Áp dụng Command pattern trong Unity.

"Trong lập trình hướng đối tượng, thì Command pattern là một pattern thiết kế hành vi trong đó một đối tượng được sử dụng để đóng gói tất cả các thông tin cần thiết để thực hiện một hành động hoặc kích hoạt một sự kiện ở thời gian sau đó. Thông tin này bao gồm tên phương thức, đối tượng sở hữu phương thức và giá trị cho các tham số của phương thức."

Về mặt cơ bản thì đọc mấy cái thứ lý thuyết này vốn nó cũng ko lấy gì làm thú vị, đôi khi là khô khan và nhàm chán, vì vậy chúng ta sẽ đi tới 1 ví dụ rất cơ bản để thấy được việc sử dụng Command pattern có ý nghĩa thế nào nhé 😉

Bài toán đặt ra là, chúng ta sẽ thực hiện các hành động của nhân vật mỗi khi nhấn một phím bất kì, nếu chỉ có vậy chúng ta sẽ có thể code như sau:

```
if(Input.GetKeyDown(KeyCode.Space))
    {
       Jump();
    }
```

Như vậy khi chúng ta nhấn phím Space thì phương thức Jump() sẽ được gọi, và nhân vật sẽ nhảy lên. Tuy nhiên, bài toán tiếp tục đặt ra yêu cầu đó là, người chơi có thể thay đổi nút bấm để thực hiện hành động nhảy thay vì nút Space trên bàn phím, ví dụ như bấm phím J.

Lúc này chúng ta không thể code thuần túy như trên được, chúng ta sẽ cần sử dụng Command pattern để dễ dàng cho việc User muốn thay gì thì thay :v
Bước 1: định nghĩa Command:

```
public abstract class Command
{
        public abstract void Execute();
}
```

Bước 2: Kế thừa class Command để định nghĩa ra class Jump và class DoNothing:

```
public class Jump : Command
{
        public override void Execute()
        {
                JumpNow();
        }
}
public class DoNothing : Command
{
        public override void Execute()
        {
        }
}
```

Note: Sở dĩ chúng ta tạo ra class DoNothing là để hủy chức năng của một phím trước đó đã được gán các bạn nhé 😉

Bước 3: Khái báo và sử dụng Jump, DoNothing:

```
Command buttonSpace = new DoNothing();
Command buttonJ = new Jump();

if (Input.GetKeyDown(KeyCode.Space))
{
        buttonSpace.Execute(); //Sẽ không còn thực hiện bất cứ hành động gì nữa.
}
if (Input.GetKeyDown(KeyCode.J))
{
        buttonJ.Execute(); //Nhân vật sẽ thực hiện hành động nhảy.
}
```

Bước 4: Thêm 1 phương thức thay đổi nút để giúp người chơi có thể thay đổi trong giao diện người dùng:

```
public void ChangeButtonJump(string buttonName)
   {
      switch (buttonName)
      {
         case "J":
            buttonSpace = new DoNothing();
            buttonJ = new Jump();
            break;
         case "Space":
            buttonJ = new DoNothing();
            buttonSpace = new Jump();
            break;
         default:
            break;
      }
   }
```

Code đầy đủ sẽ là:

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
namespace CommandPattern
{
   public abstract class Command
   {
      public abstract void Execute();
   }
```

```
    // Kế thừa class Command và ghi đè phương thức Execute theo mong muốn.
    public class Jump : Command
    {
        public override void Execute()
        {
            JumpNow();
        }
    }

    // Kế thừa class Command và ghi đè phương thức Execute để không có hành động nào
được thực hiện.
    public class DoNothing : Command
    {
        public override void Execute()
        {
        }
    }
}
```

Tạo file InputHandler.cs có nội dung như sau:

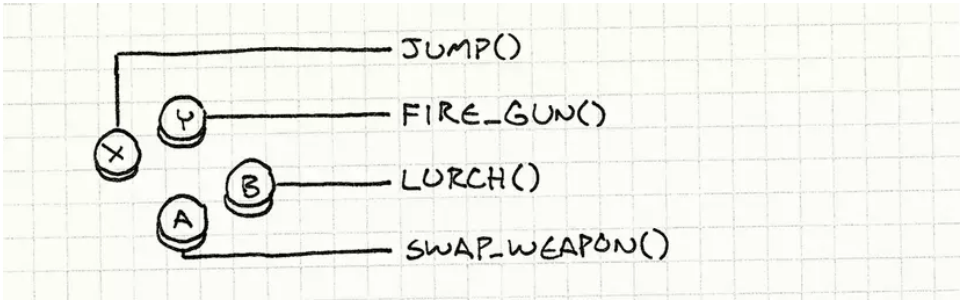
```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

namespace CommandPattern
{
    public class InputHandler : MonoBehaviour
    {
        Command buttonSpace = new DoNothing();
        Command buttonJ = new Jump();

        void Update () {
            if (Input.GetKeyDown(KeyCode.Space))
            {
                buttonSpace.Execute(); //không thực hiện bất cứ hành động gì.
            }
            if (Input.GetKeyDown(KeyCode.J))
            {
                buttonJ.Execute(); //Nhân vật sẽ thực hiện hành động nhảy.
            }
        }

        public void ChangeButtonJump(string buttonName)
        {
            switch (buttonName)
            {
                case "J":
                    buttonSpace = new DoNothing();
```

```
                buttonJ = new Jump();
                break;
            case "Space":
                buttonJ = new DoNothing();
                buttonSpace = new Jump();
                break;
            default:
                break;
        }
    }
  }
}
```

Như vậy người chơi đã có thể thay phím nhảy từ phím J thành phím Space và ngược lại một cách dễ dàng.

Một ví dụ cho việc trước và sau khi sử dụng Command Pattern để các bạn dễ hình dung hơn:

Trước



Sau