Kotlin (on Android)
No News is good News

Kasper Østerbye

Wednesday February 14th

What makes a good programming language

For whom

Researchers:

- The language support something hard in a convenient syntax
- The support is proven to be correct
- The support is efficient
- The compiler finds and reports if the features are not used correcty
- Nearly uninteresting if errors are not found until runtime.
- At least one of the above are novel (not done before in other languages)
- It is cool if it is useful too…

Developers:

- The language support something *useful* in a convenient syntax
- The language allows for third parties to write awesome libraries.
- The support is proven to be correct
- The support is efficient
- The compiler finds and reports if the features are not used correcty
- Errors found at runtime is so much better than not being able to write your code nicely.
- It has to be useful – cool is good too.

Kotlin at a glance

It is meant to be used on the java (and java-script) platforms.

- It can use all the existing Java libraries out of the box
- Kotlin classes can be called from Java (directly)

- It looks somewhat like Swift in its syntax

Your language experience?:

Java:

C#:

F#:

Javascript:

Scala:

C/C++:

...:

Practical

All the exercises and my demo will be on the online

try.kotlinlang.org

Google doc used in class: https://goo.gl/Kc28Hb

Not only classes can be top level

Link to try.kotlinlang.org

In Java you have classes as top level.

In Kotlin all declarations can be toplevel, in particular:

- Classes
- Variables (var)
- Constants/Values (val)
- Functions
- Objects
- …

The small things

var and val

- This is a return to old style. May have been annoyed in Java of the complicated way to declare constants:
  - public static final int MEANING = 42;
  - val MEANING = 24;

Null pointers

Kotlin has support for nullable types (and in particular non-nullable).

The practical usage is that you can write a function will never be called with a null reference!

var p : Person // not legal!
var pp : Person?

Look at examples, and the code demo’ed in class

Main points:

- Compile time check
- Focus on the call spot (which is where the problem is!)
- Smart checks (if and when statements)
- Possible to override a nullable (using !!)

Properties

This is close to religion for some (either way!)

In java terms a Property is a getter/setter pair.

```
kurt.setLastName(sonja.getLastName());
// kurt.lastName = sonja.lastName
somePerson.setName(null); // uups
```

Why do we make the fields private in the first place?

Properties from C# - I

```
public class Person {
  private readonly int id;
  private string name;

  public Person(int id, string name) {
    this.id = id;
    Name = name;
    }

  public int Id { get { return id; } }

  public string Name {
    get { return name; }
    set {
      if (value == null) throw new Exception();
      name = value;
      }
    }
  }
```

C# getter and setter

Now this is possible: kurt.LastName = sonja.LastName;

```
public class Person {
  public int Id { get; private set; }
  public string Name { get; set; }

  public Person(int id, string name) {
    this.id = id;
    Name = name;
    }

  }
```

Kotlin properties

Comes in two usage scenarios:

1. General classes where you would have getter setter pairs, and some setters which perhaps does something interesting
2. Read only “data classes”. These typically hold info from a database or from other external sources.

Link to code

Kotlin extension methods

Very similar to those in C#

In essence it allow you to add methods to existing classes.

The usage scenario is to allow third party developers to add methods to existing classes in a manner which seems ”natural”.

fun age(p : Person) { … }

Gives the call

Int a = age(p)

But with an extension you can write (extension property…)

Int a = p.age

Demo: Extend Array<String> with a method oddLength : Boolean

Extensions to AndroidStudio

Kotlin has gotten some attention because it allow you to access textView (and similar things) in a more convenient manner than in Java:

```
// Java
findViewById<TextView>(R.id.textView).setText ("Hello, world!")
```

```
//Kotlin
textView.setText("Hello, world!")
```

This is actually not done with ”kotlin extension”, but as a compiler plug-in for android studio.

Why cannot it not be done without changing the compiler?

The last goodies

First class functions

In language a function is called first class if it can be:

- Be stored in a variable
- Can be passed as parameter
- Can be the return value of an other function

Java is getting good at this on its own.

The typical usage scenario is in “call backs” and stream processing.

Call backs

```
btn = (Button)findViewById(R.id.firstButton);
btn.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v)    {
                        toast("Hello folks")
}});
```

Actually, the above is really a lot of stuff to say “when user clicks the btn, a toast should be shown (toast are a bit more advanced to start actually).

In Kotlin you can reduce this to:

```
    firstButton.setOnClickListener ({ v -> toast("Hello") } )
```

Which can be even further reduced:

```
    firstButton.setOnClickListener { toast("Hello") }
```

A silly axample

Using extension methods and functions

Assume we are making a game, where something has to happen with a certain probability.

In Java:

```
if ( Math.random() < 0.75 ) {
        toast(“Bingo”);
}
```

In Kotlin:

```
0.75 .perhaps {
        toast(“Bingo”);
}
```