Programming Methodology
Practice Session #5

Class Declaration

Object–Oriented Programming

- 객체를 정의하고 이들 객체간의 메시지 전달 방법을 통하여 일을 수행하도록 하는 프로그래밍 기법

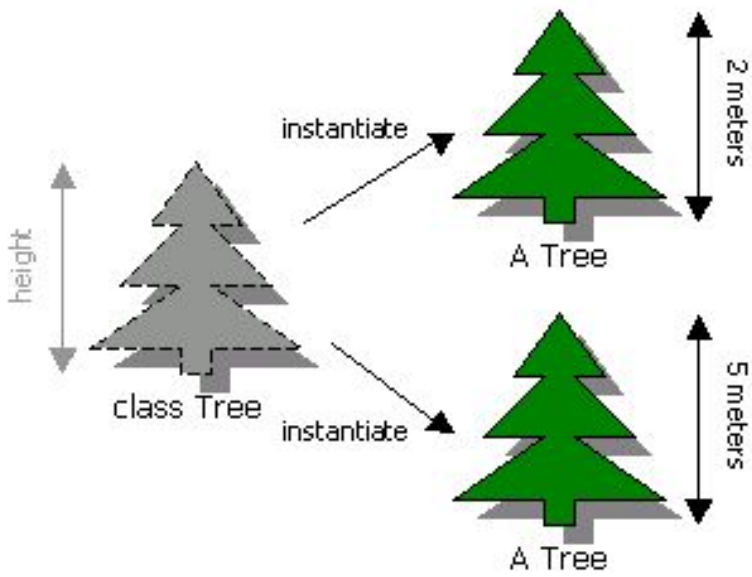
- 실제 세계(real world)를 잘 반영하기 때문에 직관적인 코드 분석을 가능하게 하고, 보다 높은 모듈화가 가능하여 대규모 프로그램을 작성하는 데 유리하다.

- OOP의 3대 특징
  - Encapsulation (Practice Session #6)
  - Inheritance (Practice Session #8)
  - Polymorphism (Practice Session #9)

Class & Object

- 클래스(Class)
  - 같은 종류(또는 문제 해결을 위한)의 집단에 속하는 **속성(attribute)**과 **행위(behavior)**를 정의한 것
  - 속성은 member variable, 행위는 member function으로 표현
  - 객체지향 프로그램의 기본적인 사용자 정의 데이터 타입

- 객체(Object)
  - 클래스의 인스턴스(instance)
  - 클래스가 데이터 타입의 추상적인 표현이라면, 객체는 값을 가지는 실체라고 할 수 있다.



Class Declaration

```
class ExampleClass
{
   private:  // 'private' access specifier
       int privateVar;           // Member Variable
       void privateFunc()        // Member Function
       {
           cout << "This is a Private Function!!" << endl;
       }

   public:  // 'public' access specifier
       int publicVar;            // Member Variable
       void publicFunc()         // Member Function
       {
           cout << "This is a Public Function!!" << endl;
       }
}; // Semicolon(;) is always needed
```

Access Specifier

- 접근 지정자**(Access specifier)**는 member variable이나 member function에 대해 다른 class로부터의 접근 가능성 여부를 지정하기 위해 사용한다.
  - **`public`** – 외부 class에서도 접근 가능
  - **`private`** – class 내부에서만 접근 가능
  - **`protected`** – Practice session #8(Inheritance)에서 설명

Declaration of Member Function

```
class ExampleClass
{
   private:
       int privateVar;
       void privateFunc()        // class 정의 내부에 구현
       {    cout << “This is a private function!!” << endl;    }

   public:
       int publicVar;
       void publicFunc();        // Member function의 선언
};

void ExampleClass::publicFunc()  // class 정의 외부에 구현
{
   privateVar = 1;     // class 내부이므로 private member에 접근 가능
   cout << “This is a public function!!” << endl;
}
```

Use of Objects

```
void main()
{
   // ExampleClass 객체의 생성
   ExampleClass exampleObject1, exampleObject2;

   // ExampleClass 객체의 private member에 접근
   // exampleObject1.privateVar = 3;      // ExampleClass 외부이므로
   // exampleObject1.privateFunc();       // 접근이 불가능하다.

   // ExampleClass 객체의 public member에 접근
   exampleObject1.publicVar = 3;
   exampleObject2.publicVar = 5;
   exampleObject1.publicFunc();
   exampleObject2.publicFunc();
}
```

Sample Practice (1)

- **Fraction** class를 구현한다.
  - Numerator(분자) / Denominator(분모)
  - 약분(reduction) 기능을 구현한다.

- Member variables
  - `int numerator;`
  - `int denominator;`

- Member functions
  - `void setValues(int n, int d);`
  - `void reduce();`
  - `void printFraction();`

Sample Practice (2)

- `void setValues(int n, int d)`
  - numerator, denominator에 각각 n과 d를 할당한다.
- `void reduce()`
  - 해당 Fraction object의 numerator와 denominator를 약분한다.
  - (12 / 18) => (2 / 3)
- `void printFraction()`
  - 해당 Fraction object의 분수값을 출력한다.
  - (*numerator / denominator*) 의 format으로 출력한다.

Sample Practice (3)

```
void main()
{
    Fraction f1;
    f1.setValues(2, 3);
    f1.printFraction();

    Fraction f2;
    f2.setValues(5, 10);
    f2.printFraction();
    f2.reduce();
    f2.printFraction();
}
```

❖ 약분을 위한 GCD(최대공약수)를 구하기 위해 유클리드 호제법 (Euclidean Algorithm)을 사용한다.

```
< 2 / 3 >
< 5 / 10 >
< 1 / 2 >
계속하려면 아무 키나 누르십시오 . . .
```