11.实现“某个按钮点击时，使所有当前页面的按钮不可被点击”

标题有一点啰嗦。但是这个需求我们经常会遇到。

我们一般的实现是加上一个事件的遮罩层，或者创建一个 bool 变量来进行标记。

而使用 UniRx 则会简单得多。

代码如下:

```
public class TestPanel : XXXPanel
{
    [SerializeField] Button mButtonA;
    [SerializeField] Button mButtonB;
    [SerializeField] Button mButtonC;

    void Start()
    {
        var buttonAStream = mButtonA.OnClickAsObservable();
        var buttonBStream = mButtonB.OnClickAsObservable();
        var buttonCStream = mButtonC.OnClickAsObservable();

        Observable.Merge(
                    buttonAStream,
                    buttonBStream,
                    buttonCStream)
                .First()
                .Subscribe(_ =>
                {
                    // any button clicked
                });
    }
}
```

使用 Merge 将三个按钮的事件流合并。并通过 First 只处理第一次点击事件。

这样标题的所说的功能就完成了。

但是还有一点问题，就是当处理按钮事件的时候要知道是哪个按钮被点击了。

很简单，使用 Select 操作符就好了。

Select 是什么意思呢?

就是选择。

Select 本身是 Linq 的操作符。

一般是传入一个索引 i/index 然后根据索引返回具体的值。

对于一个 List，什么叫做 Selecte 呢?

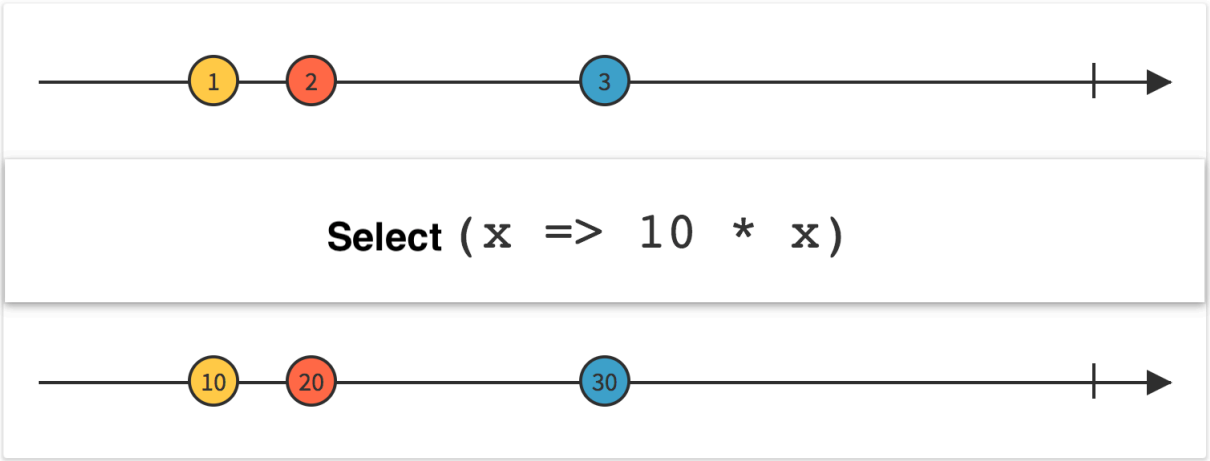
看代码就懂了,如下:

```
var testNumbers = new List<int>(){ 1,2,3}
var selectedValue = testNumbers[2];
```

其中 testNumbers[2] 就是一个选择操作。

Select 的操作符会在第二章进行详细解释。

先用下边的图，试着理解一下就好。



这里我们只要了解，使用 Select 操作符后，它返回的是一个值就好了，值的类型根据它返回的值的类型决定，也就是说是一个泛型的。

使用 Select 之后的代码如下:

```
public class TestPanel : XXXPanel
{
    [SerializeField] Button mButtonA;
    [SerializeField] Button mButtonB;
    [SerializeField] Button mButtonC;

    void Start()
    {
        var buttonAStream = mButtonA.OnClickAsObservable().Select(_=>"A");
        var buttonBStream = mButtonB.OnClickAsObservable().Select(_=>"B");
        var buttonCStream = mButtonC.OnClickAsObservable().Select(_=>"C";

        Observable.Merge(
                    buttonAStream,
                    buttonBStream,
                    buttonCStream)
                .First()
                .Subscribe(buttonId =>
                {
                    if (buttonId == "A")
                    {
```

```
                    // Button A Clicked
                }
                else if (buttonId == "B")
                {
                    // Button B Clicked
                }
                else if (buttonId == "C")
                {
                    // Button C Clicked
                }
            });
    }
}
```

这样，标题所说的功能，完美实现了。

第一章的内容就结束了。

Merge 属于处理多个流的操作符，除此之外还有更多的类似的，比如 Zip 等等。学习完它们之后可以实现非常复杂的逻辑，比如 Coroutine 的支持，可以实现按顺序执行多个 Coroutine ，也支持等待所有 Coroutine 执行完成这种复杂的操作。

这些内容我们在第二章中进行学习。