 0;
};
class LandlinePhone: public Phone {
protected:
    PhoneNumber number_;
    Name subscriber_;
    RingTone rTone_;
public:
    LandlinePhone(const char *num,
        const char *subs) :
        number_(num), subscriber_(subs),
        rTone_() { }
    // Implementations for interfaces
    void Call(const PhoneNumber *p);
    void Answer();
    // Dummy implementation not for use
    void ReDial()
    { cout << "Not implemented" << endl; }
    friend ostream& operator<<(ostream& os,
        const LandlinePhone& p);
};
```

```
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), lastDial_(0) { }
    void Call(const PhoneNumber *p); // Override
    void Call(const Name& n);        // Overload
    void ReDial();                   // Override
    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), 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);
};
```

Extended Hierarchy of Phones



Module Summary

- Using the Phone Hierarchy as an example analyzed the design process with inheritance