```
[ InputManager.h ]
#pragma once
#include <iostream>
#include <vector>
#include <string>

using namespace std;

class IListener;

class CInputManager
{
private:
	vector<IListener*> m_vecListener;
	CInputManager();

public:
	static CInputManager* GetInstance();
	~CInputManager();

	bool AddListener(IListener* listener);
	bool RemoveListener(IListener* listener);
	bool CheckInput();
};


[ InputManager.cpp ]
#include "Listener.h"
#include "InputManager.h"

CInputManager::CInputManager() { }

CInputManager* CInputManager::GetInstance()
{
	static CInputManager instance;
	return &instance;
}

CInputManager::~CInputManager() { }

bool CInputManager::AddListener(IListener* listener)
{
	for (int i = 0; i < m_vecListener.size(); i++)
	{
		if (listener->GetName() == m_vecListener[i]->GetName())
		{
			cout << "Same Name Existence" << endl;
			return false;
		}
	}

	m_vecListener.push_back(listener);
	return true;
}
```

```
bool CInputManager::RemoveListener(IListener* listener)
{
	vector<IListener*>::iterator it;
	for (it = m_vecListener.begin(); it < m_vecListener.end();)
	{
		if ((*it)->GetName() == listener->GetName())
		{
			it = m_vecListener.erase(it);
			return true;
		}
		else
			it++;
	}

	cout << "Fail Remove Listener" << endl;
	return false;
}

bool CInputManager::CheckInput()
{
	string input;
	cin >> input;

	if (input == "q")
		return false;

	for (int i = 0; i < m_vecListener.size(); i++)
		m_vecListener[i]->ReceiveMsg(input);

	return true;
}
```

```
[ Listener.h ]
#pragma once
#include <iostream>
#include <string>

using namespace std;

class IListener
{
protected:
	string m_strName;

public:
	virtual ~IListener() { }
	virtual void ReceiveMsg(string msg) { }
	virtual string GetName() { return string(); }
};
```

```
[ Player.h ]

#pragma once

#include "Listener.h"

class CPlayer : public IListener
{
public:
	CPlayer() { m_strName = "Player"; }
	~CPlayer();

	void ReceiveMsg(string msg) override;
	string GetName() override;
};
```

```
[ Player.cpp ]

#include "Player.h"
void CPlayer::ReceiveMsg(string msg)
{
	if (msg == "w")
		cout << "Player input : W" << endl;
	else if (msg == "s")
		cout << "Player input : S" << endl;
	else if (msg == "a")
		cout << "Player input : A" << endl;
	else if (msg == "d")
		cout << "Player input : D" << endl;
}

string CPlayer::GetName() { return m_strName; }
```

[ObjectCreate.h]

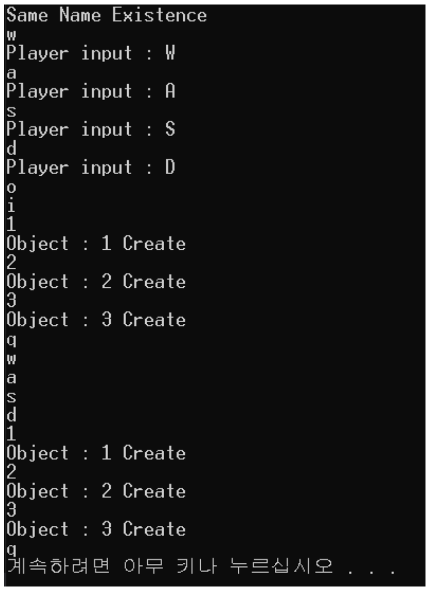
```
#pragma once
#include "Listener.h"

class CObjectCreate : public IListener
{
public:
        CObjectCreate() { m_strName = "object"; }
        ~CObjectCreate() { }

        void ReceiveMsg(string msg);
        string GetName();
};
```

[ObjectCreate.cpp]

```
#include "ObjectCreate.h"
void CObjectCreate::ReceiveMsg(string msg)
{
        if (msg == "1")
                cout << "Object : 1 Create" << endl;
        else if (msg == "2")
                cout << "Object : 2 Create" << endl;
        else if (msg == "3")
                cout << "Object : 3 Create" << endl;
}

string CObjectCreate::GetName() { return m_strName; }
```

[main.cpp]
```cpp
#include <iostream>
#include "InputManager.h"
#include "Player.h"
#include "ObjectCreate.h"
#define g_inputManager CInputManager::GetInstance()

using namespace std;

int main()
{
	CPlayer player;
	CPlayer player2;
	CObjectCreate objCreate;
	g_inputManager->AddListener(&player);
	g_inputManager->AddListener(&player2);
	g_inputManager->AddListener(&objCreate);

	while (1)
	{
		if(!g_inputManager->CheckInput())
			break;
	}

	g_inputManager->RemoveListener(&player);

	while (1)
	{
		if (!g_inputManager->CheckInput())
			break;
	}
	return 0;
}
```

[ScreenShot]



```
Same Name Existence
w
Player input : W
a
Player input : A
s
Player input : S
d
Player input : D
o
i
1
Object : 1 Create
2
Object : 2 Create
3
Object : 3 Create
q
w
a
s
d
1
Object : 1 Create
2
Object : 2 Create
3
Object : 3 Create
q
계속하려면 아무 키나 누르십시오 . . .
```