Ubiquitous and Mobile Computing
CS 528: Kotlin and NFC

Matthew McMillan, JP Bulman, Matthew Kaminski, Weixi Liu, Chao Wang

Computer Science Dept.
Worcester Polytechnic Institute (WPI)
Kotlin

- **History**
  - First released in 2011
  - Developed by JetBrains

- **Motivation**
  - JVM is great
    - Runs everywhere! (no recompiling)
    - Android apps run on the JVM
    - A lot of libraries written for Java/JVM
  - Java syntax is verbose, repetitive
  - Want to leverage power and libraries of the JVM but use a cleaner and safer syntax
Kotlin: Issues Solved

- Kotlin is null-safe, any nullable variables must be explicitly marked
- *Type inference* means that types don’t have to be explicitly written out if it’s clear from context
- *Data classes* automatically implement some methods
- You can specify *default arguments* to functions
- You can provide *named arguments* to functions that take many parameters
- Many more small features
Kotlin-Typical use case

Tooling

- Android studio
- Eclipse
- IntelliJ IDE
Kotlin-Typical use case

Lyft

American Express

Wechat

Airbnb

Pinterest

Expedia
Kotlin-Typical use case

Kotlin vs Java

- Completely interoperable with Java
- More concise with fewer lines of code
- Safety prevents common programming mistakes
- Better support for functional programming
- Reduces errors and bugs
- Smarter and safer compiler
Kotlin

Basic Features

&

Comparison with Java
**Kotlin - Key word Val & Var**

**val** means an immutable value that does not change its value. However, **var** means variable, the value of a variable can change at any time.

```kotlin
package com.cwdoh.devfest2017

class Session {
    val speaker = "cwdoh"
    val title: String = "Kotlin: How it works"
    var room: Int? = null

    fun description() = "speaker's talk: 'title' at room room"
}
```
Kotlin - Var

`var` member itself can be read and modified by its class, i.e. there are *getter & setter* methods associated with it.

```java
class Session {
    var name = "cwdoh"
}

public final class Session {
    @NotNull
    private String name = "cwdoh";

    @NotNull
    public final String getName() {
        return this.name;
    }

    public final void setName(@NotNull String var1) {
        Intrinsics.checkParameterIsNotNull(var1, "<set-?>");
        this.name = var1;
    }
}
```
Kotlin - Val

`val` member itself only can be read by its class, i.e. there is **getter** method associate with it but not setter method. PS: we can modify subfields of a `val` member if the subfields are `var` type.

class Session {
  val name = "cwdoh"
}

public final class Session {
  @NonNull
  private final String name = "cwdoh";

  @NonNull
  public final String getName() {
    return this.name;
  }
}
Unlike other languages, Kotlin’s classes are limited to inherit by default.

```kotlin
class NotOpenedClass
open class OpenedClass
```

```java
public final class NotOpenedClass {
}

public class OpenedClass {
}
```
Kotlin - String Templates

String templates allow you to include variable references and expressions into strings.

```kotlin
val greeting = "Kotliner"
println("Hello $greeting")             // 1
println("Hello ${greeting.toUpperCase()}")    // 2
```

Hello Kotliner
Hello KOTLINER
Kotlin - Nullable & NotNull

Kotlin is null-safe. Variable types in Kotlin don’t normally allow the assignment of null.

```kotlin
var neverNull: String = "This can't be null"  // 1
neverNull = null                                 // 2

// Null can not be a value of a non-null type String
```
Kotlin - Nullable & NotNull

What if we need a variable can be null?

Declare it nullable by add “?” at the end of its type

```kotlin
val nullable: String? = "You can keep a null here"  // 3
nullable = null  // 4
```
public class Person {
    private String name;
    private String email;
    private int age;

    public Person(String name, String email, int age) {
        this.name = name;
        this.email = email;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public String getEmail() {
        return email;
    }

    public int getAge() {
        return age;
    }

    @Override
    public String toString() {
        return name + " - " + email + " - " + age;
    }

    @Override
    public int hashCode() {
        int result = 17;
        result = 31 * result + name.hashCode();
        result = 31 * result + email.hashCode();
        result = 31 * result + age;
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj != null && obj.getClass() == this.getClass()) {
            Person castObj = (Person) obj;
            if (this.name.equals(castObj.getName())) return false;
            if (this.email.equals(castObj.getEmail())) return false;
            if (this.age != castObj.getAge()) return false;
        }
        return false;
    }
}
Why Kotlin?

data class Person(val name: String, val email: String, val age: Int)
Near Field Communication (NFC)
What is NFC?

- Near-field Communication or NFC is a short-range radio technology that operates with data transfers of up to 424 kilobits per second.
- NFC communication is triggered when two NFC-compatible devices are brought within close proximity, around four centimeters.

[https://www.oracle.com/technical-resources/articles/javame/nfc.html](https://www.oracle.com/technical-resources/articles/javame/nfc.html)
What is NFC?

<table>
<thead>
<tr>
<th></th>
<th>NFC</th>
<th>RFID</th>
<th>IrDa</th>
<th>Bluetooth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup time</td>
<td>&lt;0.1 ms</td>
<td>&lt;0.1 ms</td>
<td>~0.5 sec</td>
<td>~6 sec</td>
</tr>
<tr>
<td>Range</td>
<td>Up to 10 cm</td>
<td>Up to 3 m</td>
<td>Up to 5 m</td>
<td>Up to 30 m</td>
</tr>
<tr>
<td>Usability</td>
<td>Human centric</td>
<td>Item centric</td>
<td>Data centric</td>
<td>Data centric</td>
</tr>
<tr>
<td></td>
<td>Easy, intuitive, fast</td>
<td>Easy</td>
<td>Easy</td>
<td>Medium</td>
</tr>
<tr>
<td>Selectivity</td>
<td>High, given, security</td>
<td>Partly given</td>
<td>Line of sight</td>
<td>Who are you?</td>
</tr>
<tr>
<td>Use cases</td>
<td>Pay, get access, share, initiate service, easy set up</td>
<td>Item tracking</td>
<td>Control &amp; exchange data</td>
<td>Network for data exchange, headset</td>
</tr>
<tr>
<td>Consumer experience</td>
<td>Touch, wave, simply connect</td>
<td>Get information</td>
<td>Easy</td>
<td>Configuration needed</td>
</tr>
</tbody>
</table>
What is NFC?

- The NFC Standard defines three types of communication:
  - Peer-to-peer mode which allows two NFC-enabled devices to exchange information between each other.
  - Read/write mode is a one way data transmission commonly used for passive NFC devices like NFC tags
  - Card emulation allows the NFC device to be used like a smart or contactless credit card in order to make payments or tap into public transport systems.

- [https://www.androidauthority.com/what-is-nfc-270730/](https://www.androidauthority.com/what-is-nfc-270730/)
NFC - History

- Inspired by RFID
  - Charles Walton - 1983
- First appearance in 2002
  - Sony and NXP semiconductors
- 2004
  - Rise of mobile phones
  - Companies start putting in NFC chips
  - Doesn’t have much use yet
  - Mostly unidirectional
NFC - History (cont’d)

- 2006
  - Usage and abilities increase
  - Users can now receive music, photos, media, etc.
- 2009
  - Peer to peer (P2P) communication
  - Bidirectional transmissions
    - Users can now receive and *send* data with NFC
NFC - Issues that it solves

- The need to transfer sensitive data quickly, securely, and wirelessly over short distances
  - Swiping cards/entering PIN’s have visibility risks
  - Much more room for error
    - Forgotten PIN, broken magnetic strip, etc
- Better proof of purchase
  - Receipts are not reliable
    - Can be forged, easily lost, destroyed, etc.
NFC - Typical use case

- Making payments with a phone
- Sharing images
- Certain fitness devices
- Finding your location more precisely
NFC - Real world example

- Google Pay
- Apple Pay
- YubiKey 2FA
- Android Beam File share
- Nike NFC jerseys
NFC Demo
NFC

- How it works
  - Peer-to-peer NFC Messages are structured in the **NDEF format** (NFC Data Exchange Format)
  - Command-response and read-write NFC messages are use the **APDU format** (application protocol data unit)
  - Some APDU messages are compatible with devices using the NDEF specification

<table>
<thead>
<tr>
<th>Command APDU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command APDU Header</td>
</tr>
<tr>
<td>CLA</td>
</tr>
</tbody>
</table>

CLA: Class byte (command-ID), INS: Instruction, P1, P2: Parameter, Lc: Length of command data, Data: Command data, Le: Length of expected data

<table>
<thead>
<tr>
<th>Response APDU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
</tr>
</tbody>
</table>

Data: Response data, SW1, SW2: Status Word
NFC

- Use `android.nfc.NfcAdapter` to facilitate communications
  - supports NDEF and APDU message transmission
- Depending on what you are planning on doing, code may be vastly different
  - Peer to peer, card reading, and read-write modes have inherently different security, device, and data requirements
NFC

- Different operations require different levels of programming
  - Many high-level libraries are available for simple NFC tag communication
    - [https://github.com/Rgghgh/NfcActivity](https://github.com/Rgghgh/NfcActivity)
  - Peer-to-peer large file transfer can be done using the middle-level Android Beam API
    - [https://developer.android.com/training/beam-files](https://developer.android.com/training/beam-files)
  - Complex NFC tab interactions, such as YubiKey OTP and HMAC-SHA1 require low-level manually-crafted nfc commands
NFC

- Code Example for Android Tag communication using **com.rgghgh.nfcactivity**:
  - [https://github.com/Rgghgh/NfcActivity](https://github.com/Rgghgh/NfcActivity)

```java
import com.rgghgh.nfcactivity.NfcActivity;
import com.rgghgh.nfcactivity.NfcConnection;
import com.rgghgh.nfcactivity.NfcTester;

public class MainActivity extends NfcActivity {
```
@Override
protected void onStart()
{
    super.onStart();
    runNfcTest();
}
public void runNfcTest()
{
    // NfcTester object usage:
    NfcTester tester = new NfcTester(this);
    if(!tester.hasNfc()){
        Toast.makeText(getApplicationContext(),"Device does not support NFC!",Toast.LENGTH_LONG).show();
        return;
    }
    if(!tester.isNfcEnabled()) {
        Toast.makeText(getApplicationContext(),"Device NFC is not enabled!",Toast.LENGTH_LONG).show();
    }
}
@Override
public void onNfcStart(NfcConnection conn)
{
    try
    {
        String id = conn.getTagId(); // get unique tag id
        String data = conn.read(); // get string data from tag
        // write website link to NFC Tag
        conn.writeUri("https://example.com");
        // make tag "read only"
        conn.makeReadOnly();
    }
    catch (Exception e)
    {
        Toast.makeText(getApplicationContext(), e.toString(), Toast.LENGTH_LONG).show();
    }
}
Questions?
References

- https://github.com/Rgghgh/NfcActivity
- https://developer.android.com/training/beam-files
- https://www.androidauthority.com/what-is-nfc-270730/
- https://www.oracle.com/technical-resources/articles/javame/nfc.html
- http://www.nfcnearfieldcommunication.org/history.html
- https://medium.com/til-kotlin/explanation-hey-kotlin-how-it-works-c98da63c59b0