条款10：优先考虑限域enum而非未限域enum

①enum中的每个元素的底层类型都是一个由编译器决定的整型

```
enum Color { red, block, white };
```

底层类型为int，未指定则为0

甚至可以手动设置，且可以相同，往后的整型+1

②老枚举(未限定域enum)的问题

a.作用域全局

```
enum A { a, b };
enum B { b, c };
```

同名报错

b.可隐式转为整形

```
int x = red;
```

导致不同枚举类同位置条件判断居然能过

c.通常情况下无法前置声明，由于不知道要分配多大空间，除非在声明与定义处都指定其整型的类型

```
enum School : int;
enum People;
```

单个指定，均会报错

③枚举类(enum class)解决了上述问题

a.使用时要限定作用域

```
Color a = Color::blue;
```

b.条件判断不再依靠于整形值

c.无须显示指明，默认为int

④老枚举就完全不可取吗

非并，老枚举可用于位置(数)←→意义的对应

而枚举类涉及到强转，不便于使用（强转可用函数封装）

↓

编译期

⑤ C++11后，可使用列表初使化有底层类型的枚举对象

（限域，不限域都行）

应用场景：需要一个新整数类型，该类型必须严格区别于其它类型

```
enum class Color{ red, block, white};
Color _a{0}; //此时a为red
enum class Index: int {};
```

⑥ C++20后 使用using打开限域enum

```
enum class Color{ red, block, white};
const char * ColorToString(Color C){
    switch(c){
        using enum Color; 打开域
        case Red: //避免了重复指明作用域
            return "Red";
            break;
        ........}
```