Generics

You never know what you're gonna get.

Bram Janssens

Solid Innovator


by Bram Janssens


“I'm lost
in a forest”
https://www.nporadio2.nl/nieuws/top2000/47902840-10d7-4bb8-96fd-509faa351cce/met-a-forest-vond-the-cure-zijn-identiteit

Forrest Gump **(1994) - Paramount Pictures — all screenshots in these slides are taken from Netflix.**

My mama always told me...

```
void main() {
    life = like.a(box).of(chocolates);
}
```

```
> java nl.bramjanssens.RunForrest
```

```
You never know what you're gonna get.
```

– Forrest Gump

> Generics – You Never Know What You're Gonna Get.

– Bram Janssens

What could go wrong?

Required type: capture of ? extends Candy

Provided: GummyBear

```
List<byte> bites;
```

Type argument cannot be of a primitive type

Required type: capture of ? extends Chocolate

Provided: capture of ? extends Chocolate

both methods have same erasure

Cannot access class object of a type parameter

Type parameter 'T' cannot be instantiated directly


SHIT HAPPENS

Bram Janssens

Trainer/consultant

infoSupport

Solid Innovator

Introducing...

Forrest

🏃

Jenny

🍬 = ❤️

🧸 = 💔

Bubba

💪 🦐


I'm not a smart man, but I know what love is...

The story

- Forrest ❤️ Jenny.
- He buys her a 🎁 of 🍬s.
- Before handing it over, he inspects the box to:
  - 👃 Smell if everything is fresh.
  - ➕ Append if needed.

First try

```
List<byte> bites = new ArrayList<>();

bites.add(bite1);
bites.add(bite2);
bites.add(bite3);
bites.add(bite4);
```

Type argument cannot be of a primitive type

Replace 'byte' with 'java.lang.Byte' ⌥⇧↵ More actions... ⌥↵

Whát 🤔?

infoSupport Solid Innovator

Why?

- Generics introduced in Java 5 (2004)
  - Java 4 compiled code should still run on JVM 5
  - ..... JVM 6
  - ..... JVM 7

    
  - ..... JVM 26!

The JVM is backwards compatible.

... let me tell you a secret

The JVM doesn't know generics...

How?

Compile time

```
List<Integer> box = …

boolean add(T e) { … }

T getFirst() {
  return (T) get(0);
}
```

Run time

```
List box = …

boolean add(Object e) { … }

Object getFirst() {
  return (Object) get(0);
}
```

byte != Object


That’s called Type Erasure.

Implication

- Because generics are a compile time construct in Java....
- ... `T` is gone at runtime: non-reifiable

```
public <T> void materialize() {
    var clz = T.class; // ❌
    var t = new T();   // ❌
}
```

Cannot access class object of a type parameter

Type parameter 'T' cannot be instantiated directly

Implication

When compiling `Box<?>`

`?` must be temporarily “captured” during compilation.

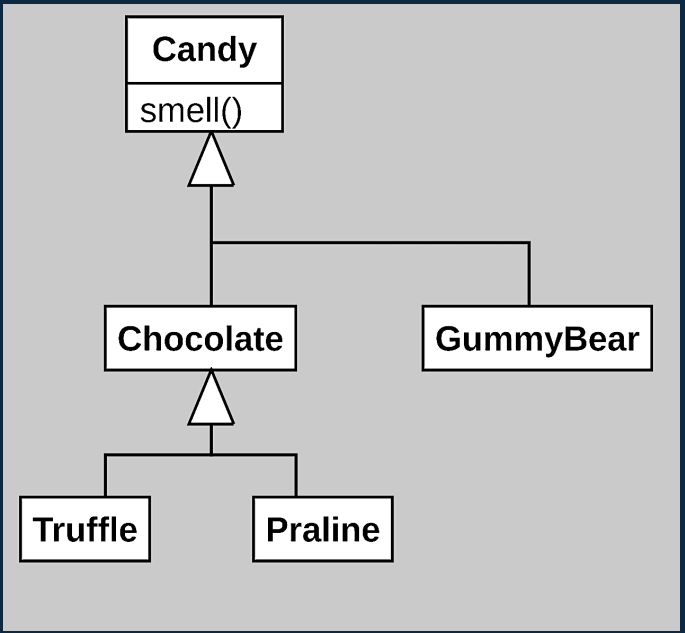
- wild card capturing = inferring the type of a wildcard
- `capture of ?` = the inferred type (compiler internal)

```
java: incompatible types: nl.bramjanssens.domain.GummyBear cannot be
converted to capture#1 of ? extends nl.bramjanssens.domain.Candy
```

> I don't wanna be called stupid.

– Forrest Gump

Let's help him out...



Subtype

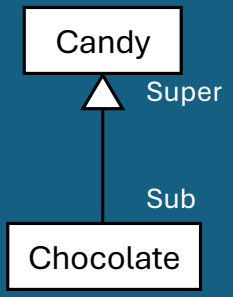
```
void inspect(Candy c) {
    c.smell();
}
```

```
forrest.inspect(candy);
forrest.inspect(chocolate);
```

Allowed?

✅

Simple types



Subtype with generics?

```
// 1. Declaration site
public class Box<T> { } // with type parameter T
//     class ^^^^^^
```

```
// 2. Use site
void inspect(Box<Candy> box) { } // with type argument Candy
//     type  ^^^^^^^^^^
```

```
forrest.inspect(candies);
forrest.inspect(chocolates);
```

Cannot resolve method 'inspect(Box<Chocolate>)'

Why not?

Type **unsafe** from *this* context!

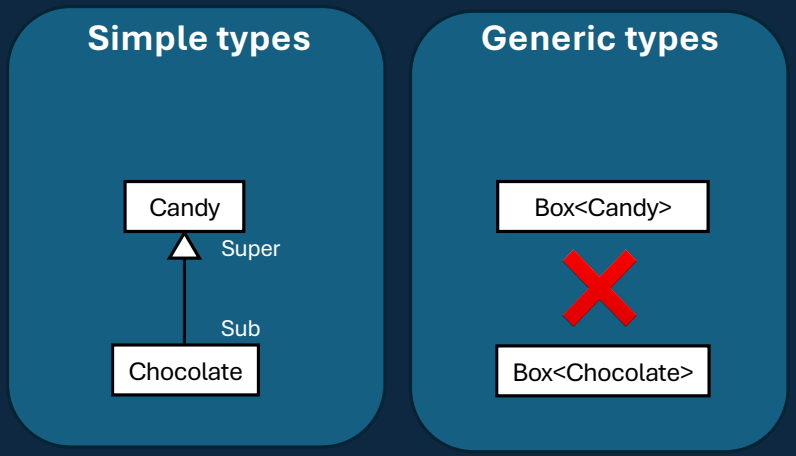
`chocolates` + 🧸 = 💔

```
public void inspect(Box<Candy> box) {
    box.forEach(Candy::smell);    // read  ✅
    if (box.isNotSquare()) {
        box.add(new GummyBear()); // write ✅
    }
}
```

Type **safe** in *this* context.

Conclusion

`Box<Chocolate>` is NOT a subtype of `Box<Candy>`



Invariant = “**no** direction”

Overloading?

```
void inspect(Box<Candy> box) { }     // with type argument Candy
void inspect(Box<Chocolate> box) { } // with type argument Chocolate
```

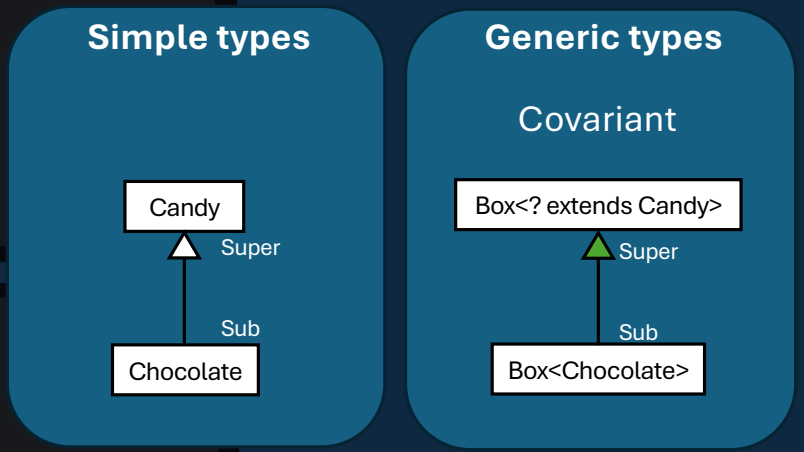
'inspect(Box<Candy>)' clashes with 'inspect(Box<Chocolate>)'; both methods have same erasure

- Aha! 💡
  - `Box<Candy> == Box<Chocolate> == Box`

Next attempt…

```
// 2. Use site
public void inspectSafely(Box<? extends Candy> box) {
//                  type  ^^^^^^^^^^^^^^^^^^^
}
```

```
forrest.inspectSafely(candies);    // ok
forrest.inspectSafely(chocolates); // ok
```



Covariant = “**same** direction”

infoSupport
Solid Innovator

Implications covariant

candies = 💖

chocolates + ~~🧸~~ = 💖

```
public void inspectSafely(Box<? extends Candy> box) {



}
```

Provided: GummyBear

box is a *producer*

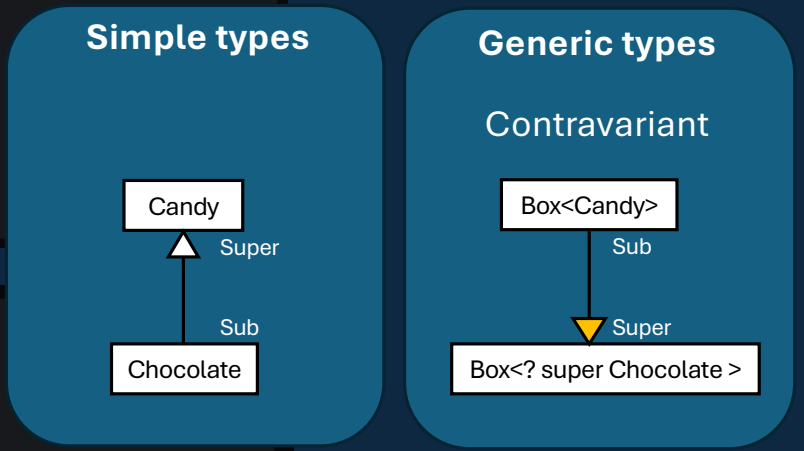
Not done yet…

Backlog	Done?
👃 Smell if each candy is fresh.	✅
➕ Safely append.	❌

Final attempt

```
// 2. Use site
public void inspectSafely(Box<? super Chocolate> box) {
//                  type  ^^^^^^^^^^^^^^^^^^^^^^
}
```

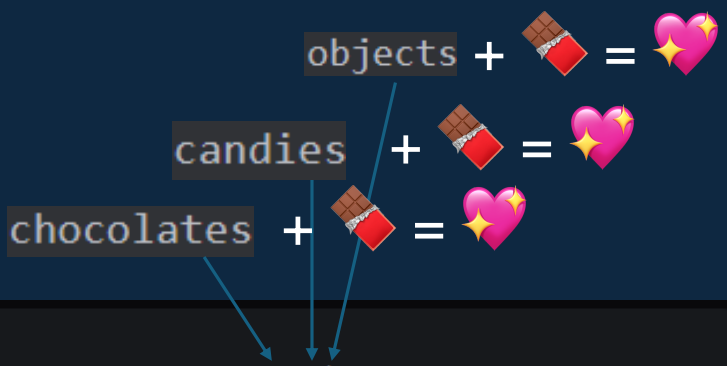
```
forrest.inspectSafely(candies);    // ok
forrest.inspectSafely(chocolates); // ok
```



Contravariant = “**opposite** direction”

We have a winner! 🥇

Implications...



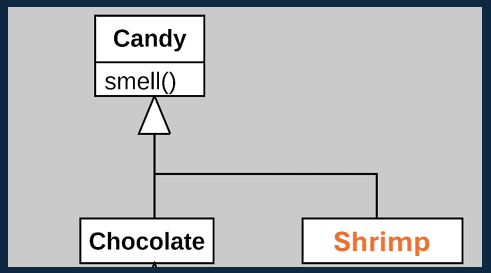
```
public void inspectSafely(Box<? super Chocolate> box) {



}
```

box is a *consumer*

infoSupport
Solid Innovator

```
bubba.inspect(arrayOfChocolates);
```

```
public void inspect(Candy[] candies) {
    stream(candies).forEach(Candy::smell);
    candies[candies.length - 1] = new Shrimp();
}
```



Does this compile..? ✅ (so array is covariant by default!)

Runtime: `chocolates` + 🍤 = ...

infoSupport
Solid Innovator

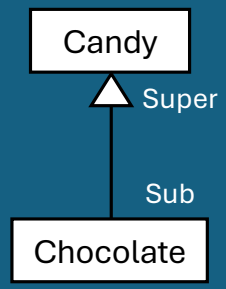

Exception in thread "main" java.lang.ArrayStoreException: nl.bramjanssens.domain.Shrimp
Arrays are covariant but NOT type safe, Bubba!

Wrap up

Java Variance Cheat Sheet

Bridging the gap between inheritance hierarchies and generic type systems.

Simple types



Generic types

Invariant	Covariant producer	Contravariant consumer
Box<Candy>	Box<? extends Candy>	Box<Candy>
✖	Super	Sub
Box<Chocolate>	Sub Box<Chocolate>	Super Box<? super Chocolate >

PECS

Producer **`extends`**
Consumer **`super`**

Arrays are covariant and not **type safe!**

by Bram Janssens

infoSupport
Solid Innovator

Generics

Now you know what you’re gonna get!

bramjanssens.nl

@sajanssens

/sajanssens

InfoSupport
Solid Innovator