Programming Methodology
Practice Session #6

Using Class & Data Abstraction

Constructor & Destructor (1)

- Constructor
  - Class의 object가 생성될 때 가장 먼저 호출되는 함수
  - Constructor의 종류
    - Default constructor (아무런 parameter도 가지지 않는다)
    - Parameter를 가지는 constructor
    - Copy constructor

- Destructor
  - Class의 object가 파괴될 때 마지막으로 호출되는 함수

Constructor & Destructor (2)

```
class ExampleClass
{
    public:
        ExampleClass()      // Default constructor, 클래스 정의 내부에 구현
        {   cout << "Default Constructor" << endl;   }
        ExampleClass(int param);                // Parameter를 가지는constructor
        ExampleClass(ExampleClass& source);     // Copy constructor
        ~ExampleClass()                         // Destructor
}

// 클래스 정의 외부에 구현된 constructors & destructor
ExampleClass::ExampleClass(int param)
{   cout << "Constructor with Parameter" << endl;   }

ExampleClass::ExampleClass(ExampleClass& source)
{   cout << "Copy Constructor" << endl;   }

ExampleClass::~ExampleClass()
{   cout << "Destructor" << endl;   }
```

Use of Constructor

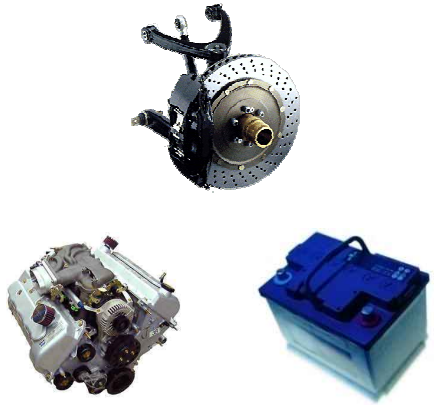
```
void main()
{
    // Default constructor를 이용한 객체 생성
    ExampleClass obj1;

    // int를 paramter로 가지는 constructor를 이용한 객체 생성
    ExampleClass obj2(5);

    // Copy constructor를 이용한 객체 생성
    ExampleClass obj3(obj1);
    ExampleClass obj4 = obj2;

    // 지역변수가 사라지면서 자동으로 destructor가 호출된다.
}
```

Encapsulation

- 문제를 해결하기 위한 data와 method를 하나의 단위로 묶는 방식으로서, 클래스 내부 정의에 대해 외부에서 볼 수 없게 한다.
- Interface를 통해 object의 조작을 가능하게 한다.

Interfaces

Car object



Class Internal

Data Hiding (1)

- Encapsulation의 한 방법으로서, class의 멤버 변수를 private으로 선언하여 외부에서 접근이 불가능하게 하고, 이에 접근하기 위한 방법으로 **public으로 선언된 멤버 함수(getter와 setter)**를 이용한다.

- 객체를 외부의 부정적인 방법(또는 잘못된 방법)으로 사용하는 것을 방지한다.

Data Hiding (2)

```
class ExampleClass
{
    // private으로 선언된 멤버는 다른 객체에서 접근할 수 없다.
    private:
          int data;

    // public으로 선언된 멤버는 다른 객체에서도 접근할 수 있다.
    public:
          int getData() { return data; }                // getter function
          void setData(int value) { data = value; }     // setter function
}

void main()
{
    ExampleClass obj;                  // 객체 생성
    // obj.data = 7;                   // private이기 때문에 error
    obj.setData(7);                    // setter function
    obj.getData(2);                    // getter function
}
```

Sample Practice (1)

- **`Point`** Class를 구현한다.
  - Integer type의 x, y좌표를 멤버 변수로 갖는다.
  - x, y는 **`private`**으로 선언되어야 한다.
- 세 가지의 Constructor를 구현한다.
  - Default Constructor
  - 초기 x, y를 parameter로 받는 constructor
  - **`Point&`** type의 인자를 받아 x, y값을 복사하는 copy constructor를 구현한다.
- Destructor를 구현한다.
- x, y 멤버 변수에 대한 **getter/setter function**을 구현한다.

Sample Practice (2)

- `movX(int), movY(int)` 함수를 구현한다.
  - Parameter 만큼의 값만큼 x 혹은 y 좌표를 이동시킨다.

- `main()` 함수를 구현한다.
  - 1. **Default constructor**를 사용하여 `Point` object `p1`을 생성한 뒤 setter function을 통해 좌표를 다시 설정한다.
  - 2. **Parameter가 있는 constructor**를 사용하여 `Point` object `p2`를 생성한 뒤 getter function과 `mov()` function을 이용하여 `p1`의 좌표만큼 `p2`를 이동한다.
  - 3. **Copy constructor**를 사용하여 `p1`과 같은 좌표를 가지는 `Point` object `p3`를 생성한다.

Sample Practice (3)

- Constructor와 Destructor 각각의 구현부에 해당 function이 호출되었다는 메시지를 출력하도록 한다.

- 각 좌표의 출력은 **getter function**을 이용하도록 한다.

Sample Practice (4)

```
void main( )
{
    cout << "1. Create p1 as (0, 0)" << endl;
    // TODO: Create p1 using default constructor and print it using getter functions.
    cout << "₩tSet p1 = (-2, 4)" << endl;
    // TODO: Set p1=(-2, 4) using setter functions.
    cout << "2. Create p2 as (3, 5)" << endl;
    // TODO: Create p2 using constructor with parameters and print it.
    cout << "₩tAdd p2 to p1..." << endl;
    // TODO: Add p1 and p2 using getter and setter functions. And print p2.
    cout << "3. Create p3 as copy of p1" << endl;
    // TODO: Create p3 using copy constructor.
}
```

Sample Practice (5)

```
1. Create p1 as (0, 0)
        Constructor Point() called...
                p1 = (0, 0)
        Set p1 = (-2, 4)
                p1 = (-2, 4)
2. Create p2 as (3, 5)
        Constructor Point(int, int) called...
                p2 = (3, 5)
        Add p2 to p1...
                p2 = (1, 9)
3. Create p3 as copy of p1
        Copy Constructor called...
                p3 = (-2, 4)
        Destructor called...
        Destructor called...
        Destructor called...
계속하려면 아무 키나 누르십시오 . . .
```