Kotlin & android

Paweł Hajduk
https://about.me/pawelhajduk

Katowice, 19.08.2015

1.

Kotlin

- ✘ Created by JetBrains
- ✘ General purpose JVM language compiled to JVM byte code
- ✘ Statically typed
- ✘ Inspired by: C#, Scala, Groovy
- ✘ Open source

Interesting parts

- ✘ val & var e.g. `val size = list.size()`
- ✘ no `switch` statement (instead `when` statement with pattern matching)
- ✘ no `new` operator e.g. `val instance = MyClass()`
- ✘ by default all classes are final (`open` operator to allow inheritance)
- ✘ no static methods (companion object and package-level functions)
- ✘ ranges e.g. `for (i in 1..100) { ... }`
- ✘ extension methods

```
fun MyType.newMethod() {
  this // 'this' corresponds to the list
}
```

Data class

```java
public class Customer {
    private String name;
    private String email;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

Data class

```java
public class Customer {
    private String name;
    private String email;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

```java
@Override
    public boolean equals(Object o) {
        if(this == o) {
            return true;
        }
        if(o == null || getClass() != o.
getClass()) {
            return false;
        }

        Customer customer = (Customer) o;

        if(!name.equals(customer.name)) {
            return false;
        }
        return email.equals(customer.email);
    }

    @Override
    public int hashCode() {
        int result = name.hashCode();
        result = 31 * result + email.hashCode();
        return result;
    }
```

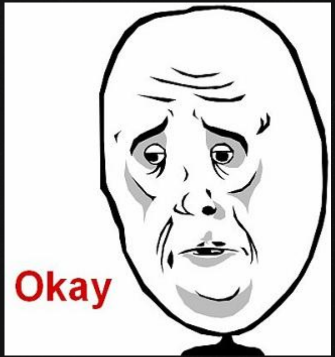
Data class

```java
public class Customer {
    private String name;
    private String email;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

```java
@Override
    public boolean equals(Object o) {
        if(this == o) {
            return true;
        }
        if(o == null || getClass() != o.
getClass()) {
            return false;
        }

        Customer customer = (Customer) o;

        if(!name.equals(customer.name)) {
            return false;
        }
        return email.equals(customer.email);
    }

    @Override
    public int hashCode() {
        int result = name.hashCode();
        result = 31 * result + email.hashCode();
        return result;
    }
```

Lombok?
AutoValue?

Write it again?



Data class

```
data class Customer(val name: String, val email: String)
```

- ✘ getters (and setters in case of var’s) for all properties
- ✘ equals
- ✘ hashCode
- ✘ toString
- ✘ others

I call it my billion-dollar mistake. It was the invention of the null reference in 1965.

Sir Charles Antony Richard Hoare

Null safety

In Kotlin the type system distinguishes between references that can hold null

Wrong:

```
var a: String = "abc"
a = null // compilation error
```

Correct:

```
var b: String? = "abc"
b = null // ok
```

Null safety

In Kotlin the type system distinguishes between references that can hold null

Wrong:

```
var a: String = "abc"
a = null // compilation error
```

```
val l = a.length()
```

✓

Correct:

```
var b: String? = "abc"
b = null // ok
```

```
val l = b.length() //variable 'b' can be null
```

✗

```
val length = if (b != null) b.length() else -1
```

Null safety

Safe calls

```
b?.length()
```

Returns b.length() if b is not null, and null otherwise.

The type of this expression is *Int*?

Null safety

Safe calls

```
b?.length()
```

Returns b.length() if b is not null, and null otherwise.

The type of this expression is *Int?*

Elvis operator

```
val l = b?.length() ?: -1
```

If the expression to the left of ?: is not null, the elvis operator returns it, otherwise it returns the expression to the right

!! operator

```
val l = b!!.length()
```

We know that b is not null for sure, or NPE will be thrown

Syntax

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        srl_tasks.setOnRefreshListener({ downloadData() })
        btn_add.setOnClickListener { startActivity<AddTaskActivity>() }
        downloadData()
    }
```

Syntax

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        srl_tasks.setOnRefreshListener({ downloadData() })
        btn_add.setOnClickListener { startActivity<AddTaskActivity>() }
        downloadData()
    }
```

no @Override
no return type
fun keyword

Syntax

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        srl_tasks.setOnRefreshListener({ downloadData() })
        btn_add.setOnClickListener { startActivity<AddTaskActivity>() }
        downloadData()
    }
```

Scala syntax
Nullable type

Syntax

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        srl_tasks.setOnRefreshListener({ downloadData() })
        btn_add.setOnClickListener { startActivity<AddTaskActivity>() }
        downloadData()
    }
```

Java style
no semicolon

Syntax

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        srl_tasks.setOnRefreshListener({ downloadData() })
        btn_add.setOnClickListener { startActivity<AddTaskActivity>() }
        downloadData()
    }
```

Syntax

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        srl_tasks.setOnRefreshListener({ downloadData() })
        btn_add.setOnClickListener { startActivity<AddTaskActivity>() }
        downloadData()
    }
```

srl_tasks

Android Extensions feature

Syntax

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        srl_tasks.setOnRefreshListener({ downloadData() })
        btn_add.setOnClickListener { startActivity<AddTaskActivity>() }
        downloadData()
    }
```

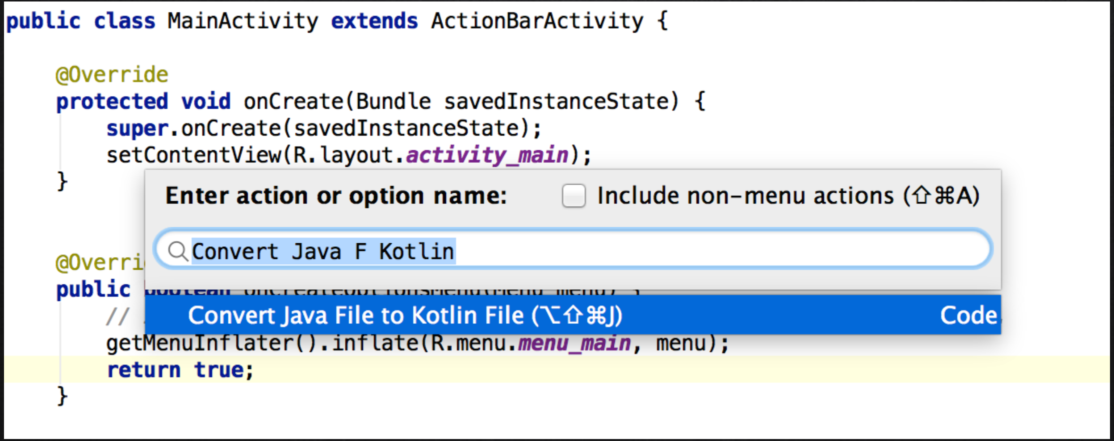
single parameter lambda

Syntax

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        srl_tasks.setOnRefreshListener({ downloadData() })
        btn_add.setOnClickListener { startActivity<AddTaskActivity>() }
        downloadData()
    }
```

Java Interop

- ✘ You can mix Kotlin & Java without any problems
- ✘ T! means "T or T?"
- ✘ Java class get by: `javaClass<MainActivity>()`
- ✘ Easy (one-way) conversion



```
public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }


    @Overri
    public
        //
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
```

Enter action or option name: ☐ Include non-menu actions (⇧⌘A)

Convert Java F Kotlin

Convert Java File to Kotlin File (⌥⇧⌘J) Code

Android Extensions

IDE plugin + gradle dependency

Synthetic properties:

```
import kotlinx.android.synthetic.<layout>.*
(...)
srl_tasks.setOnRefreshListener({ downloadData() })
```

Under the hood reference
http://kotlinlang.org/docs/tutorials/android-plugin.html#under-the-hood

Anko

- ✘ JetBrains library
- ✘ DSL to build Android views hierarchy

```
verticalLayout {
    val name = editText()
    button("Say Hello") {
        onClick { toast("Hello, ${name.text}!") }
    }
}
```

Anko

Intents in pure Kotlin

```
val intent = Intent(this, javaClass<SomeOtherActivity>())
intent.putExtra("id", 5)
intent.setFlag(Intent.FLAG_ACTIVITY_SINGLE_TOP)
startActivity(intent)
```

Anko

Intents in pure Kotlin

```kotlin
val intent = Intent(this, javaClass<SomeOtherActivity>())
intent.putExtra("id", 5)
intent.setFlag(Intent.FLAG_ACTIVITY_SINGLE_TOP)
startActivity(intent)
```

Anko

```kotlin
startActivity(intentFor<SomeOtherActivity>("id" to 5).singleTop())
```

Anko

Intents in pure Kotlin

```
val intent = Intent(this, javaClass<SomeOtherActivity>())
intent.putExtra("id", 5)
intent.setFlag(Intent.FLAG_ACTIVITY_SINGLE_TOP)
startActivity(intent)
```

Anko

```
startActivity(intentFor<SomeOtherActivity>("id" to 5).singleTop())
```

or without flags:

```
startActivity<SomeOtherActivity>("id" to 5)
```

Anko

Services

```
(NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE)
```

Is just available in variable

```
notificationManager
```

- ✘ `layoutInflater`
- ✘ `displayManager`
- ✘ `sensorManager`
- ✘ `vibrator`

Anko

Toasts

```
toast("Hi Mobile Silesia!")
toast(R.string.message)
longToast("Mobile Silesia long toast!")
```

THANKS!

Any questions?

Credits & resources

- ✘ http://kotlinlang.org
- ✘ http://kotlinlang.org/docs/tutorials/koans.html
- ✘ https://github.com/JetBrains/anko