Module 24: Programming in C++

Inheritance: Part 4

Partha Pratim Das

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

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan

Module Objectives

- Model a hierarchy of phones using inheritance

Module Outline

- `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

Phone Hierarchy

- 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

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

Land line Phone Model

- 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);
};
```

Mobile Phone Model

- 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);
};
```

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

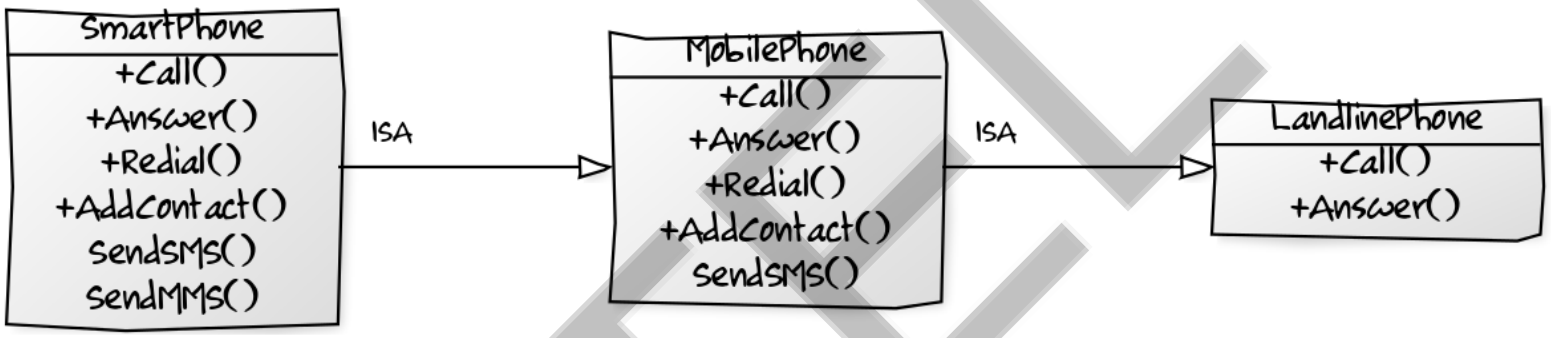
```
class SmartPhone {
    PhoneNumber number_;
    Name subscriber_;
    RingTone rTone_;
    AddressBook aBook_;
    PhoneNumber *lastDial_;
    void SetLastDialed(const PhoneNumber& p);
    void ShowNumber();
    unsigned int size_;
    void DisplayPhoto();

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);
};
```

Comparison of Phones

- 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**

Hierarchy of Phones



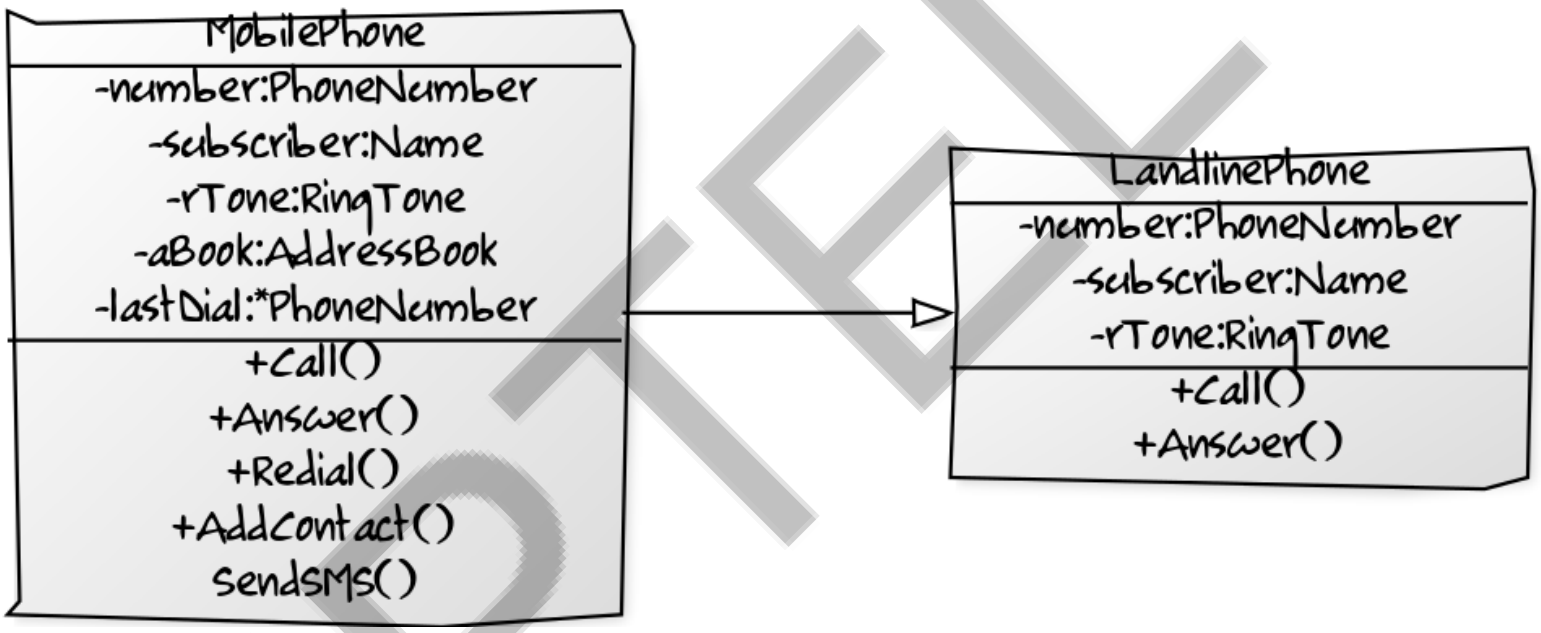
- `MobilePhone` **ISA** `LandlinePhone`
  - `LandlinePhone` is *generalization*
  - `MobilePhone` is *specialization*
  - `MobilePhone` inherits the properties of `LandlinePhone`
- `SmartPhone` **ISA** `MobilePhone`
  - `MobilePhone` is *generalization*
  - `SmartPhone` is *specialization*
  - `SmartPhone`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

Compare LandlinePhone and MobilePhone


```
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);
};
```

```
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);
};
```

Hierarchy of Phones



MobilePhone **ISA** LandlinePhone

Base Class

```
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

```
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);
};
```

SmartPhone **ISA** MobilePhone

Base Class

```
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

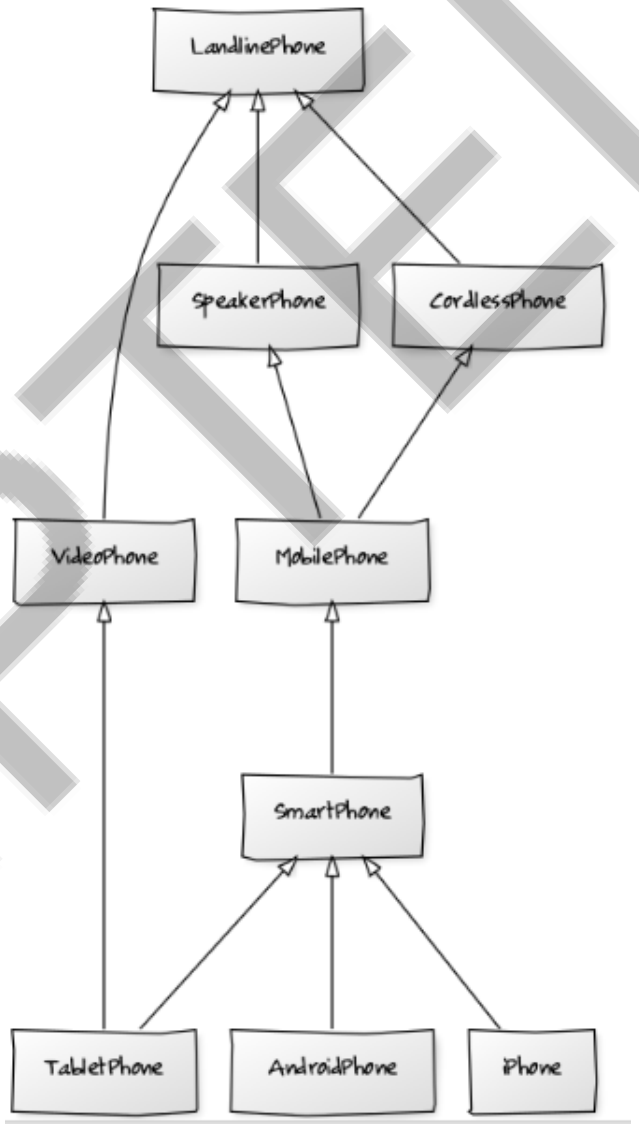
```
class SmartPhone : public MobilePhone {
protected:
    //PhoneNumber number_;
    //Name subscriber_;
    //RingTone rTone_;
    //AddressBook aBook_;
    //PhoneNumber *lastDial_;
    //void SetLastDialed(const PhoneNumber& p);
    //void ShowNumber();
    unsigned int size_;
    void DisplayPhoto()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

Phone Hierarchy


```
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);
};
```

```
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);
};
```

Hierarchy of Phones



Module Summary

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

Instructor and TAs

Name	Mail	Mobile
Partha Pratim Das, *Instructor*	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, *TA*	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, *TA*	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, *TA*	himadribhuyan@gmail.com	9438911655