Inheritance: The Fundamental Functions

Lecture 26

Sections 15.2 - 15.3

Robb T. Koether

Hampden-Sydney College

Wed, Mar 28, 2018

Outline

Inheritance of Constructors

Constructor Rules

1. A derived-class constructor will automatically invoke the base-class default constructor, *unless instructed otherwise*.
2. We may instruct the derived-class constructor to invoke a specific base-class constructor.
3. The base-class constructor is invoked *before* the derived-class constructor is executed.

Invoking the Base Class Constructor

Base Class Constructor

```
Derived-class(parameters) :  Base-class(parameters)
{
    body of the Derived-class constructor
}
```

- We may specify other constructors through an initializer.
- The only control the derived class has over the construction of the base-class object is the choice of base-class constructor.
- I we do not specify the base-class constructor, then the base-class default constructor is used.

Inheritance of Constructors

Example (A List Hierarchy)

- Suppose we create a `List` base class and then derive an `ArrayList` class and a `LinkedList` class from it.

Inheritance of Constructors

Example (A List Hierarchy)

- The `ArrayList` and `LinkedList` classes have `m_size` in common.
- Therefore, we could put `m_size` in the `List` base class.

Inheritance of Constructors

Example (A List Hierarchy)

```
class List
{
    public:
    //  Constructors
        List() {m_size = 0;}
        List(int sz) {m_size = sz;}
    //  Inspectors
        int size() const {return m_size;}
        bool isEmpty() const {return m_size == 0;}

    protected:
    //  Data members
        int m_size;
}
```

Inheritance of Constructors

Example (A List Hierarchy)

```
class ArrayList : public List
{
    public:
    //  Constructors
        ArrayList() {m_element = NULL;}
        ArrayList(int sz, const T& value)  : List(sz)
            {...}
    protected:
    //  Data members
        T* m_element;
};
```

Inheritance of Constructors

Example (A List Hierarchy)

```
class LinkedList : public List
{
    public:
    //  Constructors
        LinkedList() {m_head = NULL;}
        LinkedList(int sz, const T& value) : List(sz)
            {...}
    protected:
    //  Data members
        LinkedListNode* m_head;
}
```

Inheritance of Constructors

Example (A List Hierarchy)

- When we construct an `ArrayList` using

  ```
  ArrayList list(5, 123);
  ```

  what would happen?

Inheritance of Constructors

Example (A List Hierarchy)

- When we construct an `ArrayList` using

  ```
  ArrayList list(5, 123);
  ```

  what would happen?
- The `ArrayList` constructor will *first* call a `List` constructor `List(5)` which will initialize `m_size`.

Inheritance of Constructors

Example (A List Hierarchy)

- When we construct an `ArrayList` using

  ```
  ArrayList list(5, 123);
  ```

  what would happen?
- The `ArrayList` constructor will *first* call a `List` constructor `List(5)` which will initialize `m_size`.
- Then it will initialize `m_element` by allocating memory, etc.

  ```
  m_element = new T[sz];
        :
  ```

Larger Hierarchies

Larger Hierarchies

- What is class `A` is derived from class `B` and class `B` is derived from class `C`?
- How does class `A` determine which class `C` constructor to use?

Larger Hierarchies

Larger Hierarchies

- What is class `A` is derived from class `B` and class `B` is derived from class `C`?
- How does class `A` determine which class `C` constructor to use?
- It doesn't. Class `A` cannot extend control beyond class `B`.

Outline

Inheritance of Destructors

Destructor Rules

1. The derived-class destructor automatically invokes the base-class destructor.
2. The base-class destructor is invoked *after* the derived-class destructor is executed.

Outline

Inheritance of the Assignment Operator

Assignment Operator Rules

- The automatic assignment operator invokes the assignment operator for the base class.
- A programmer-defined assignment operator does not automatically copy the base-class data members.
- A programmer-defined assignment operator must copy the base-class members, or else they won't be copied.

Inheritance of the Assignment Operator

Assignment Operator Rules

- The automatic assignment operator invokes the assignment operator for the base class.
- A programmer-defined assignment operator does not automatically copy the base-class data members.
- A programmer-defined assignment operator must copy the base-class members, or else they won't be copied.
- This is a problem if the base-class members are private.

Inheritance of the Assignment Operator

Assignment Operator Rules

- The automatic assignment operator invokes the assignment operator for the base class.
- A programmer-defined assignment operator does not automatically copy the base-class data members.
- A programmer-defined assignment operator must copy the base-class members, or else they won't be copied.
- This is a problem if the base-class members are private.
- It is a problem even if the base-class members are not private. Why?

Inheritance of the Assignment Operator

Example (Inheritance of the Assignment Operator)

```
class List
{
    public:
        List& operator=(const List& lst);
    protected:
        int m_size;
};

class ArrayList : public List
{
    public:
        ArrayList& operator=(const ArrayList& lst);
    protected:
        T* m_element;
};
```

Inheritance of the Assignment Operator

Example (Inheritance of the Assignment Operator)

```
ArrayList& operator=(const ArrayList& lst)
{
    if (this != &lst)
    {
    //  Clear out the old
        delete[] m_element;
    // Copy the new
        m_size = lst.m_size        // Wrong!
        m_element = new T[m_size];
        for (int i = 0; i < m_size; i++)
            m_element[i] = lst.m_element[i];
    }
    return *this;
}
```

Inheritance of the Assignment Operator

Example (Inheritance of the Assignment Operator)

```
ArrayList& operator=(const ArrayList& lst)
{
    if (this != &lst)
    {
    //  Clear out the old
        delete[] m_element;
    // Copy the new
        List::operator=(lst)       // Right!
        m_element = new T[m_size];
        for (int i = 0; i < m_size; i++)
            m_element[i] = lst.m_element[i];
    }
    return *this;
}
```

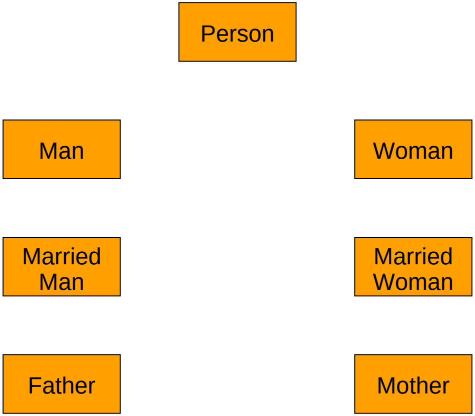
Outline

Example

Example (Inheritance)

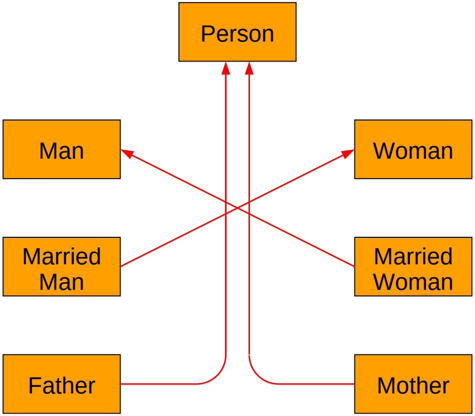
- Create the following classes:
  - Person
  - Man
  - Woman
  - Father
  - Mother
  - MarriedMan
  - MarriedWoman

Example



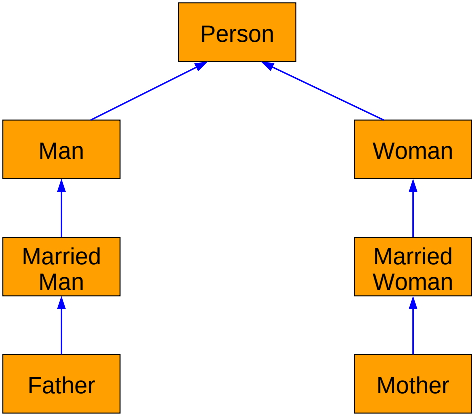
The classes

Example



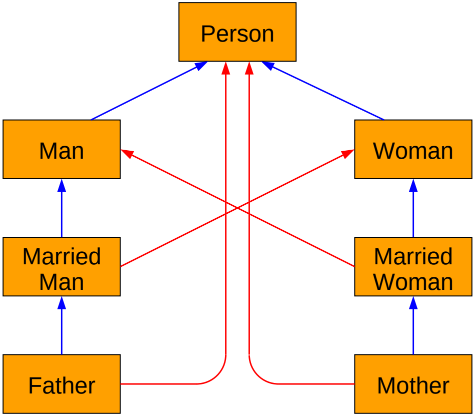
The HAS-A Relation

Example



The ISS-A Relation

Example



The whole shebang

Example

Example (Inheritance)

- `person.h`
- `man.h`
- `woman.h`
- `marriedman.h`
- `marriedwoman.h`
- `father.h`
- `mother.h`

Outline

Assignment

Homework

- Read Section 15.2 - 15.3.