Generics & Inheritance

…

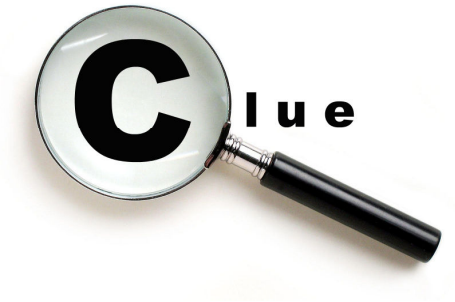
Let's play a bit…

```
//Quick Reminder on Polymorphism
Object     someObject      = new Object();
Integer    someInteger     = new Integer(10);
someObject = someInteger;   // OK since Integer is-a Object

// Integer is-a Number so we can go further…
public void someMethod(Number n) { /* ... */ }
someMethod(new Integer(10));      // OK
someMethod(new Double(10.1));     // OK

// Same w/ Generics! Well… Kinda…
Box<Number> box    = new Box<Number>();
box.add(new Integer(10));      // OK
box.add(new Double(10.1));     // OK

// Integer / Double are alright since both inherit from Number
```

Now, let's consider the following…

```
public void boxTest(Box<Number> n) { /* ... */ }
```

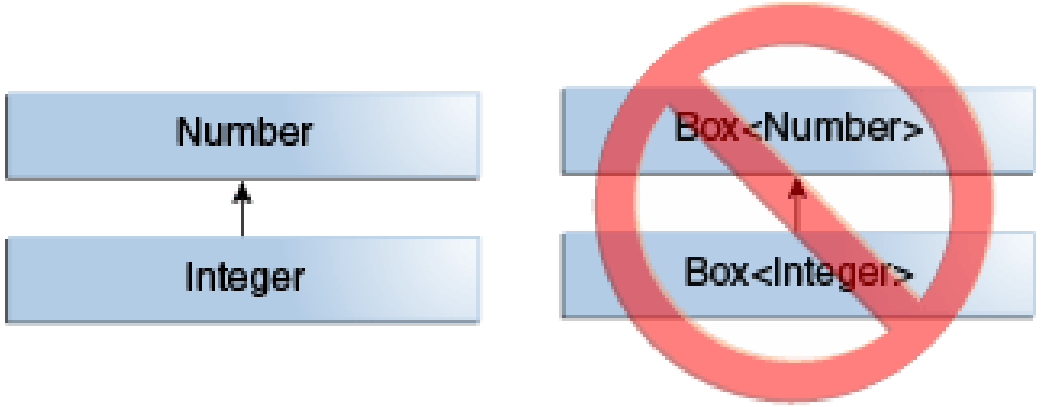
? Are Box<Integer> and Box<Double> suitable as data types for the argument?



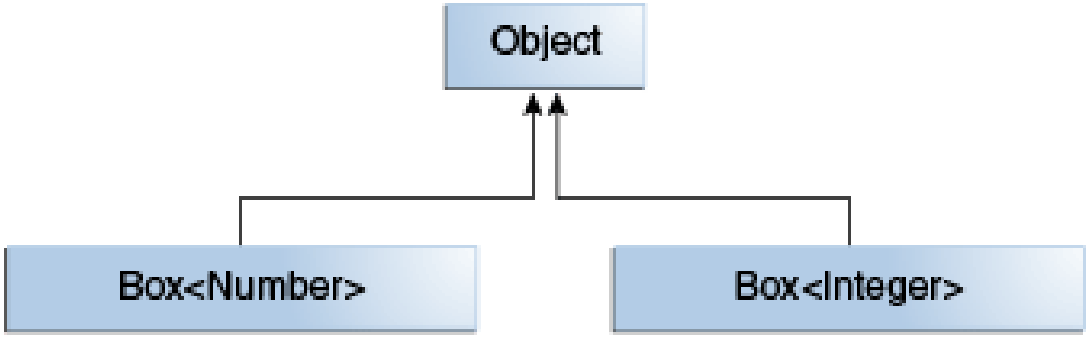
Think about Polymorphism…

! Nope, these are not allowed
Yes, the clue was misleading

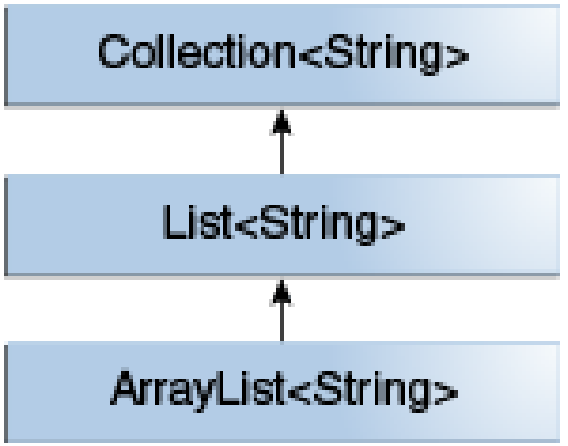
Here are some rules to keep in mind…



- Given two concrete types A and B
- MyClass<A> has no relationship to MyClass<B>, regardless of whether A and B do
- Common parent of MyClass<A> and MyClass<B> is Object



However… e.g. Java Collections Classes



So long as you do not vary the type argument…

the subtyping relationship is preserved between the types

New Requirements!

Let's extend a List<E> interface so that it integrates a 2nd Type Parameter P

```
interface PayloadList<E,P> extends List<E> {
  void setPayload(int index, P val);
  ...
}
```

New Requirements!

The following parameterizations of PayloadList are subtypes of List<String>;

```
PayloadList<String,String>
PayloadList<String,Integer>
PayloadList<String,Exception>
```