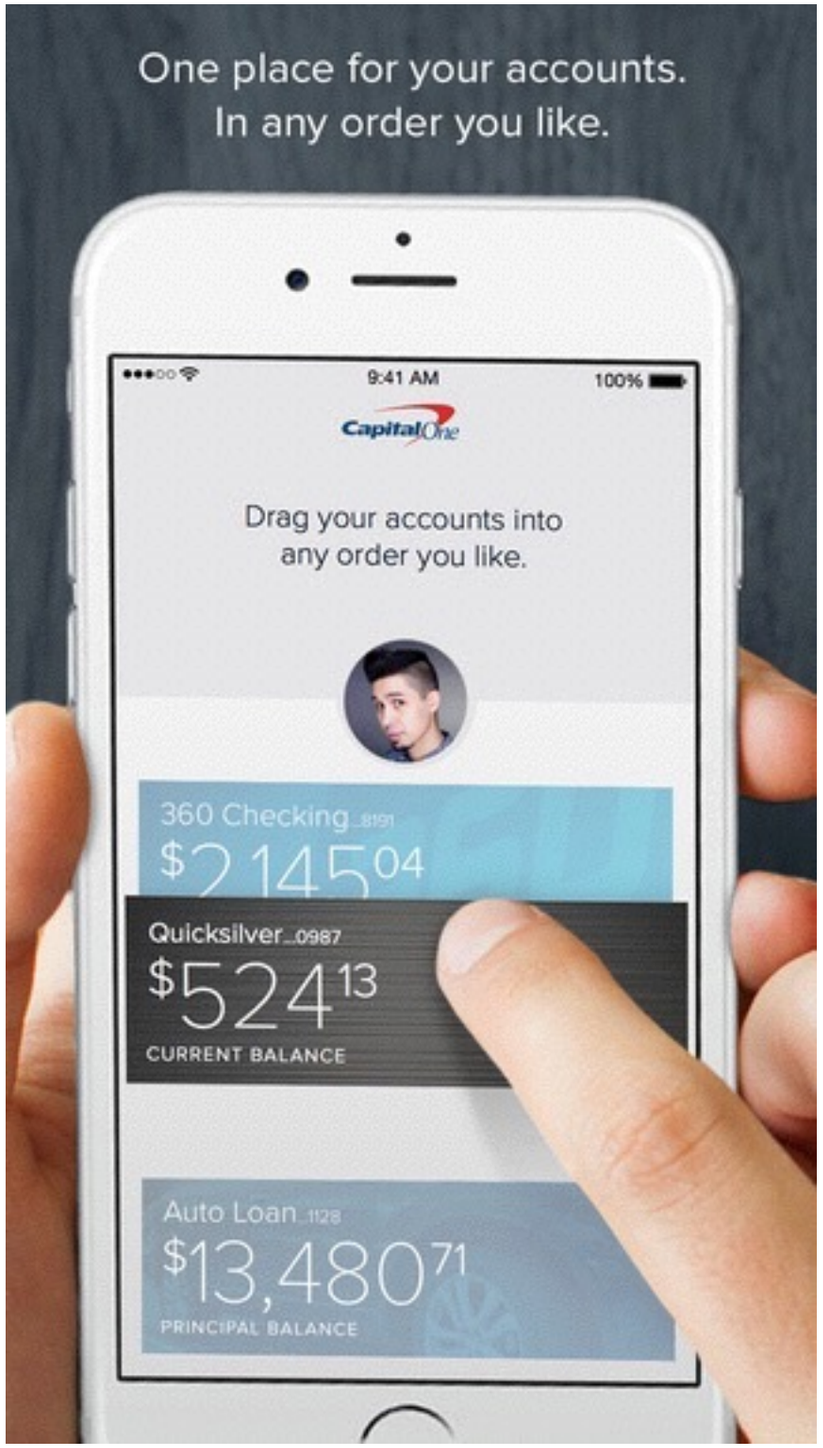
How to Use **Covariance** and **Contravariance** to Build *Flexible* and *Robust* Programs

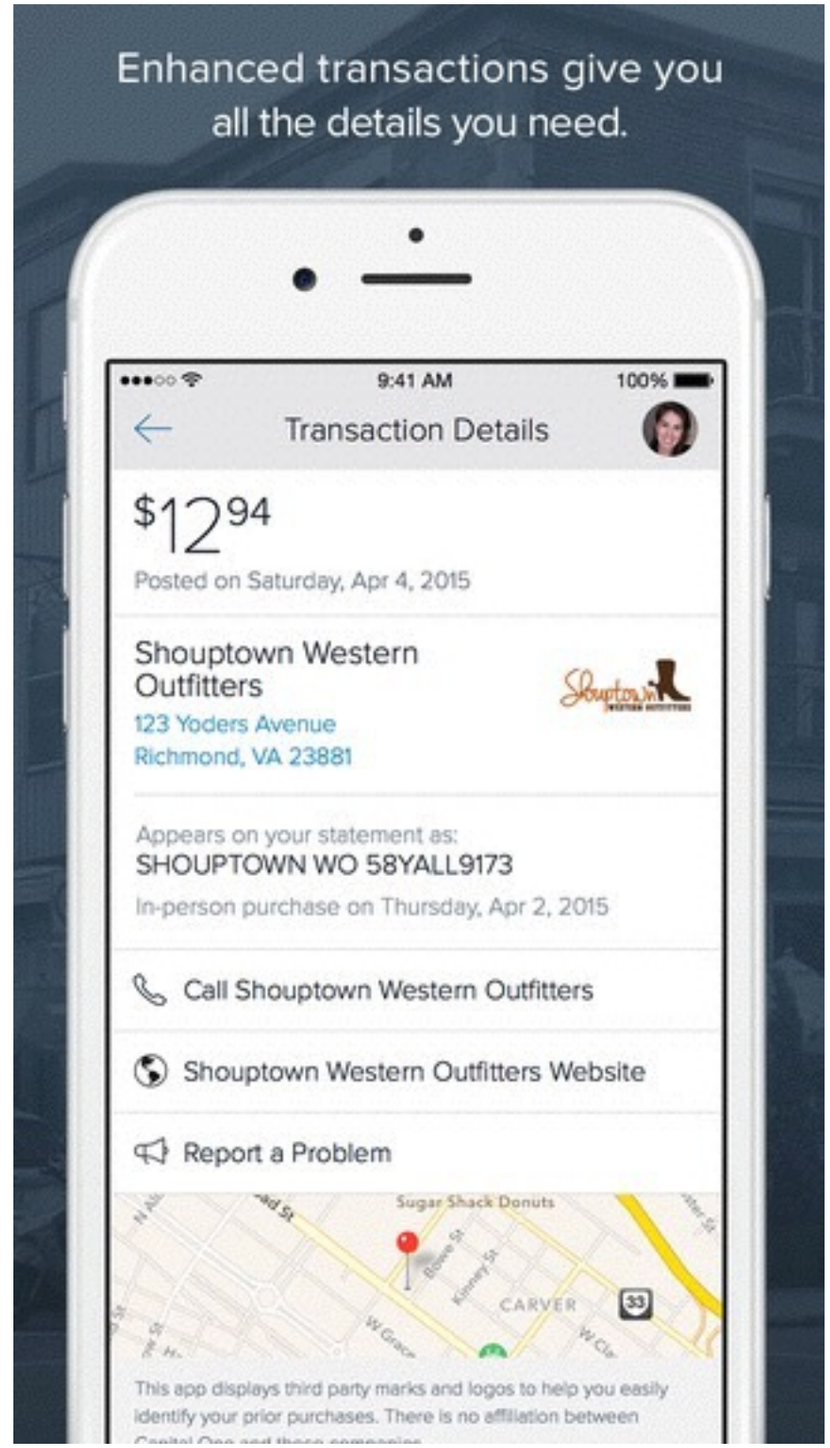
Jimmy Sambuo

What is **Covariance** and **Contravariance**?

C# 4.0

Generic co- and contravariance


One place for your accounts.
In any order you like.
9:41 AM
100%
Capital One
Drag your accounts into
any order you like.
360 Checking ...8191
$2,145.04
Quicksilver ...0987
$524.13
CURRENT BALANCE
Auto Loan ...1128
$13,480.71
PRINCIPAL BALANCE


Enhanced transactions give you
all the details you need.
9:41 AM
100%
Transaction Details
$12.94
Posted on Saturday, Apr 4, 2015
Shouptown Western
Outfitters
123 Yoders Avenue
Richmond, VA 23881
Appears on your statement as:
SHOUPTOWN WO 58YALL9173
In-person purchase on Thursday, Apr 2, 2015
Call Shouptown Western Outfitters
Shouptown Western Outfitters Website
Report a Problem
Sugar Shack Donuts
CARVER
This app displays third party marks and logos to help you easily
identify your prior purchases. There is no affiliation between

- Conversions between function types are supported, exhibiting covariance in function result types and contravariance in function parameter types.

  For example, it is legal to assign a function of type `Any -> Int` to a variable of type `String -> Any`. (19517003)

Swift 2.1

What are you talking about?

How does it benefit me?

Potential issues

```
String name = "Jimmy";
```

SOLID

Liskov Substitution Principle

```
String name = "Jimmy";
Object objName = name;
```

```java
String name = "Jimmy";
Object objName = name;
objName.hashCode();
```

```java
String name = "Jimmy";
Object objName = name;
objName.hashCode();
objName = "Sambuo";
```

```java
String name = "Jimmy";
Object objName = name;
objName.hashCode();
objName = "Sambuo";
objName = 100;
```

```
String[] names = new String[10];
```

```
String[] names = new String[10];
Object[] objects = names;
```

```
String[] names = new String[10];
Object[] objects = names;
objects[0] = "Homer";
```

```
String[] names = new String[10];
Object[] objects = names;
objects[0] = "Homer";
objects[1] = 42;
```

```
String[] names = new String[10];
Object[] objects = names;
objects[0] = "Homer";
objects[1] = 42;
```

Build Successful

```
String[] names = new String[10];
Object[] objects = names;
objects[0] = "Homer";
objects[1] = 42;
```

Exception in thread "main" java.lang.ArrayStoreException: java.lang.Integer

```
String[] names = new String[10];
Object[] objects = names;
objects[0] = "Homer";
objects[1] = 42;
```

Exception in thread "main" java.lang.ArrayStoreException: java.lang.Integer

```
String[] names = new String[10];
Object[] objects = names;
objects[0] = "Homer";
objects[1] = 42;
```

Exception in thread "main" java.lang.ArrayStoreException: java.lang.Integer

```
String[] names = new String[10];
Object[] objects = names;
objects[0] = "Homer";
objects[1] = 42;
```

Exception in thread "main" java.lang.ArrayStoreException: java.lang.Integer

```
String[] names = new String[10];
Object[] objects = names;
objects[0] = "Homer";
objects[1] = 42;
```

Exception in thread "main" java.lang.ArrayStoreException: java.lang.Integer

String[]

is *not* a

Object[]

String

is a

Object

Why?

Designed for Type Safety

```
String[] names = new String[10];
Object[] objects = names;
objects[0] = "Homer";
objects[1] = 42;
```

```
ArrayList<String> names = new ArrayList<String>();
ArrayList<Object> objects = names;
objects.add("Homer");
objects.add(42);
```

```java
ArrayList<String> names = new ArrayList<String>();
ArrayList<Object> objects = names;
objects.add("Homer");
objects.add(42);
```

error: incompatible types: ArrayList<String> cannot be converted to ArrayList<Object>

Variance

~

~

Types

definables

definable (adj.) (usu. neg.) A contrasts, easily defined, sees, suggest, possible. **definably** adv. [Latin: related to **FINE**]

■ **Usage** See note at *definitive*.

definite article *n.* the word (*the* English) preceding a noun and imply a specific instance.

definition /ˌdefɪˈnɪʃ(ə)n/ *n.* **1 a** defi **b** statement of the meaning of a etc. **2** distinctness in outline, esp photographic image. [Latin: rela DEFINE]

definitive /dɪˈfɪnɪtɪv/ *adj.* **1** (answer, verdict, etc.) decisive, ditional, final. **2** (of a book etc

Variance

Variance?

Variance

Variance

Vary: Change / be different

Variance

Vary: Change / be different

-ance: State, action, or quality

Variance

The quality of how things are changing

Variant

Variant

A thing that varies

Covariance
Contravariance
Invariance

Covariance

Co-: With / Together

Covariant

A thing that varies with

Co-

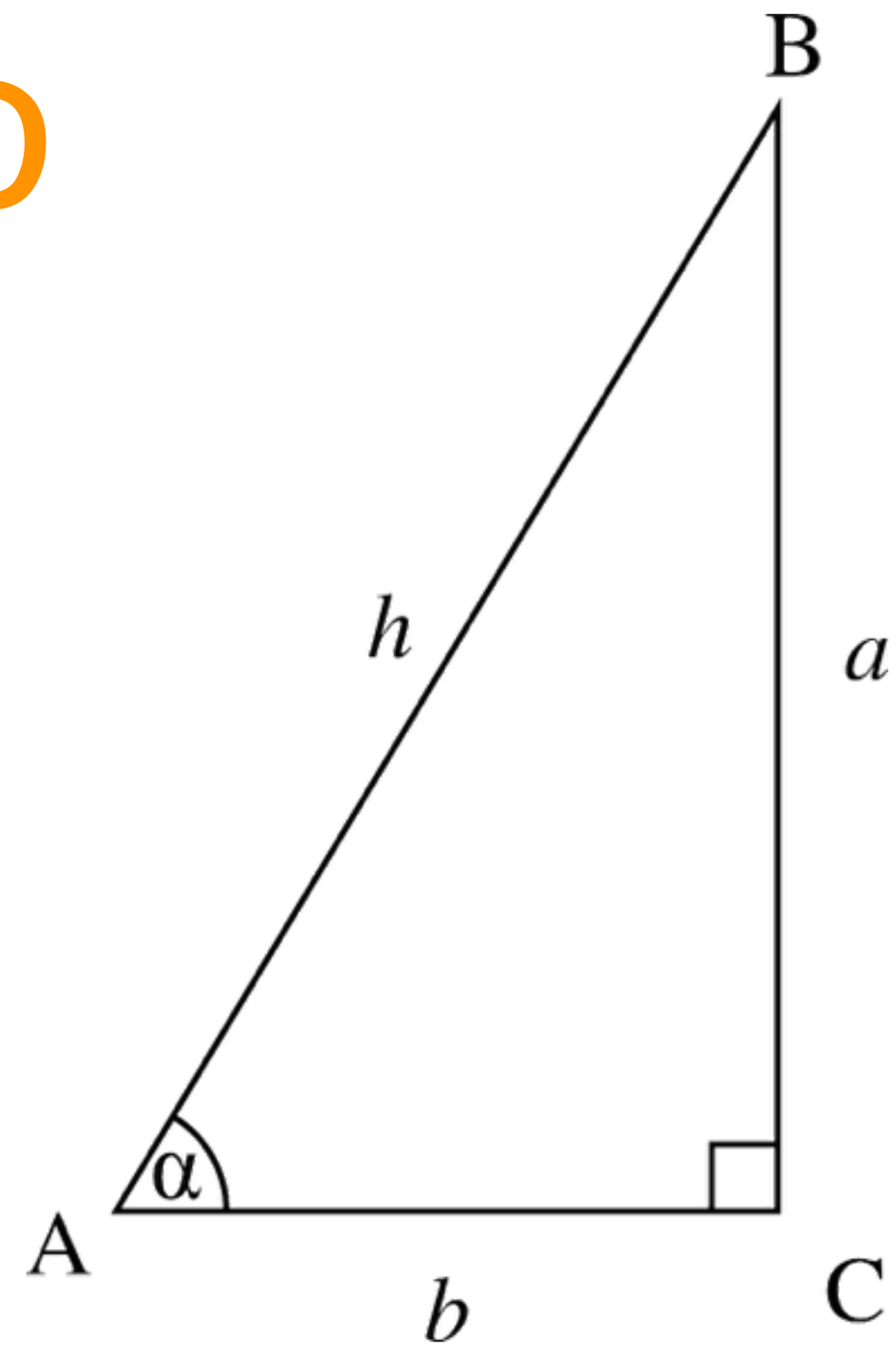
Co-: Opposite / Complement

Co

Sine

Tangent

Secant

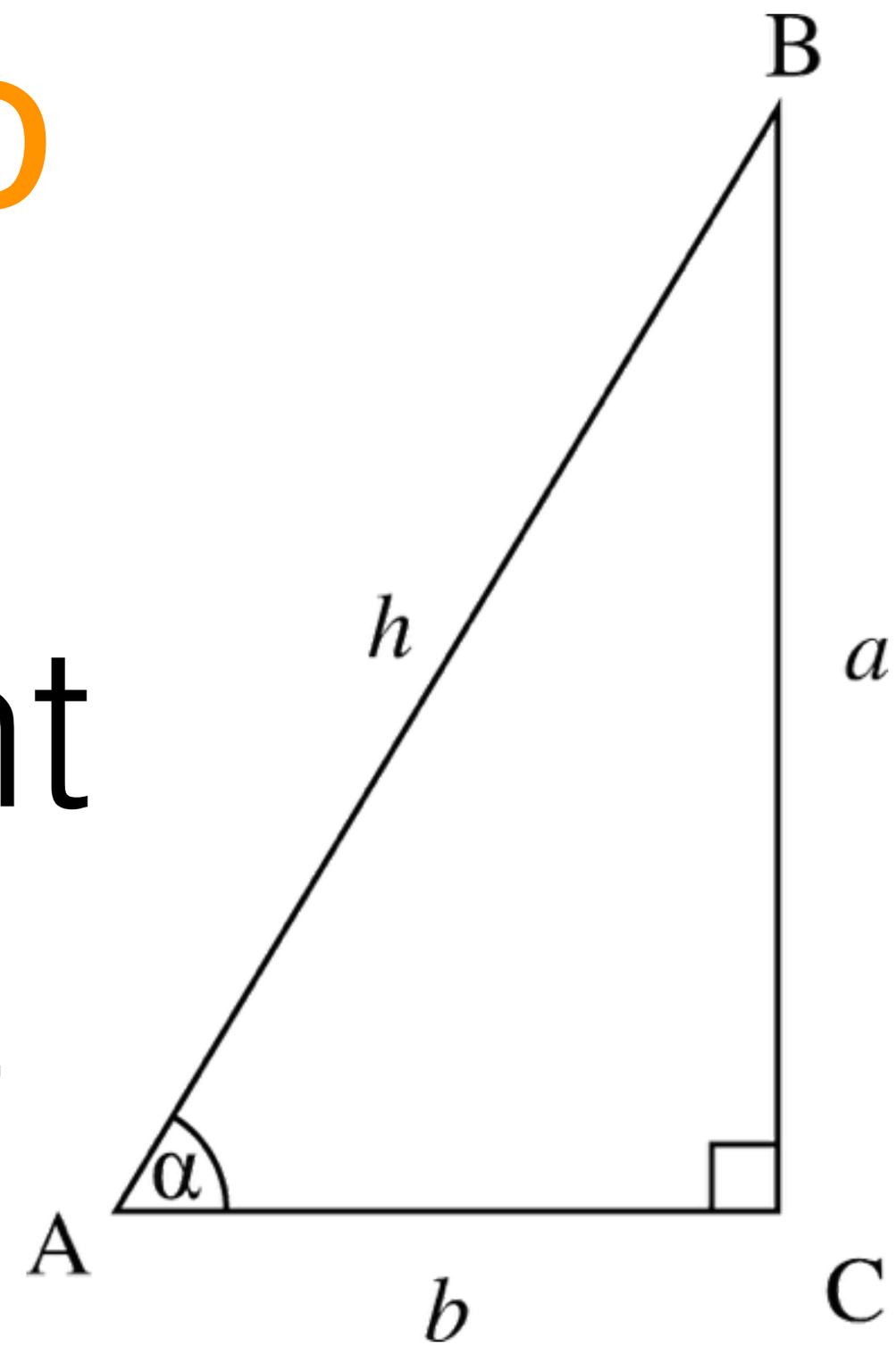


Co

Cosine

Cotangent

Cosecant



Functor

Monad

Cofunctor

Comonad

Covariance

Co-: With / Together

Contravariance

Contra-: Against / Opposite

Contravariant

A thing that varies against

Covariance

Contravariance

Invariance

Invariance

In-: Not

Invariant

A thing that does not vary

Covariance
Contravariance
Invariance

Vary?

Relationship?

is-a

<:

Cat <: Animal

Banana <: Fruit

String <: Object

Subtype Relationship

Type Polymorphism

One way

Transitive

Transitive

Bulldog <: Dog <: Animal

Bulldog <: Animal

Vary?

Parametric Polymorphic Types

Parametric Polymorphic

Generics

Parametric
Polymorphic

Generics

Parameterized Types

List<T>
Future<T>
IO<T>
Async<T>
Maybe<T>
Func<T,U>
IEnumerable<T>

Basket<Apple>

Apple <: Fruit

Basket<Apple>

<:

Basket<Fruit>

```java
ArrayList<String> names = new ArrayList<String>();
ArrayList<Object> objects = names;
objects.add("Homer");
objects.add(42);
```

error: incompatible types: ArrayList<String> cannot be converted to ArrayList<Object>

Why?

ArrayList<T>

!=

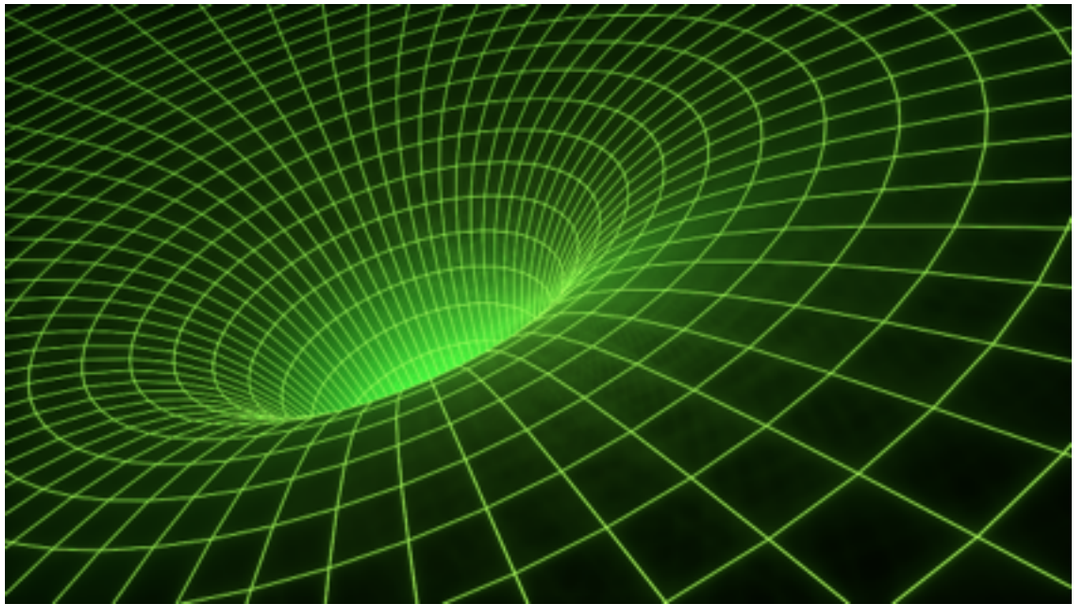
Basket<T>

The secret is…

Sources and Sinks

Source

Sink



Sources - Read-only
Sinks - Write-only

Sources - Covariant
Sinks - Contravariant

```
Basket<Fruit> b =
            appleBasket;
```

Covariance

```
Basket<Fruit> b =
            appleBasket;
Fruit f = b.get();
```

Covariance

```
Basket<Fruit> b =
            appleBasket;
Fruit f = b.get();
b.put(Banana());
```

Covariance

```java
ArrayList<String> names = new ArrayList<String>();
ArrayList<Object> objects = names;
objects.add("Homer");
objects.add(42);
```

error: incompatible types: ArrayList<String> cannot be converted to ArrayList<Object>

Array<T>

Invariant

Sources - Covariant

Mix - Invariant

Sinks - Contravariant

Contravariance

```
Zoo<Monkey> z =
        Zoo<Animal>();
```

Contravariance

```
Zoo<Monkey> z =
        Zoo<Animal>();
z.add(monkey);
```

Contravariance

```
Zoo<Monkey> z =
        Zoo<Animal>();
z.add(monkey);
Monkey a = z.take();
```

Contravariance

Let’s see some code

C#
Scala
OCaml
Java
Kotlin

Haskell

Haskell

- No Subtypes

Haskell

- No Subtypes
- Ad hoc Polymorphism

Haskell

- No Subtypes
- Ad hoc Polymorphism
  - Type class

Scala / OCaml

Covariant	Contravariant
+	-

C# / Kotlin

Covariant	Contravariant
out	in

Contravariant goes in input?

Input Covariant

```
fruitBasket.put(Apple());
```

Perspective

Caller:
Input Covariant

```
fruitBasket.put(Apple());
```

Callee:
Input Contravariant

```
void put(T thing) {
    //...
}
```

Callee: Covariant

```
Fruit take() {
    return Apple();
}
```

Caller: Contravariant

```
Fruit fruit =

    appleBasket.take();
```

Receive

Contravariant

Transmit

Covariant

Robustness Principle

Variance reverses based on Perspective

Basket in C#

```
interface Basket<T> {
 T get();
 void put(T thing);
}
```

Invariant

```
interface Basket<out T> {
 T get();
 void put(T thing);
}
```

Covariant

```
interface Basket<out T> {
 T get();
 void put(T thing);
}
```

Invalid variance: The type parameter 'T' must be contravariantly valid on 'Basket<T>.put(T)'. 'T' is covariant.

```
interface Basket<out T> {
 T get();
}
```

Covariant

```
interface Basket<in T> {
 void put(T thing);
}
```

Contravariant

Basket in Scala

```scala
trait Basket[T] {
  def take: T
  def put(thing: T)
}
```

Invariant

```scala
trait Basket[+T] {
  def take: T
  def put(thing: T)
}
```

Covariant

```scala
trait Basket[+T] {
  def take: T
  def put(thing: T)
}
```

covariant type T occurs in contravariant position in type T of value thing

```scala
trait Basket[+T] {
  def take: T
}
```

Covariant

```scala
trait Basket[-T] {
  def put(thing: T)
}
```

Contravariant

Covariant

```
Basket<Apple>
<:
Basket<Fruit>
```

Contravariant

```
Basket<Fruit>
<:
Basket<Apple>
```

Declaration-site variance annotations

Use-site variance annotations

Java

```
interface Basket<T> {
 T get();
 void put(T thing);
}
```

Invariant

Java Wildcards

Unbounded Wildcard

Unbounded Wildcard

```
Basket<?> b;
```

Unbounded Wildcard

```
Basket<?> b;
b = Basket<Fruit>();
```

Upper bound Wildcard Covariant

Upper bound Wildcard
Covariant

```
Basket<? extends Fruit> b;
```

Upper bound Wildcard Covariant

```
Basket<? extends Fruit> b;
b = Basket<Apple>();
```

Upper bound Wildcard Covariant

```
Basket<? extends Fruit> b;
b = Basket<Apple>();
Fruit fruit = b.get();
```

Upper bound Wildcard Covariant

```
Basket<? extends Fruit> b;
b = Basket<Apple>();
Fruit fruit = b.get();
b.put(Apple());
```

Upper bound Wildcard Covariant

```
Basket<? extends Fruit> b;
b = Basket<Apple>();
Fruit fruit = b.get();
b.put(Apple());
```

The method put(capture#4-of ? extends Fruit) in the type Basket<capture#4-of ? extends Fruit> is not applicable for the arguments (Apple)

Lower bound Wildcard Contravariant

Lower bound Wildcard Contravariant

```
Basket<? super Banana> b;
```

Lower bound Wildcard Contravariant

```
Basket<? super Banana> b;
b = Basket<Fruit>();
```

Lower bound Wildcard Contravariant

```
Basket<? super Banana> b;
b = Basket<Fruit>();
b.put(Banana());
```

Lower bound Wildcard Contravariant

```
Basket<? super Banana> b;
b = Basket<Fruit>();
b.put(Banana());
b.put(Apple());
```

The method put(capture#1-of ? super Banana) in the type Basket<capture#1-of ? super Banana> is not applicable for the arguments (Apple)

Lower bound Wildcard Contravariant

```
Basket<? super Banana> b;
b = Basket<Fruit>();
```

Lower bound Wildcard Contravariant

```
Basket<? super Banana> b;
b = Basket<Fruit>();
Object obj = b.get();
```

Lower bound Wildcard Contravariant

```
Basket<? super Banana> b;
b = Basket<Fruit>();
Object obj = b.get();
Banana banana = b.get();
```

Lower bound Wildcard Contravariant

```
Basket<? super Banana> b;
b = Basket<Fruit>();
Object obj = b.get();
Banana banana = b.get();
```

Type mismatch: cannot convert from capture#5-of ? super Banana to Banana

Pros: No vanity interfaces

Cons: Complexity

Compiler inferred variance

Summary

- Variance is generics as subtype
- Covariance (Source) - vary with
- Contravariance (Sink) - vary against
- Invariance (Source+Sink) - not vary
- declaration vs use-site variance annotations

Be contravariant
with your inputs
and covariant
with your output

Liskov Substitution Principle

Robustness Principle

Robust and Flexible

Thanks

Thanks

- Capital One

Thanks

- Capital One
  - We’re hiring in DC, Plano, SF, Chicago

Thanks

- Capital One
  - We’re hiring in DC, Plano, SF, Chicago
- LambdaConf organizers

Thanks

- Capital One
  - We’re hiring in DC, Plano, SF, Chicago
- LambdaConf organizers
- You!

Thanks!

Jimmy Sambuo
@jsambuo

Definitions

- http://www.oxfordlearnersdictionaries.com/definition/english/co_3
- http://www.oxfordlearnersdictionaries.com/definition/english/contra
- http://www.oxfordlearnersdictionaries.com/definition/english/in_6
- http://www.oxfordlearnersdictionaries.com/definition/english/vary
- http://www.oxfordlearnersdictionaries.com/definition/english/ance
- http://www.oxfordlearnersdictionaries.com/definition/english/ant_2
- http://www.oxfordlearnersdictionaries.com/us/definition/english/poly_2
- http://www.oxfordlearnersdictionaries.com/us/definition/english/ism_2

Images

- https://pixabay.com/en/punctuation-marks-question-mark-1019729/
- https://pixabay.com/en/children-win-success-video-game-593313/
- https://pixabay.com/en/definition-word-dictionary-text-390785/
- https://pixabay.com/en/right-angle-triangle-trigonometry-39887/
- https://pixabay.com/en/apples-basket-red-fruit-harvest-1114059/
- https://pixabay.com/en/still-life-fruits-pineapple-840008/
- https://pixabay.com/en/cat-box-predator-650770/
- https://pixabay.com/en/database-storage-data-storage-152091/
- https://pixabay.com/en/silhouette-faucet-hahn-drip-tiles-1312158/
- https://pixabay.com/en/wormhole-space-time-light-tunnel-739872/
- https://pixabay.com/en/water-drops-sink-dark-chrome-wet-219733/
- https://commons.wikimedia.org/wiki/File:C_Sharp_wordmark.svg
- https://commons.wikimedia.org/wiki/File:Swift_logo_with_text.svg