Covariance/Contravariance

Statically override a method, use same argument and result types.

Safe but overly restrictive.

Covariance allows:

More *precise* argument and result types.

Contravariance allows:

More *precise* result type.
More *general* argument type.

Covariance

The following methods would be legal:

```
Number Number.add(Number);
Integer Integer.add(Integer);
```

`Integer` is a subtype of `Number`.

What happens with the following?

```
Number i = new Integer(...);
Number n = new Number(...);
Number m;

m = n.add(n);
m = i.add(n);
```

This passes *static* type-checking but `i.add(n)` fails at runtime.

Contravariance

The following methods would be legal:

```
Number Number.add(Number);
Integer Integer.add(Number);
```

`Integer` is a subtype of `Number`.

What happens with the following?

```
Number i = new Integer(...);
Number n = new Number(...);
Number m;

m = n.add(n);
m = i.add(n);
```

This passes *static* type-checking and *dynamic* type-checking.