

Home › Design Patterns › Behavioral patterns › Command

Command in C++: Before and after



[Read full article](#)



[See the code](#)

 **BOOKMARK**

Search



contents

Design Patterns

- § [Creational patterns](#)
- § [Structural patterns](#)
- § [Behavioral patterns](#)
 - [Behavioral patterns](#)
 - [Chain of Responsibility](#)
 - [Command Design Pattern](#)
 - [Interpreter Design Pattern](#)
 - [Iterator Design Pattern](#)
 - [Mediator Design Pattern](#)
 - [Memento Design Pattern](#)

- Null Object Design Pattern
- Observer Design Pattern
- State Design Pattern
- Strategy Design Pattern
- Template Method Design Pattern
- Visitor Design Pattern

Before

01 the client has to query the “type” of each object, and manually invoke the desired method.

02

```
class Giant
{
public:
    enum Type
    {
        Fee, Phi, Pheaux
    };
    Giant()
```

C++

```

{
    m_id = s_next++;
    m_type = (Type)(m_id % 3);
}
Type get_type()
{
    return m_type;
}
void fee()
{
    cout << m_id << "-fee  ";
}
void phi()
{
    cout << m_id << "-phi  ";
}
void pheaux()
{
    cout << m_id << "-pheaux  ";
}
private:
    Type m_type;
    int m_id;
    static int s_next;
};
int Giant::s_next = 0;

template <typename T> class Queue
{
public:
    Queue()
    {
        m_add = m_remove = 0;
    }
    void enqueue(T *c)
    {

```

```

        m_array[m_add] = c;
        m_add = (m_add + 1) % SIZE;
    }
    T *deque()
    {
        int temp = m_remove;
        m_remove = (m_remove + 1) % SIZE;
        return m_array[temp];
    }
private:
    enum
    {
        SIZE = 8
    };
    T *m_array[SIZE];
    int m_add, m_remove;
};

int main()
{
    Queue que;
    Giant input[6], *bad_guy;

    for (int i = 0; i < 6; i++)
        que.enqueue(&input[i]);

    for (int i = 0; i < 6; i++)
    {
        bad_guy = que.dequeue();
        if (bad_guy->get_type() == Giant::Fee)
            bad_guy->fee();
        else if (bad_guy->get_type() == Giant::Phi)
            bad_guy->phi();
        else if (bad_guy->get_type() == Giant::Pheaux)
            bad_guy->pheaux();
    }
}

```

```
cout << '\n';  
}
```

0-fee 1-phi 2-pheaux 3-fee 4-phi 5-pheaux

output

After

03 the desired method is encapsulated in each Command object.

04

```
class Giant  
{  
public:  
    Giant()  
    {  
        m_id = s_next++;  
    }  
    void fee()  
    {  
        cout << m_id << "-fee ";  
    }  
    void phi()  
    {  
        cout << m_id << "-phi ";  
    }  
    void pheaux()  
    {  
        cout << m_id << "-pheaux ";  
    }  
private:
```

C++

```

    int m_id;
    static int s_next;
};
int Giant::s_next = 0;

class Command
{
public:
    typedef void(Giant:: *Action)();
    Command(Giant *object, Action method)
    {
        m_object = object;
        m_method = method;
    }
    void execute()
    {
        (m_object-> *m_method)();
    }
private:
    Giant *m_object;
    Action m_method;
};

template <typename T> class Queue
{
public:
    Queue()
    {
        m_add = m_remove = 0;
    }
    void enqueue(T *c)
    {
        m_array[m_add] = c;
        m_add = (m_add + 1) % SIZE;
    }
    T *deque()

```

```

    {
        int temp = m_remove;
        m_remove = (m_remove + 1) % SIZE;
        return m_array[temp];
    }
private:
    enum
    {
        SIZE = 8
    };
    T *m_array[SIZE];
    int m_add, m_remove;
};

int main()
{
    Queue que;
    Command *input[] =
    {
        new Command(new Giant, &Giant::fee), new Command(new Giant, &Giant::phi),
        new Command(new Giant, &Giant::pheaux), new Command(new Giant, &Giant
            ::fee), new Command(new Giant, &Giant::phi), new Command(new Giant,
            &Giant::pheaux)
    };

    for (int i = 0; i < 6; i++)
        que.enqueue(input[i]);

    for (int i = 0; i < 6; i++)
        que.dequeue()->execute();
    cout << '\n';
}

```

0-fee 1-phi 2-pheaux 3-fee 4-phi 5-pheaux

output

List of Command examples

C# examples

- [Command in C#](#)

C++ examples

- [Command in C++: Before and after <=\[You are here\]](#)
- [Command in C++: Simple and "macro" commands](#)
- [Command in C++](#)

Delphi examples

- [Command in Delphi](#)

Java examples

- [Command in Java: Decoupling producer from consumer](#)
- [Command in Java](#)

PHP examples

- [Command in PHP](#)

< Chain of
Responsibility

↑ Command

Interpreter Design
Pattern >



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 3.0 Unported License](#)

