

Types

Julien Wetterwald

CS94SI

Subtyping

(substitution principle)

If S is a subtype of T , then objects of type T may be replaced with objects of type S without altering any of the desirable properties of the program.

Type Inference

```
abstract class A  
class B extends A  
class C extends A
```

```
val answer: ? =  
  if (condition) new B  
  else new C
```

Type Inference

```
abstract class A  
class B extends A  
class C extends A
```

```
val answer: ? =  
    if (condition) new B  
    else throw new Exception
```

Compound types and structural subtyping

(subtyping doesn't imply inheritance)

see next slide

Anonymous classes in Java

```
class Person { }
```

```
? p1 = new Person();
```

```
? p2 = new Person() {  
    String hi() {  
        return "hello";  
    }  
}
```

```
p2.hi();
```

Abstract type members

Type parameters

Type bounds

Variance

Path-dependent types

Type constructor polymorphism (higher-kinded types)

Generics of a Higher Kind

by Adriaan Moors, K.U.Leuven, Frank Piessens, K.U.Leuven, and Martin Odersky, EPFL