A Brief Introduction to Behavioral Subtyping

Traditional Subtyping

```scala
class Foo {
  def add1(n: Int) = n + 1
}

class Bar extends Foo {
  override def add1(n: Int) = n - 1
}


val a = new Foo
a.add1(5) // => 6

val b = new Bar
b.add1(5) // => 4

val c: Foo = new Bar  // expecting behavior of Foo
c.add1(5) // => 4, instead get that of Bar
```

Behavioral Subtyping

Let q(x) be a property provable about objects x of type T. Then q(y) should be true for objects y of type S where S is a subtype of T.

```
def m(a: Int): (b: Int)
  // pre-condition: -10 < a < 0
  // post-condition: 10 < b < 30

def m'(a': Long): (b': Short)
  // pre-condition: -20 < a' < 10
  // post-condition: 15 < b' < 25
```

Short: 16-bit integer
Int: 32-bit integer
Long: 64-bit integer

$\text{Short} \sqsubseteq \text{Int} \sqsubseteq \text{Long}$

(assuming automatic type coercion)

Substitution Rule

m’ is a subtype of m if m’ can be used in place of m while satisfying the type signature, pre-/post-conditions such that anyone using m is not affected by such substitution

Relax Pre-Condition

```
def m(a: Int): (b: Int)
  // pre-condition: -10 < a < 0
  // post-condition: 10 < b < 30

def m'(a': Long): (b': Short)
  // pre-condition: -20 < a' < 10
  // post-condition: 15 < b' < 25
```

Relax Input Type

```
def m(a: Int): (b: Int)
  // pre-condition: -10 < a < 0
  // post-condition: 10 < b < 30

def m'(a': Long): (b': Short)
  // pre-condition: -20 < a' < 10
  // post-condition: 15 < b' < 25
```

Restrict Output Type

```
def m(a: Int): (b: Int)
  // pre-condition: -10 < a < 0
  // post-condition: 10 < b < 30

def m'(a': Long): (b': Short)
  // pre-condition: -20 < a' < 10
  // post-condition: 15 < b' < 25
```

Restrict Post-Condition

```
def m(a: Int): (b: Int)
  // pre-condition: -10 < a < 0
  // post-condition: 10 < b < 30

def m'(a': Long): (b': Short)
  // pre-condition: -20 < a' < 10
  // post-condition: 15 < b' < 25
```

m		m'	
pre	$\Longrightarrow$	pre'	relax pre-condition
A	$\sqsubseteq$	A'	relax input type
$\downarrow$		$\downarrow$	
B	$\sqsupseteq$	B'	restrict output type
post	$\Longleftarrow$	post'	restrict post-condition

Airline Reservation

Genesis

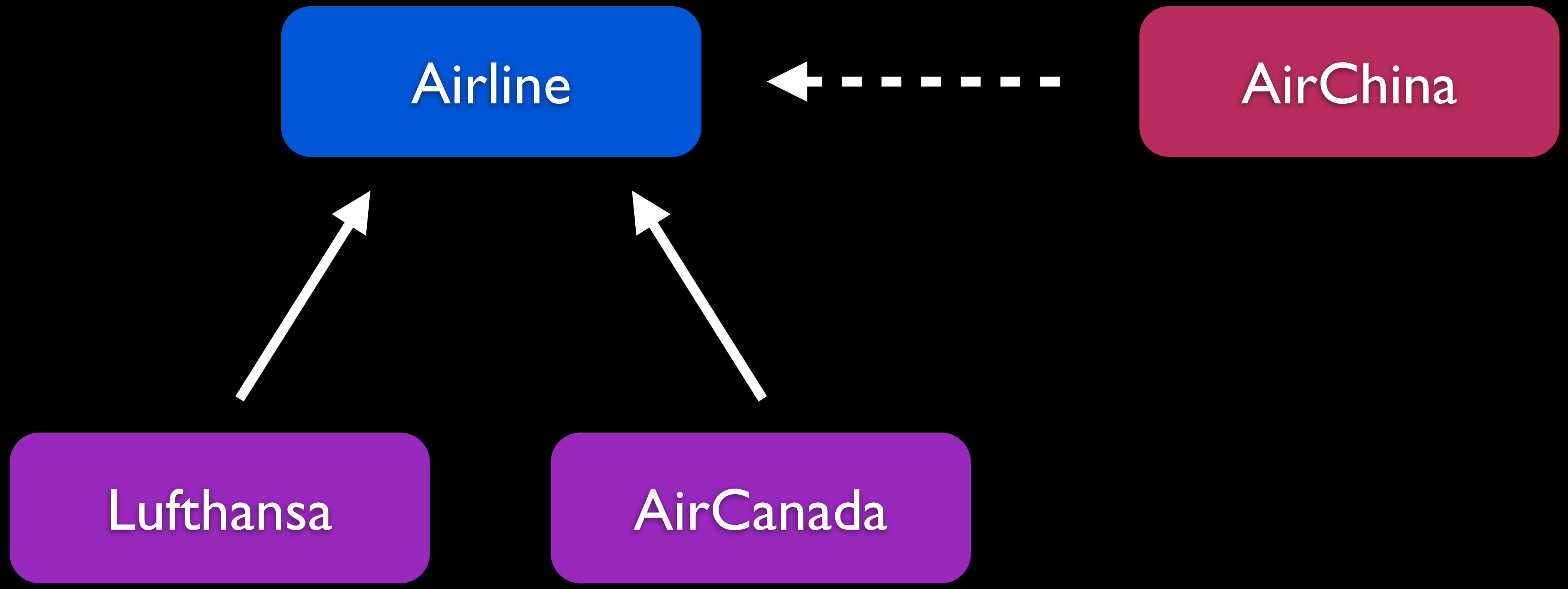
- Universal time and single currency
- 10 cities
  - each city has at least one airport
  - some cities must have multiple airports
  - http://airportcode.riobard.com might help
- 30 flights between cities your airline serves
  - regular schedule (e.g. on a weekly basis)
  - must have connect flights

Demo

basic types

Demo

specification in pseudo Scala code


Airline
AirChina
Lufthansa
AirCanada

Questions?

email me@riobard.com
visit http://groups.google.com/group/scala-course-project