12.用接口，泛型，枚举，函数来定义抽象数据类型（ADT）

抽象数据类型尽可能用泛型

抽象数据类型的函数可能被实现为返回值是“int”，“double”，或者其他的，所以尽可能用泛型

为什么用接口实现抽象数据类型

废话，抽象数据类型要求被实现为具体的数据类型，要写在接口中

Suppose you have an abstract data type for rational numbers, which is currently represented as a TypeScript class:

```
class Rational {
    ...
}
```

You decide to change to a TypeScript interface instead, along with an implementation class called : `Rational` `IntFraction`

```
interface Rational {
    ...
}

class IntFraction implements Rational {
    ...
}
```

当值的可能集合较小时用枚举

函数尽可能用getter 和 setter

ADTs in TypeScript

We've now completed our TypeScript toolbox of ADT concepts from the first ADTs reading:

ADT concept	Ways to do it in TypeScript	Examples
Abstract data type	Class	`Date`
	Interface + class(es) [1]	`ArrayLike` and `Array`
	Enum [2]	`PenColor`
Creator operation	Constructor	`Array()`
	Static (factory) method	`Array.of()`
	Constant [3]	`Number.POSITIVE_INFINITY`
Observer operation	Instance method	`String.charAt()`
	Static method	`Object.entries()`
	Getter	`Map.size`
Producer operation	Instance method	`String.trim()`
	Static method	`Math.floor()`
Mutator operation	Instance method	`Array.push()`
	Static method	`Object.assign()`
	Setter	
Representation	`private` fields	