게임 자료구조와 알고리즘

-CHAPTER12-

SOULSEEK

목차

1. 클래스(Class)와 객체(Object)
2. Inline함수와 Const

클래스(CLASS)와 객체(OBJECT)

1. 클래스(CLASS)와 객체(OBJECT)

- 절차지향이라는 말은 잊어버리자 어차피 **C**도 **C++**도 우리가 구상한 순서도에 따라 절차를 거쳐 흘러가고 있다.
- 객체를 생성하고 결과의 흐름속에서도 유연하게 대응할 수 있는 것이라고 생각하자.
- 결국프로그래밍은 쓰여질 곳에서 쓰고자 하는 의도와 목적에 의해 같이 바뀌어야 하는 부분이 발생하기 마련이다.

클래스(Class)

- 다른 여러 자료형의 변수와 함수를 묶을 수 있는 **사용자 정의 자료형**이다.
- 변수 → **Array** → **Struct** → **Class**
- 객체**(Object)**를 설계할 때 필요한 설계도 또는 틀이 된다.
- 자료형 중 하나이다.

객체(Object)

- **Class**의 틀을 참고로 하여 만들어진 결과물이다.

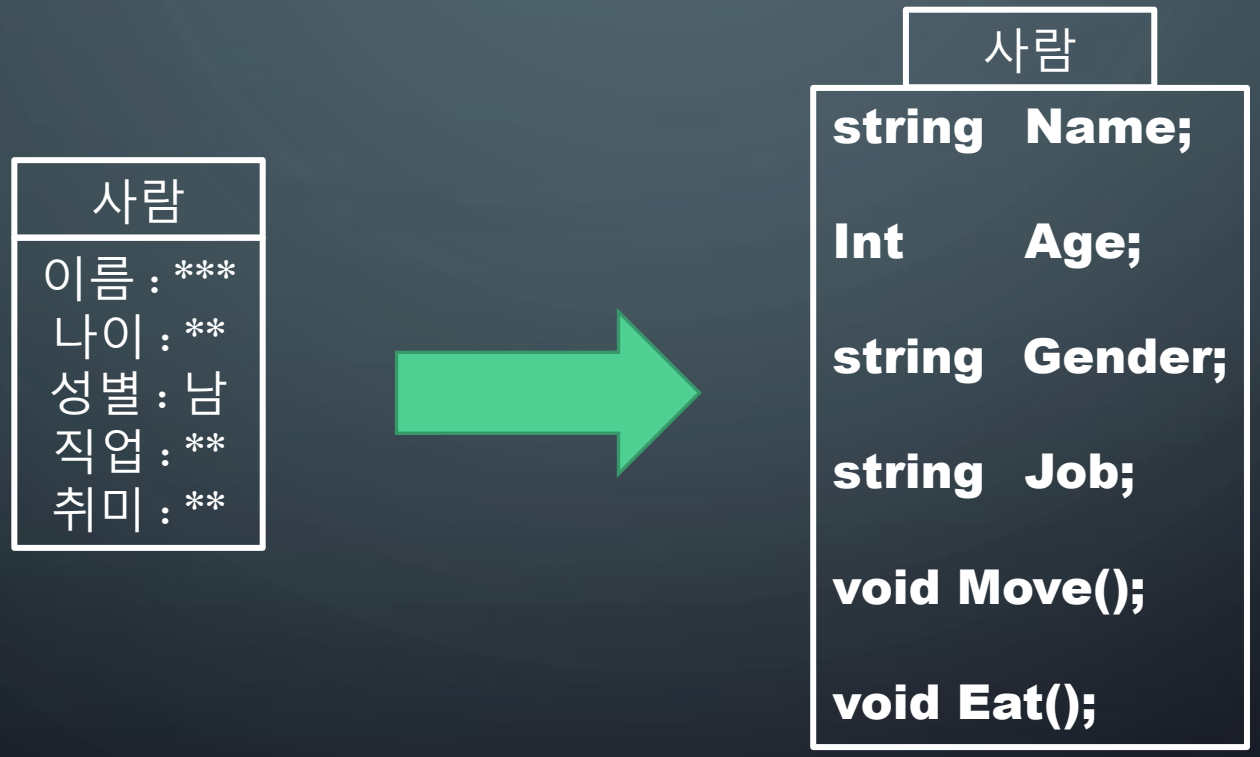
Instance

- 만들어진 **Object**를 개별적으로 정보를 부여하여 만들어진 최종 사용 가능한 상태의 정보

1. 클래스(CLASS)와 객체(OBJECT)

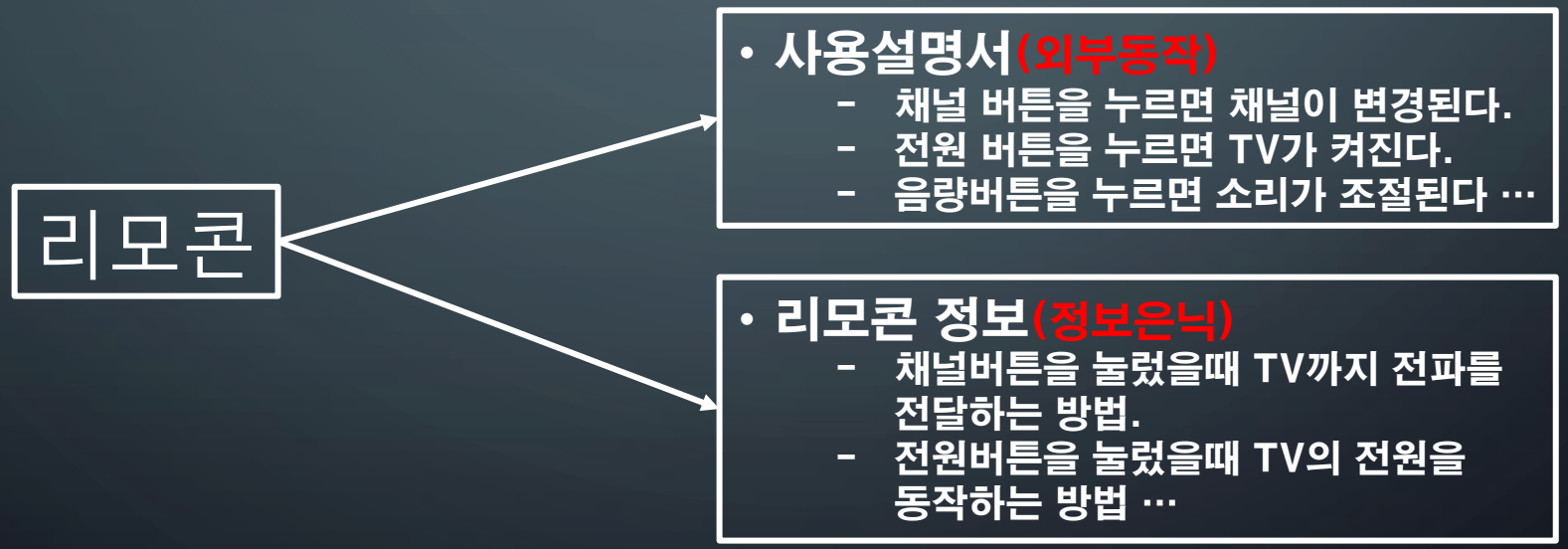
객체 지향의 특징

- **추상화(Abstraction)**
  - 대상의 특성을 제외한 공통점을 모아 정보화해서 구현할 수 있게 대략적인 구현을 시도한다.



1. 클래스(CLASS)와 객체(OBJECT)

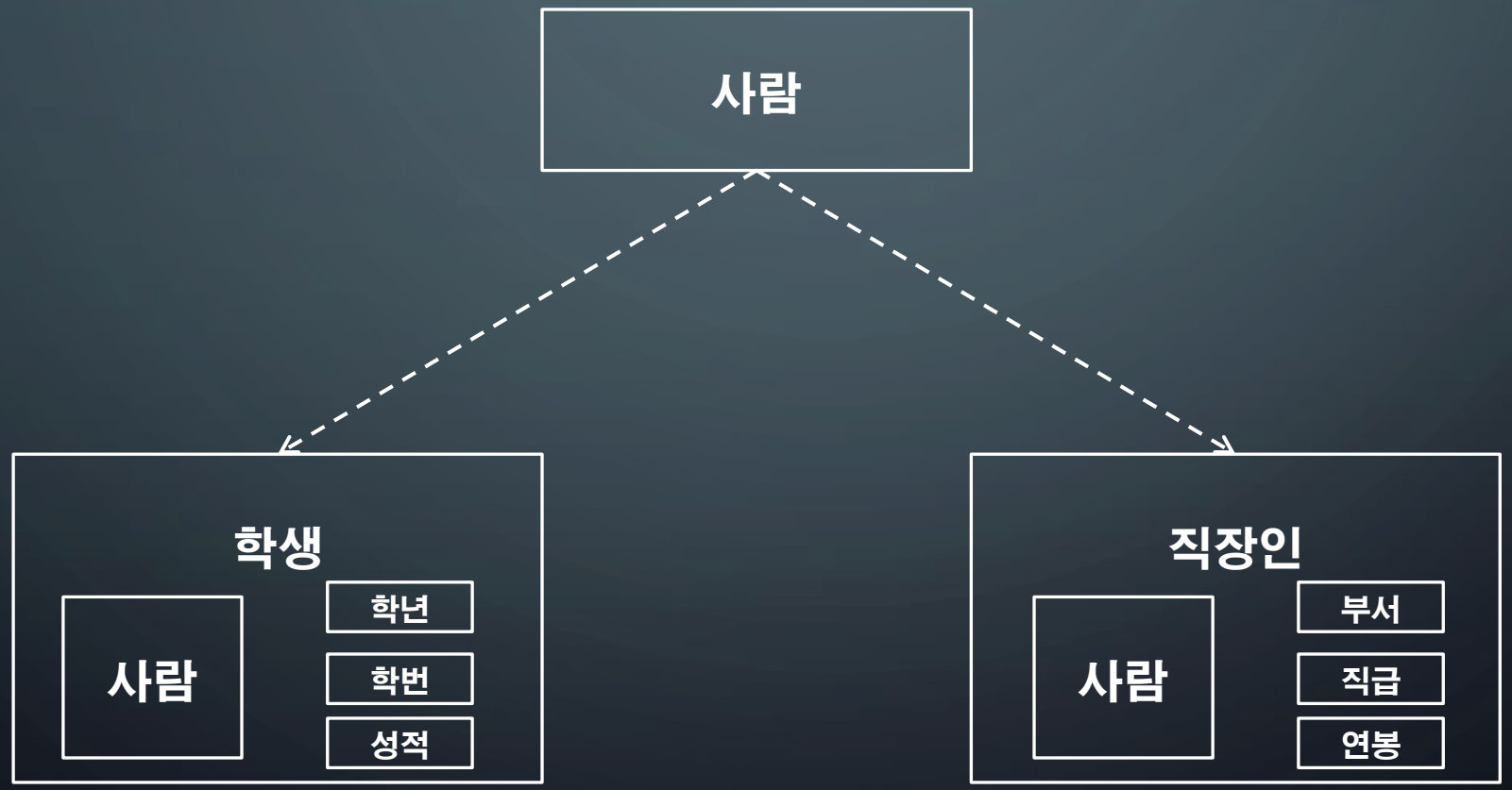
- **캡슐화(encapsulation)**
  - 약속된 부분을 제외한 자신의 정보를 숨긴다.**(private, public)**
  - 외부에서 사용하는 것을 허락하지 않고 결정해준 것에 대한 결과만 활용하는 것
  - 변수는 **private,** 함수는 **publi**으로 정해진 것은 아니다**.**



1. 클래스(CLASS)와 객체(OBJECT)

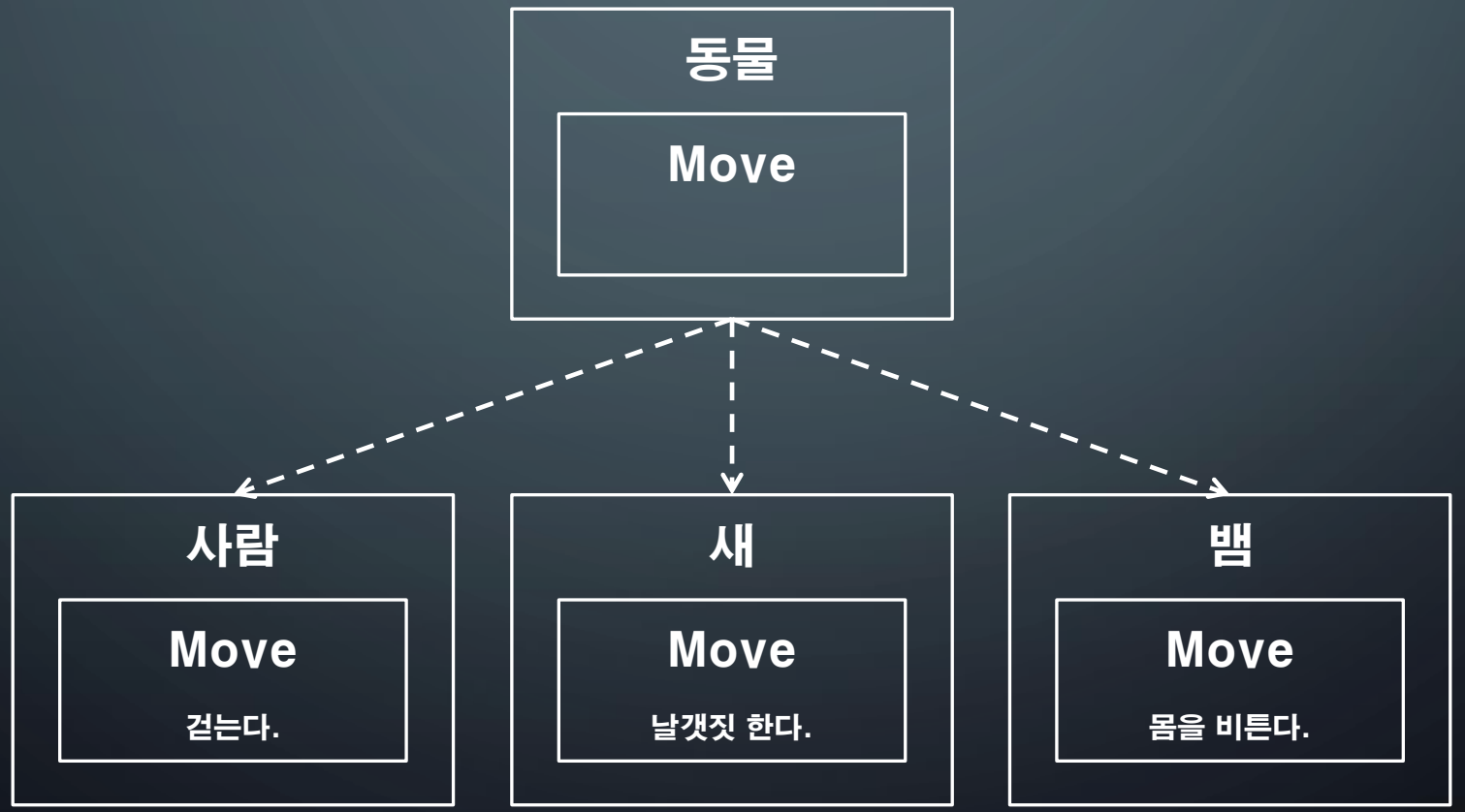
상속(inheritance)

- 이전에 만들어진 class의 기능을 가져와 **재활용**한다.



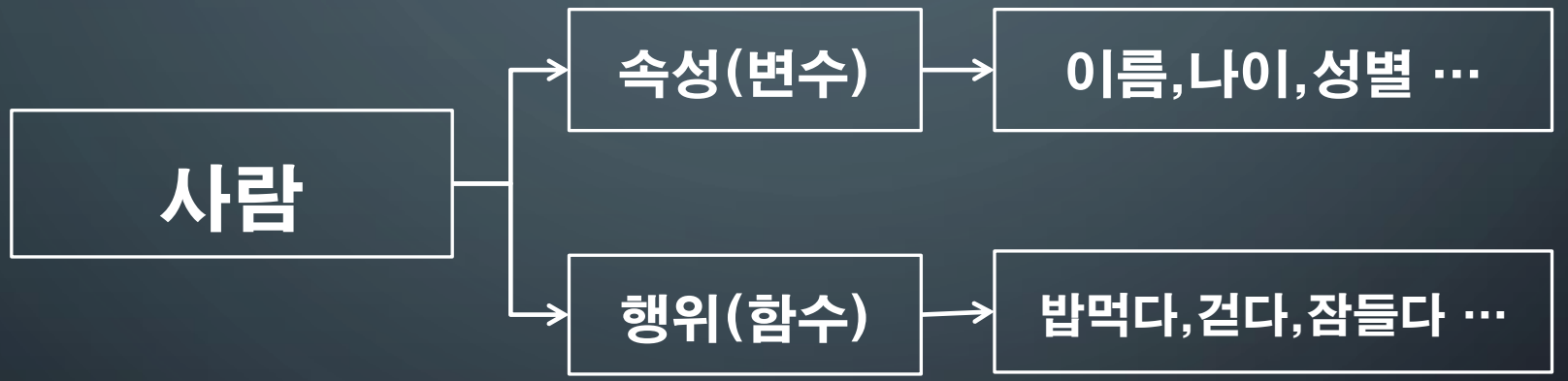
1. 클래스(CLASS)와 객체(OBJECT)

- **다형성(polymorphism)**
  - 약속된 부분만 유지한다면 얼마든지 **접근방식**을 바꿀 수 있다.
    (오버로딩,오버라이딩,가상함수)



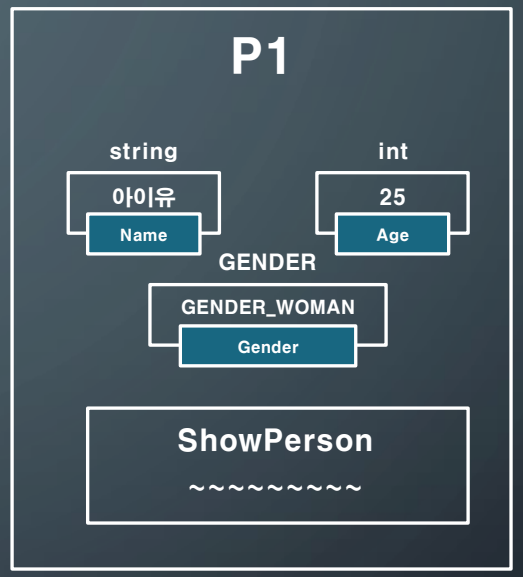
1. 클래스(CLASS)와 객체(OBJECT)

클래스(Class) 생성법



1. 클래스(CLASS)와 객체(OBJECT)

```
#include<iostream>
#include<string>
using namespace std;

enum GENDER
{
	GENDER_MAN,
	GENDER_WOMAN
};

class Person
{
public:
	string Name;
	int Age;
	GENDER Gender;
	void ShowPerson()
	{
		cout << "이름 : " << Name << endl;
		cout << "나이 : " << Age << endl;
		cout << "성별 : ";
		switch (Gender)
		{
		case GENDER_MAN:
			cout << "남자" << endl;
			break;
		case GENDER_WOMAN:
			cout << "여자" << endl;
			break;
		}
	}
};
void main()
{
	Person P1 = { "아이유",25,GENDER_WOMAN };
	cout << "======" << P1.Name << "의 정보"  << "======" << endl;
	P1.ShowPerson();
}
```



1. 클래스(CLASS)와 객체(OBJECT)

Class 접근범위

- **public**
  - 맴버 변수,함수를 어디서든 접근이 가능하다
- **private**
  - 맴버 변수,함수를 자신 **Class**내부에서만 접근이 가능하다
- **protected**
  - 맴버 변수,함수를 자신 **Class**내부와 상속받은 자식 **Class**내부에서만 접근이 가능하다.

1. 클래스(CLASS)와 객체(OBJECT)

```
#include<iostream>
#include<string>
using namespace std;

enum GENDER
{
	GENDER_MAN,
	GENDER_WOMAN
};

class Person
{
private:
	string Name;
	int Age;
	GENDER Gender;
public:
	void SetName(string _Name, int _Age, GENDER _Gender)
	{
		Name = _Name;
		Age = _Age;
		Gender = _Gender;
	}
	....
	string GetName()
	{
		return Name;
	}
	void ShowPerson()
	{
		cout << "이름 : " << Name << endl;
		cout << "나이 : " << Age << endl;
		cout << "성별 : ";
		switch (Gender)
		{
		case GENDER_MAN:
			cout << "남자" << endl;
			break;
		case GENDER_WOMAN:
			cout << "여자" << endl;
			break;
		}
	}
};
void main()
{
	Person P1;
	P1.SetName("아이유", 25, GENDER_WOMAN);
	cout << "======" << P1.GetName();
	cout   << "의 정보" << "======" << endl;
	P1.ShowPerson();
}
```

정보의 은닉

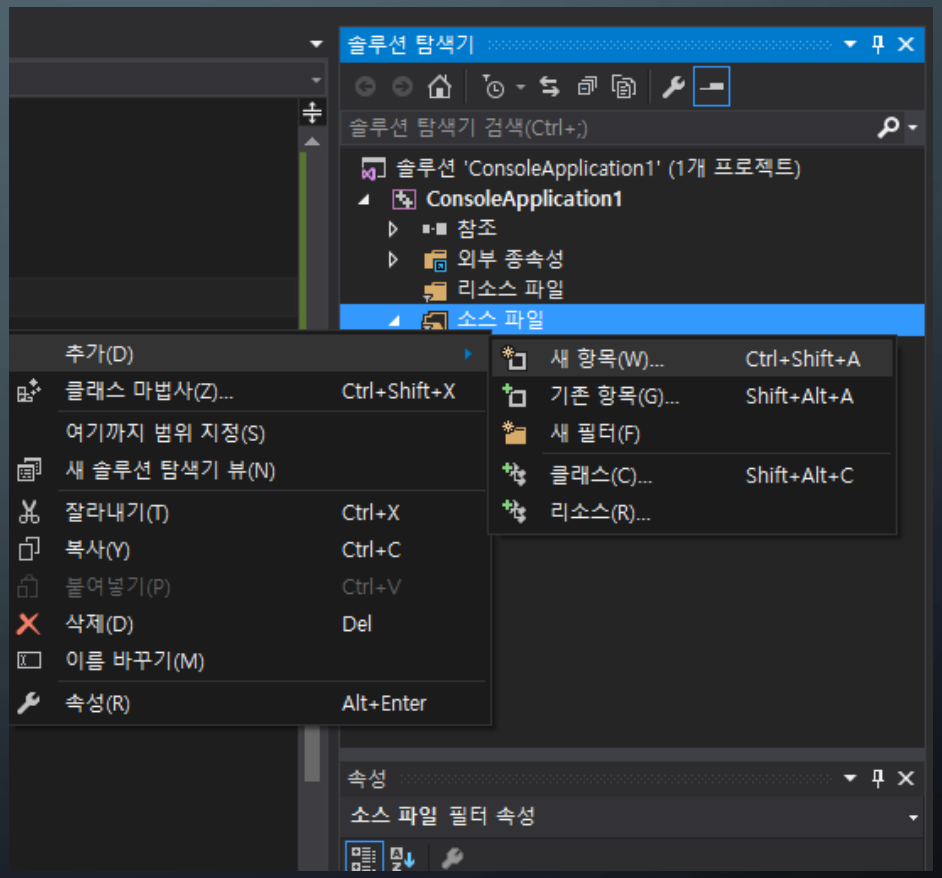
캡슐화

1. 클래스(CLASS)와 객체(OBJECT)

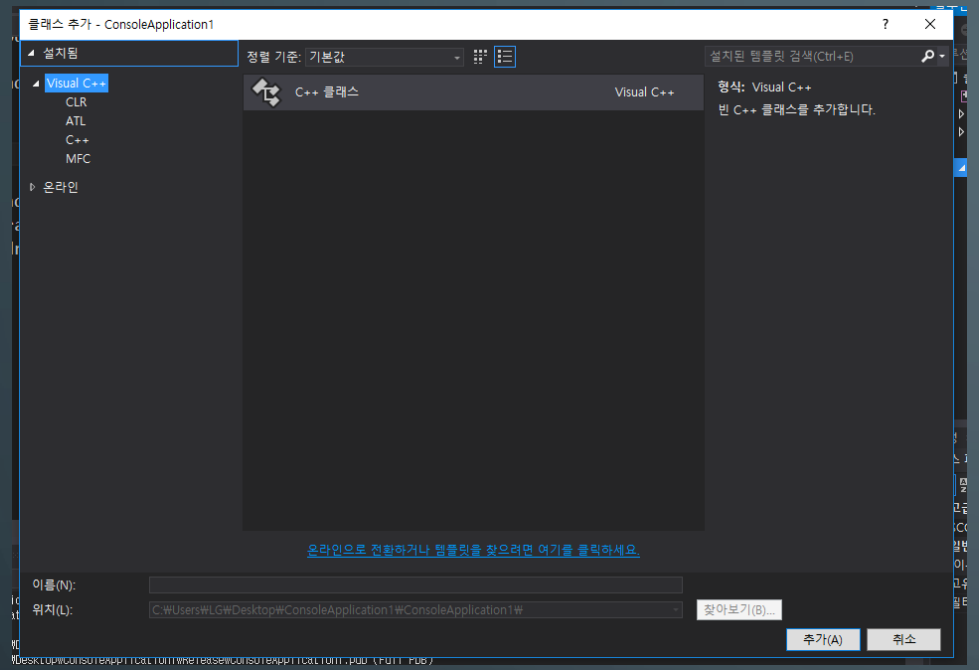
파일 나누기

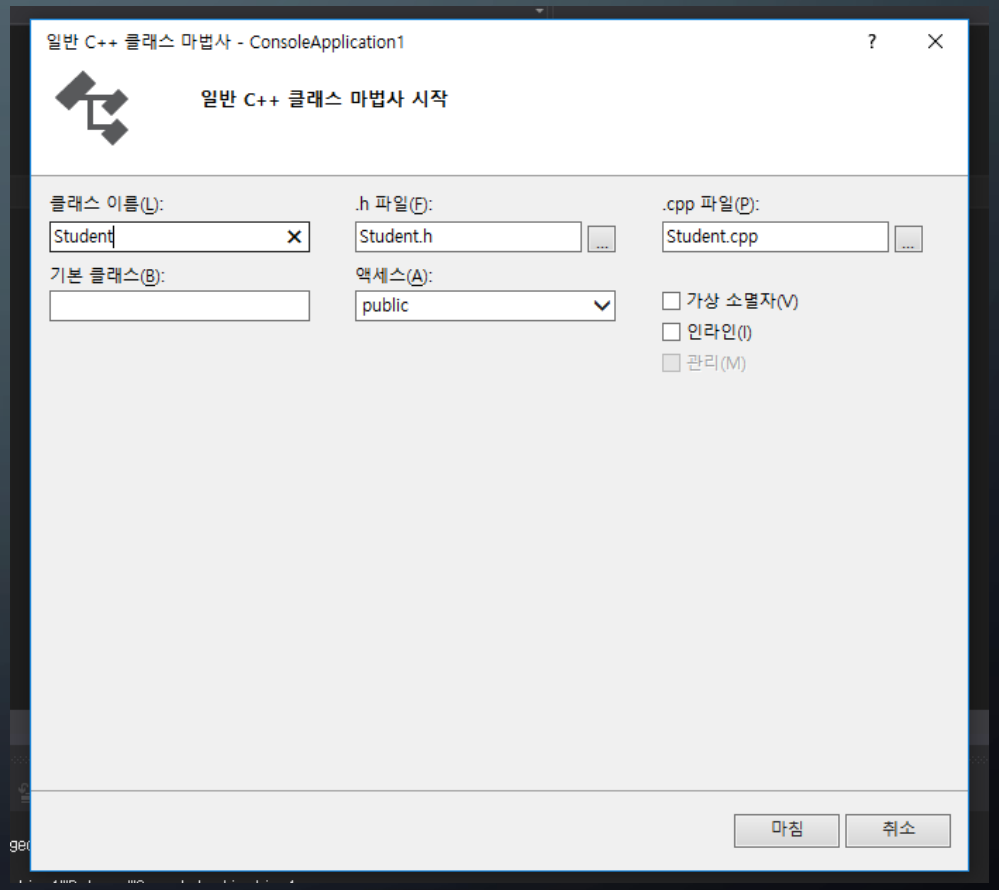
- **Class**를 **.cpp**와 **.h**로 나누어 파일단위로 관리

- **.h**
  Class의 원형 작성
- **.cpp**
  Class의 맴버함수 종속문장 작성



1. 클래스(CLASS)와 객체(OBJECT)





INLINE함수와 CONST

2. INLINE함수와 CONST

헝가리안 표기법

변수의 이름만으로도 변수의 자료형과 선언위치를 알 수 있는 **변수 명명법**

bool = b (ex : bool bState)
int short = i,n (ex : int iNum)
float = f (ex : float fNum)
char = ch (ex : char chNum)
enum = e (ex : GENDER eGender)
NULL 문자 종료 문자열 = sz (ex : char szName[10])
string = str (ex : string strName)
배열 = arr(조합가능) (ex : int iarrNum[10])
포인터 = p(조합가능) (ex : int* ipNum)
전역변수 = g_ (ex : int g_iNum)
맴버변수 = m_ (ex : int m_iNum)
스테틱 변수 = s_ (ex : int s_iNum)

```
#include<iostream>
#include<string>
using namespace std;

int g_iNum;
class A
{
private:
	int m_iNum;
};
void main()
{
	bool bState;
	char szName[20];
	string strName;
	A* parrA[10];
}
```

2. INLINE함수와 CONST

Inline 함수

- 함수의 호출 과정을 생략하고 호출위치에 해당 코드를 적용한다.

장점 : 함수호출 과정이 생략되어 **빠른 호출**이 가능하다.
단점 : 과도하게 사용할 경우 오히려 **속도가 저하될** 수 있다.

```
#include<iostream>
using namespace std;

inline void Test()
{
	cout << "인라인 함수!!" << endl;
}

void main()
{
	cout << "인라인 함수!!" << endl;
}
```

```
#include<iostream>
using namespace std;

inline void Test()
{
	cout << "인라인 함수!!" << endl;
}

void main()
{
	Test();
}
```

2. INLINE함수와 CONST

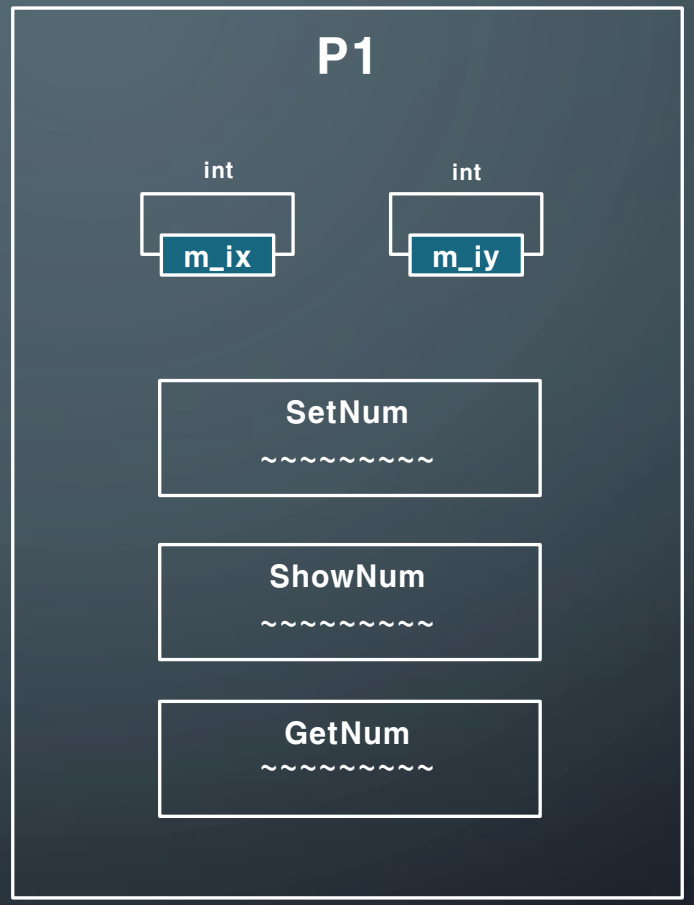
- **Const변수**
  - 해당 변수를 정보의 변경이 불가능한 **상수**로 변환한다.
  - **#define** 매크로 대신 사용하는 것을 권장한다.

- **Const함수**
  해당 함수 종속문장에서 정보의 변환을 막거나 반환 값을 **상수**로 바꾼다.

```
#include<iostream>
Using namespace std;

Void main()
{
    const int Num = 10;
    Num = 20;        →        값을 대입해서 변경이 불가능하다.
}
```

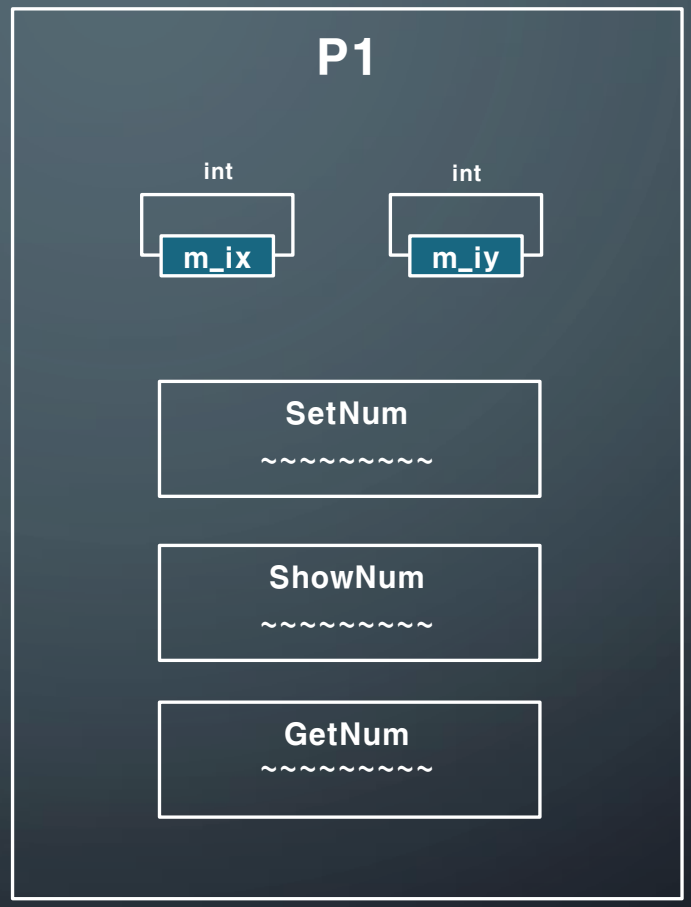
2. INLINE함수와 CONST

```
#include<iostream>
using namespace std;

class Point
{
private:
	int m_ix;
	int m_iy;
public:
	Point(){}
	void SetNum(int x, int y)
	{
		m_ix = x;
		m_iy = y;
	}
	void ShowNum()
	{
		cout << "x = " << m_ix << endl;
		cout << "y = " << m_iy << endl;
		m_ix = 30;
		m_iy = 40;
		cout << "x = " << m_ix << endl;
	}
	int* GetNum()
	{
		return &m_ix;
	}
};

void main()
{
	Point p;
	p.SetNum(10, 20);
	p.ShowNum();
	*(p.GetNum()) = 100;
	p.ShowNum();
}
```



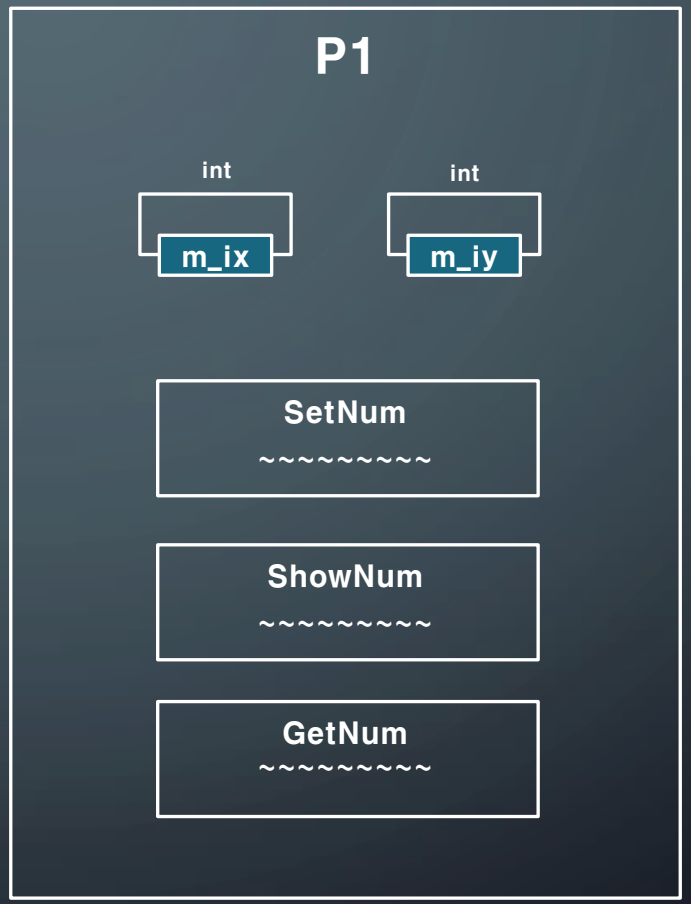
2. INLINE함수와 CONST

```
#include<iostream>
using namespace std;

class Point
{
private:
	const int m_ix;
	int m_iy;
public:
	Point(){}
	void SetNum(int x, int y)
	{
		m_ix = x;
		m_iy = y;
	}
	void ShowNum()
	{
		cout << "x = " << m_ix << endl;
		cout << "y = " << m_iy << endl;
		m_ix = 30;
		m_iy = 40;
		cout << "x = " << m_ix << endl;
	}
	int* GetNum()
	{
		return &m_ix;
	}
};

void main()
{
	Point p;
	p.SetNum(10, 20);
	p.ShowNum();
	*(p.GetNum()) = 100;
	p.ShowNum();
}
```



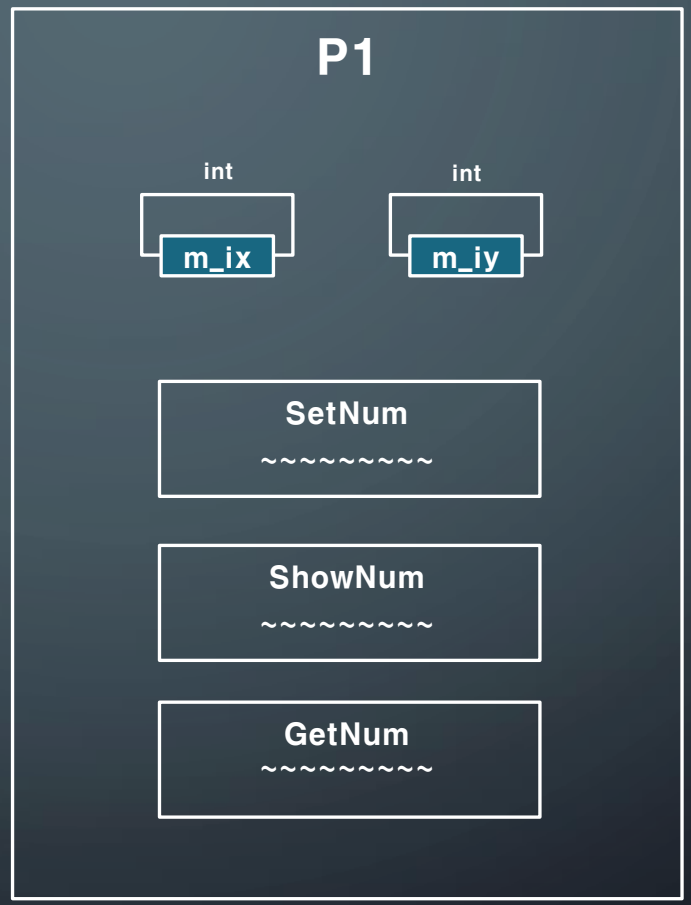
2. INLINE함수와 CONST

```
#include<iostream>
using namespace std;

class Point
{
private:
	int m_ix;
	int m_iy;
public:
	Point(){}
	void SetNum(int x, int y)
	{
		m_ix = x;
		m_iy = y;
	}
	void ShowNum() const // 종속변수에 대한 대입 행위를 금지한다.
	{
		cout << "x = " << m_ix << endl;
		cout << "y = " << m_iy << endl;
		m_ix = 30;
		m_iy = 40;
		cout << "x = " << m_ix << endl;
	}
	int* GetNum()
	{
		return &m_ix;
	}
};

void main()
{
	Point p;
	p.SetNum(10, 20);
	p.ShowNum();
	*(p.GetNum()) = 100;
	p.ShowNum();
}
```



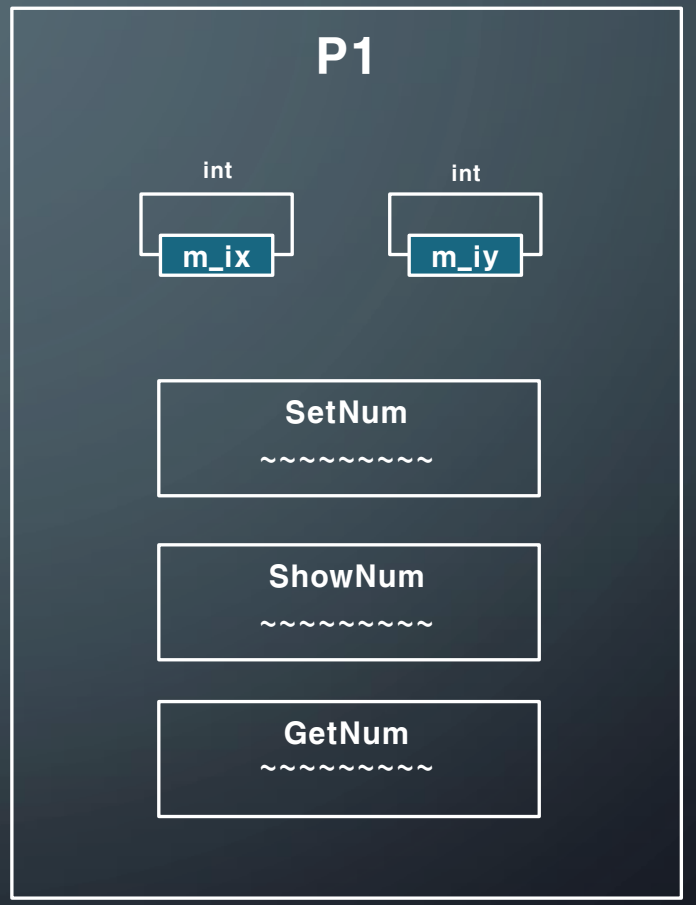
2. INLINE함수와 CONST

```
#include<iostream>
using namespace std;

class Point
{
private:
	int m_ix;
	int m_iy;
public:
	Point(){}
	void SetNum(int x, int y)
	{
		m_ix = x;
		m_iy = y;
	}
	void ShowNum()
	{
		cout << "x = " << m_ix << endl;
		cout << "y = " << m_iy << endl;
		m_ix = 30;
		m_iy = 40;
		cout << "x = " << m_ix << endl;
	}
	const int* GetNum() // 반환 값에 대한 변환을 금지한다.
	{
		return &m_ix;
	}
};

void main()
{
	Point p;
	p.SetNum(10, 20);
	p.ShowNum();
	*(p.GetNum()) = 100;
	p.ShowNum();
}
```



2. INLINE함수와 CONST

```
#include<iostream>
using namespace std;

class Point
{
private:
	int m_ix;
	int m_iy;
public:
	Point(){}
	void SetNum(int x, int y)
	{
		m_ix = x;
		m_iy = y;
	}
	void ShowNum()
	{
		cout << "x = " << m_ix << endl;
		cout << "y = " << m_iy << endl;
		m_ix = 30;
		m_iy = 40;
		cout << "x = " << m_ix << endl;
	}
	const int* GetNum() const // 반환 값의 변환과 반환을 금지 한다.
	{
		m_ix = 50;
		return &m_ix;
	}
};

void main()
{
	Point p;
	p.SetNum(10, 20);
	p.ShowNum();
	*(p.GetNum()) = 100;
	p.ShowNum();
}
```



학습과제

- 구구단을 입력 받고 출력하는 **Class**를 만들어 보자**.**
- 사각형의 가로와 세로를 입력 받고 그리는 **Class**를 만들어 보자**.**
- 학생 관리 클래스 만들기
- 미로찾기를 클래스를 이용해서 업데이트 해보자**.**