example of mutability vs immutability

dogs Dog[] = {new Dog "mydog1", new Dog "mydog2", }
animals animal[] = dogs
animals[0] = new Cat "mycat"

Java-like
Mutability, updates are in place

Immutability,
no updates are in place,
allocate another part of the memory
Lists and Arrays must be invariant in languages with update/mutation.

Suppose Arrays are covariant.
Let’s have a class Dog with method “bark” and a class Cat without it.
Let’s have Dog and Cat are subclasses of Animal

```java
Dog[] dogs = { new Dog("mydog1"), new Dog("mydog2") };  
Animal[] animals = dogs;  // type checks because Dog <: Animal
animals[0] = new Cat("mycat");  // type checks because Cat can go in Animal
dogs[0].bark();  // type checks because Dog has “bark”
```

However: unsound at run-time

Java has covariant Arrays but then it has to double check arrays at run-time:
Here, Java throws ArrayStoreException at run-time