C++: destructor & friend

Spring 2018

Destructor

1. Special member function
2. Name of the destructor is the tilde (~) followed by the class name
3. Destructor is implicitly called when the object is destroyed
4. Default destructor: “empty” destructor
5. Does not release the object’s storage/memory. It only performs housekeeping
6. Destructors are automatically called in the reverse order of constructors:
   – Derived class’ s destructor is called before base class’s destructor

Example of C++ Destructor

```
Class Item{
   private:
         int m_id ;
         string * m_pDesc ;
   public:
        Item(int id, string desc)
        {
              cout << "Constructing an Item object." << endl;
              m_id = id ;
              m_pDesc = new string(desc);
         }

       ~Item()
        {
            cout << "Freeing the obj with description: " << *m_pDesc << endl;
            delete m_pDesc ;
        }
        string toString()
        {
            return (*m_pDesc);
        }
};
```

“friend” functions and “friend” classes

1. Non-member function that has the right to access public and non-public class members

2. Standalone functions, entire classes or member functions of other classes may be declared to be “friends” of another class

3. Example:
   – friend class BankAccount ;
   – friend void deposit(BankAccount acct, double amount);

Example of C++ “friend”

```
Class Item{
   private:
          int m_id ;
          string * m_pDesc ;
   public:
         Item(int id, string desc)
         {
                cout << "Constructing an Item object." << endl;
                m_id = id ;
                m_pDesc = new string(desc);
          }

        ~Item()
         {
             cout << "Freeing the obj with description: " << *m_pDesc << endl;
             delete m_pDesc ;
         }
         string toString()
         {
             return (*m_pDesc);
         }

         friend void print(Item &item) ;
         friend class Book;
};
```

Exercise: how to use C++ destructor & friend

1. Define a “Book” class with only 2 data members:
   1. author and title as “pointer to string”
   2. Dynamically get the strings for author and title

   Make sure it defines the destructor to free up the dynamic storage

2. Define a “friend” function named “hasTheSameAuthor” that returns true if two given books have the same author and false otherwise

3. Write a simple program named “Library” that creates a few books and calls the “hasTheSameAuthor” to compare books.