C++ Programming

8th Study: Object-Oriented Programming (4/8)

- Member variable:
  (static / const / reference / const reference)
- Member function
  (static / const / default parameter)

C++ Korea 옥찬호 (utilForever@gmail.com)

Member variable

A variable that is associated with a specific object, and accessible for all its methods

Member variable

- C++은 여러 종류의 멤버 변수들을 지원
  - static 멤버 변수
  - const 멤버 변수
  - 레퍼런스 멤버 변수
  - const 레퍼런스 멤버 변수

Member variable – static

- 어떤 경우에는 객체별로 변수를 따로따로 가지는 것이 너무 중복되거나 의도에 맞지 않을 수가 있음
  - 예를 들어, Bank 클래스를 통해 여러 지점 객체들을 생성한 경우 은행의 이자율을 저장하는 멤버 변수가 있을 때, 모든 지점들은 같은 이자율을 가짐
  - 이자율이 바뀔 때, 모든 객체의 멤버 변수를 동기화하는 작업은 매우 비효율적 → C++에서는 static 멤버 변수를 지원!
  - static 멤버 변수는 C에서의 전역 변수와 유사하지만 클래스에 종속된다는 점이 다름
    - class Bank { static double interestRate; }; double Bank::interestRate = 3.5;

Member variable – const

- 클래스의 데이터 멤버를 const로 선언하면, 생성 시점에 초깃값을 부여하고 나면 더는 값을 변경할 수 없게 됨
  - 객체 수준에서 상숫값을 보유하는 것은 대부분 메모리 낭비, 하지만…
  - static const 멤버 변수를 이용해 객체 간에 상숫값을 공유할 수 있음
  - 예를 들어, 은행 공간의 최대 폭 / 높이를 관리해야 된다면…
    - class Bank {
      public:
      static const int kMaxWidth = 300;
      static const int kMaxHeight = 300;
      };

Member variable – reference

- 은행에는 돈을 보관하는 금고가 있음
  - Bank 클래스에서 Vault 클래스를 멤버 변수로 가질 때, Vault 객체를 참조할 것이기 때문에 포인터 또는 레퍼런스형을 사용할 수 있음
  - 이 경우 포인터보다는 레퍼런스형을 사용하는 것이 바람직함
    - 포인터와 달리 레퍼런스 타입은 적합한 객체로 초기화되어야만 존재할 수 있기 때문에 훨씬 안전함 → class Bank { Vault& vault; };
  - 레퍼런스형 멤버 변수는 생성과 동시에 다른 객체를 참조하도록 초기화되어야만 하므로 멤버 이니셜라이저에서 초기화함
    - 생성자 : Bank::Bank(Vault& _vault) : vault(vault) { … }
    - 복사 생성자 : Bank::Bank(const Bank& src) : vault(src.vault) { … }

Member variable – const reference

- 레퍼런스형 멤버 변수는 const 객체를 참조할 수 있음
  - 예를 들어, Bank 객체가 Vault 객체를 const 타입으로 참조해야만 한다면 다음처럼 vault 변수를 const로 선언하면 됨
    - class Bank {
      private:
      const Vault& vault;
      public:
      Bank(const Vault& _vault);
      };
  - const 참조된 객체는 const 멤버 함수만 이용할 수 있음
    - const 참조된 객체에서 const가 아닌 멤버 함수를 호출하면 컴파일 오류 발생

Member variable: Example

```
class Vault
{
private:
    int money;
public:
    Vault(int _money)
        : money(_money)
    {

    }
};
```

```
class Bank
{
private:
    static double interestRate;
    const Vault& vault;
    int width, height;
public:
    static const int kMaxWidth = 300;
    static const int kMaxHeight = 300;

    Bank(const Vault& _vault,
        int _width, int _height);
    Bank(const Bank& src);
};

double Bank::interestRate = 3.5;
```

Member variable: Example

```
Bank::Bank(const Vault& _vault,
    int _width, int _height)
    : vault(_vault), width(_width),
      height(_height)
{

}

Bank::Bank(const Bank& src)
    : vault(src.vault),
      width(src.width),
      height(src.height)
{

}
```

```
int main()
{
    Vault daeguVault(200'000'000);
    Bank daeguBank(daeguVault, 50, 50);
}
```

Member function

A procedure associated with an object class

Member function

- 멤버 변수에 여러 종류가 있듯, 멤버 함수도 여러 종류가 있음
  - static 멤버 함수
  - const 멤버 함수
  - 디폴트 매개변수
- 멤버 함수를 다른 말로 메서드(Method)라고도 함

Member function – static

- 멤버 변수와 마찬가지로 멤버 변수도 특정 클래스의 모든 객체에 공통적으로 적용되어야 할 때가 있음
  - static 멤버 함수의 선언 방법은 static 멤버 변수와 같음
  - 객체에 종속되는 부분이 없다면, static 멤버 함수로 선언 가능
  - 예를 들어, 소숫값을 버리는 함수는 객체에 종속되지 않으므로…
    - static int roundDown(double val);
  - roundDown 함수는 객체에 변화를 주지 않기 때문에 const로 선언할 수 있지만, static 멤버 함수를 선언하려면 const를 빼야 함
    - static 멤버 함수는 객체에 묶이지 않기 때문

Member function – static

- static 멤버 함수
  - 선언하더라도 구현 부분은 바뀌지 않음 (static 붙일 필요 X)
  - static 멤버 함수는 연결된 객체가 없으므로 코드 구현부에서 this 포인터를 이용할 수 없음
    - 단, static 멤버 변수는 이용할 수 있음
  - 같은 클래스의 멤버 함수라면 static 멤버 함수를 보통의 멤버 함수처럼 호출할 수 있음
  - 같은 클래스 내 멤버 함수가 아닌 바깥에서 이 멤버 함수를 호출해야 한다면 스코프 지정 연산자 ::를 이용해야 함
    - 예 : int cuttedInerestRate = Bank::roundDown(3.4);

Member function – const

- 값이 절대로 바뀔 수 없는 객체 : const 객체
  - 객체에 대한 레퍼런스나 포인터 변수를 const로 선언하고 객체의 멤버 함수를 호출하면 해당 멤버 함수가 객체의 멤버 변수 값을 바꾸지 않는다는 보증이 있어야만 정상적으로 컴파일을 진행함
    - 이 때 멤버 함수가 객체의 멤버 변수 값을 바꾸지 않는다는 선언이 바로 const!
      - 예를 들어, class Bank { public: string getBranchName() const; };
    - const 제한자는 멤버 함수 선언에 포함되기 때문에 구현부에서도 똑같이 적용해야 함
      - string Bank::getBranchName() const { return branchName; }

Member function – const

- const 멤버 함수
  - 멤버 함수를 const로 선언하는 것은 멤버 함수 호출 때문에 객체의 멤버 변수 값이 바뀌지 않는다는 것을 보증해 주는 계약과도 같음
    - const로 선언한 멤버 함수 안에서 객체의 멤버 변수를 변경하면 컴파일 오류 발생
  - static 멤버 함수에는 const 제한자를 적용할 수 없음
    - 클래스 공통으로서 연계되는 객체가 없으므로 const 선언이 무의미하기 때문
  - const가 아닌 객체에 대해서는 모든 메서드 호출 가능
  - 하지만 const 객체에 대해서는 const 메서드만 호출 가능
  - 가능하면 객체를 변경하지 않는 모든 멤버 함수에 const 적용 권장
    - const 객체에서도 호출할 수 있게 하는 것이 바람직함

Member function – default parameter

- 디폴트 매개변수는 멤버 함수의 프로토타입을 선언할 때 각 매개변수에 디폴트 값을 지정할 수 있음
  - 만약 사용자가 해당 인자를 직접 제공하면 디폴트 값은 무시됨
  - 만약 해당 인자를 공란으로 해 호출하면 디폴트 값이 자동으로 적용됨
- 디폴트 매개변수는 가장 마지막(오른쪽의) 매개변수부터 시작해 파라미터 건너뜀 없이 연속적으로만 적용할 수 있음
  - 그렇게 하지 않으면 멤버 함수가 호출될 때 어느 매개변수에 디폴트 값을 적용할지 컴파일러가 판단할 수 없음
    - `Bank(string _branchName, int width = kMaxWidth, int height = kMaxHeight);`

Member function – default parameter

- 디폴트 매개변수
  - 매개변수에 디폴트 값을 지정해도 구현부는 바뀌지 않음
  - 디폴트 값은 멤버 함수의 프로토타입을 선언할 때만 지정 가능
    - 구현부 정의에서는 지정할 수 없음
  - 생성자가 디폴트 매개변수를 가졌기 때문에 하나의 생성자만으로 인자가 하나, 둘, 셋인 경우 모두 이용할 수 있음
    - Bank b1("Gangnam");
      Bank b2("Gangnam", 100);
      Bank b3("Gangnam", 100, 200);
    - 디폴트 매개변수로 할 수 있는 일은 멤버 함수 오버로딩으로도 할 수 있음
      (어떤 방법이든 선호하는 대로 이용하면 됨)

Member function: Example

```
class Bank
{
private:
    string branchName;
public:
    static int roundDown(double val);

    string getBranchName() const;
    void setBranchName(string _bName);
};

Bank::Bank(const Vault& _vault,
    int _width = kMaxWidth,
    int _height = kMaxHeight)
    : vault(_vault), width(_width),
        height(_height) { }
```

```
int Bank::roundDown(double val)
{
    return static_cast<int>(val);
}

string Bank::getBranchName() const
{
    return branchName;
}

void Bank::setBranchName(string _bName)
{
    branchName = _bName;
}
```

Member function: Example

```
int main()
{
    Vault daeguVault(200'000'000);

    Bank b1(daeguVault);
    Bank b2(daeguVault, 50);
    Bank b3(daeguVault, 50, 50);

    Bank daeguBank(daeguVault, 50, 50);
    daeguBank.setBranchName("Daegu");
    cout << daeguBank.getBranchName();
    cout << endl;
    cout << Bank::roundDown(3.4);
    cout << endl;
}
```