OBJECT-ORIENTED PROGRAMMING PRINCIPLES
1ST WEEK LECTURE

엄현상(Eom, Hyeonsang)
School of Computer Science and Engineering
Seoul National University

©COPYRIGHTS 2019 EOM, HYEONSANG ALL RIGHTS RESERVED
Outline

- Object Oriented Programming (OOP)
  - Basic Terms
  - Class in OOP
  - C++ Examples
  - C++ Constructor and Destructor
  - Other Stuff (Part of Overview)
  - Summary
  - Some Differences between C & C++
- Q&A
Basic Terms

• Object
  – Collection of Data and Operations on This Data
• Type
  – Characteristics Associated with Objects or Data Elements
• OOP
  – Programming with Objects User-defined types
Class in OOP

• Means to Define Data Types
  – Collection of Members: Data Elements and Operations
    • E.g., Music CD Class
    • Possibly with Access Control
  – Used for Instantiation of Objects

• Means to Realize OOP Concepts
  – Abstraction
  – Encapsulation
  – Inheritance
  – Polymorphism

All Rights Reserved.
C++ Example: Abstraction

- Online Retailer Such as Amazon.Com
  - Item: Type, Title, Maker, Price, Availability, etc.

```cpp
class Item { // Class definition
    public:
        String title; // String is a class defined earlier
        double price; // double is a predefined data type
        double SalePrice() { return (price * 0.9); }
    
    Item A; // Class object definition
    // OKAY: A.title, A.price, and A.SalePrice()
};
```
C++ Example: Encapsulation

• Online Retailer Example Cont’d

class Item { // Class definition
    public:
        String title;
        double price;
        double SalePrice() { return (price*0.9); }
        bool isAvailable() { return (inStockQuantity > 0); }
    private:
        int inStockQuantity;
};
Item A; // Class object definition
// NOT OKAY: A.inStockQuantity
// OKAY: A.isAvailable()
C++ Example: Inheritance

• Online Retailer Example Cont’d

```cpp
class MusicCDItem : public Item {
    public:
    String singer_name;
};
MusicCDItem B; // Class object definition
// OKAY: B.singer_name, B.title, B.price, B.SalePrice(),
// and B.isAvailable()
// NOT OKAY: B.inStockQuantity
```

• Friendship

```cpp
class Item {
    friend class MusicCDItem;
    ...
```

Derivation
- private: public &
  -> private
- protected: public &
  -> protected
class Item { // Class definition
    public:
        String title; // String is a class defined earlier
        double price; // double is a predefined data type
        double SalePrice() { return (price*0.9);}
        int isAvailable() { return (inStockQuantity > 0 ? 1 : 0); } 
        virtual void specificInfo() {
            cout << "no Info: a base-class object" << endl; 
        }
    private:
        int inStockQuantity;
};
C++ Example: Polymorphism Cont’d

• Online Retailer Example Cont’d

```cpp
class MusicCDItem : public Item {
    public:
        String singer_name;
        void specificInfo() { cout << "singer name = " <<
            singer_name
            << " : a derived-class object" << endl; }
    void printSpecificInfo(Item *P) { P->specificInfo(); }
    Item A; // Class object definition
    MusicCDItem B; // Class object definition
    printSpecificInfo(&A); // Call Item::specificInfo()
    printSpecificInfo(&B); // Call MusicCDItem::specificInfo()
    // -Another derived class (e.g., MovieDVDItem) with specificInfo()
};
```

C++ Constructor and Destructor

• Example

```cpp
#include <assert.h>
class String {
public:
    String(const char *s) {
        len = strlen(s);
        str = new char[len + 1];
        assert(str != 0);
        strcpy(str, s);
    }
    ~String() { delete [] str; }

private:
    int len;
    char *str;
};
```

String(int ln) { …}
// Function overloading
// String buf = 1024;

String() { …}
// Default constructor
// String st(); -> Error

String name0 = String("Andrew");
// Definition
String name1("Karl");
String *name_ptr = new String("Thomas");
delete name_ptr;
// Explicit destruction
Other Stuff

- Overloading (w/ Distinguished Argument Lists)
  - Function
    - operator

```cpp
String& String::operator+=(const String &s) {
    len += s.len;
    char *p = new char[len+1];
    assert(p != 0);
    strcpy(p, str);
    strcat(p, s.str);
    delete str;
    str=p;
    return *this;
}
```

String s1("Thank ");
s1 += "you!";
Other Stuff Cont’d

• Reference Type
  – Reference Object to Be Initialized
    • Unable to alias another object once initialized
      
      ```
      int &refVal = val; // int &const refVal = val;
      const int &cir = 1024;
      ```

• Class Template
  – Automatic Generation of Class Instances Bound to a Particular Type

```
template <class SDT>
class Stack { …

Stack<int> s; // typedef int SDT
```
Other Stuff Cont’d

• Multiple Inheritance
  – Child Class as a Composite of Its Multiple Base Classes
    Class C : public A, public B { … }
  – Qualification to resolve ambiguity
• Dominance in the Inheritance Chain
  – Most Derived Instance Dominating
    e.g., C::func() dominates over A::func()
Summary

• Class
  – To Define New Types in OOP
  – To Realize OOP
    • Concepts: Abstraction
    • Encapsulation
    • Inheritance
    • Polymorphism
Some Differences between C & C++

• Type Checking (Regarding Function Declarations)
  – Meaning of No Argument
    • ANSI C: zero or more arguments of any data type
    • C++: no argument
  – Effect of No Declaration
    • ANSI C: permitted
    • C++: error

• C++ Support for Default Arguments
  ```
  void new_line(int n=1) {
    while(n-- > 0) putchar(‘\n’);
  }
  ```

• Dynamic Memory Allocation