- **Creators** create new objects of the type. 构造器
  - A creator may take an object as an argument, but not an object of the type being constructed.
- **Producers** create new objects from old objects of the type. 生产器
  - The `concat()` method of `String`, for example, is a producer: it takes two strings and produces a new one representing their concatenation.
- **Observers** take objects of the abstract type and return objects of a different type. 观察器
  - The `size()` method of `List`, for example, returns an `int`.
- **Mutators** change objects. 变值器，改变对象属性的方法
  - The `add()` method of `List`, for example, mutates a list by adding an element to the end.

Representation Independence

测试creators, producers, and mutators：调用observers来观察这些 operations的结果是否满足spec；

测试observers：调用creators, producers, and mutators等方法产生或 改变对象，来看结果是否正确。

除非迫不得已，否则不要把希望寄托于客户端上，ADT有责任保证自 己的invariants，并避免"表示泄露"。

最好的办法就是使 用immutable的类型，彻底避免表示泄露

AF(abstract function)：ADT中的属性表示的一个抽象物体，通俗来说就是这个ADT是用来干什么的

RI(rep invariant)：所有表示值的一个子集，包含了所有合法的表示值。也可以说RI就是ADT中的属性应该遵循的规范

- Recall that a type is immutable if and only if a value of the type never changes after being created.
- With our new understanding of the abstract space A and rep space R, we can refine this definition: the *abstract value* should never change.
- But the implementation is free to mutate a *rep value* as long as it continues to map to the same abstract value, so that the change is invisible to the client.
- This kind of change is called **beneficent mutation** (有益的可变性).

- 对immutable的ADT来说，它在A空间的abstract value应是不变的。
- 但其内部表示的R空间中的取值则可以是变化的。

例子：分数有最简分数和不是最简分数，但不影响结果

AF，RI，safety from exposure均以注释形式写出，为了不暴露给客户端