Module 21: Programming in C++

Inheritance: Part 1 (Inheritance Semantics)

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**

Module Objectives

- Revisit ISA Relationship in OOAD and understand how hierarchy can be created in C++ with Inheritance

Module Outline

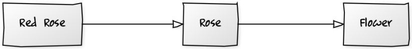
- ISA Relationship
- Inheritance in C++
  - Semantics
  - Data Members and Object Layout
  - Member Functions
    - Overriding
    - Overloading
  - protected Access
  - Constructor & Destructor
  - Object Lifetime
- Example – Phone Hierarchy
- Inheritance in C++ (private)
  - Implemented-As Semantics

ISA Relationship

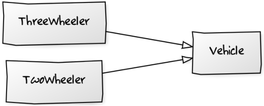
- We often find one object is a *specialization* / *generalization* of another
- OOAD models this using **ISA** relationship
- C++ models **ISA** relationship by *Inheritance* of classes

ISA Relationship

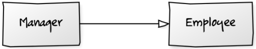
- Rose **ISA** Flower
  - Rose has the properties of Flower – like fragrance, having petals etc.
  - Rose has some additional properties – like rosy fragrance
  - Rose is a *specialization* of Flower
  - Flower is a *generalization* of Rose
- Red Rose **ISA** Rose
  - Red Rose has the properties of Rose – like rosy fragrance etc.
  - Red Rose has some additional properties – like it is red
  - Red Rose is a *specialization* of Rose
  - Rose is a *generalization* of Red Rose



- TwoWheeler **ISA** Vehicle; ThreeWheeler **ISA** Vehicle

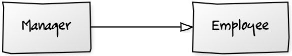


- Manager **ISA** Employee



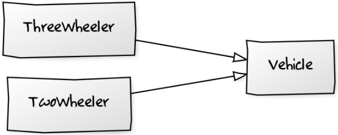
Inheritance in C++: Hierarchy

- Manager **ISA** Employee **[Single Inheritance]**



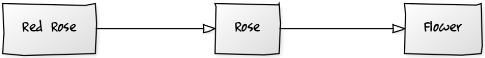
```
class Employee;                   // Base Class = Employee
class Manager: public Employee;  // Derived Class = Manager; Base Class = Employee
```

- TwoWheeler **ISA** Vehicle; ThreeWheeler **ISA** Vehicle **[Hybrid Inheritance]**



```
class Vehicle;                      // Base Class = Vehicle -- Root
class TwoWheeler: public Vehicle;    // Derived Class = TwoWheeler; Base Class = Vehicle
class ThreeWheeler: public Vehicle;  // Derived Class = ThreeWheeler; Base Class = Vehicl
```

- Red Rose **ISA** Rose **ISA** Flower **[Multi-Level Inheritance]**



```
class Flower;                 // Base Class = Flower -- Root
class Rose: public Flower;    // Derived Class = Rose; Base Class = Flower
class RedRose: public Rose;   // Derived Class = RedRose; Base Class = Rose
```

Inheritance in C++: Phones

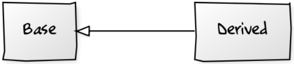
- Landline Phone
  - Call: By dial / keyboard
  - Answer
- Mobile Phone
  - Call: By keyboard – shows number
    - By Number
    - By Name
  - Answer
  - Redial
  - Set Ring Tone
  - Add Contact
    - Number
    - Name

- 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**

Inheritance in C++: Semantics

- Derived **ISA** Base



```
class Base;                      // Base Class = Base
class Derived: public Base; // Derived Class = Derived
```

- Use keyword `public` after class name to denote inheritance
- Name of the Base class follow the keyword

> **"Public inheritance means "is-a." Everything that applies to base classes must also apply to derived classes, because every derived class object is a base class object"**
> – Scott Meyers in Item 32, Effective C++ (3rd. Edition)

Inheritance in C++: Semantics

- Derived **ISA** Base
- Data Members
  - Derived class *inherits* all data members of Base class
  - Derived class may *add* data members of its own
- Member Functions
  - Derived class *inherits* all member functions of Base class
  - Derived class may *override* a member function of Base class by *redefining* it with the same signature
  - Derived class may *overload* a member function of Base class by *redefining* it with the *same name*; but *different signature*
- Access Specification
  - Derived class *cannot access* private members of Base class
  - Derived class *can access* protected members of Base class
- Construction-Destruction
  - A *constructor* of the Derived class *must first* call a *constructor* of the Base class to construct the Base class instance of the Derived class
  - The *destructor* of the Derived class *must* call the *destructor* of the Base class to destruct the Base class instance of the Derived class

Module Summary

- Revisited Hierarchy or `ISA` Relationship in OOAD
- Introduced the Semantics of Inheritance in C++