THE
DEVELOPER'S
CONFERENCE

@PARCELIZE

Parcelables que você vai gostar de implementar

O QUE É SERIALIZAÇÃO?

"É o processo de tradução de estruturas de dados ou estado de objeto em um formato que possa ser armazenado e reconstruído posteriormente no mesmo ou em outro ambiente computacional."

POR QUE É TÃO IMPORTANTE NO ANDROID?

THE DEVELOPER'S CONFERENCE

- Activities só recebem parâmetros através dos Intents
- Intents só suportam dados simples ou serializáveis
- Cada rotação de tela depende de serialização para manter o estado

TIPOS RELEVANTES DE SERIALIZAÇÃO

THE DEVELOPER'S CONFERENCE

- Serializable
- Parcelable

java.io.SERIALIZABLE

THE DEVELOPER'S CONFERENCE

- Interface padrão de serialização do Java
- Interface de marcação
- Usa *reflection*

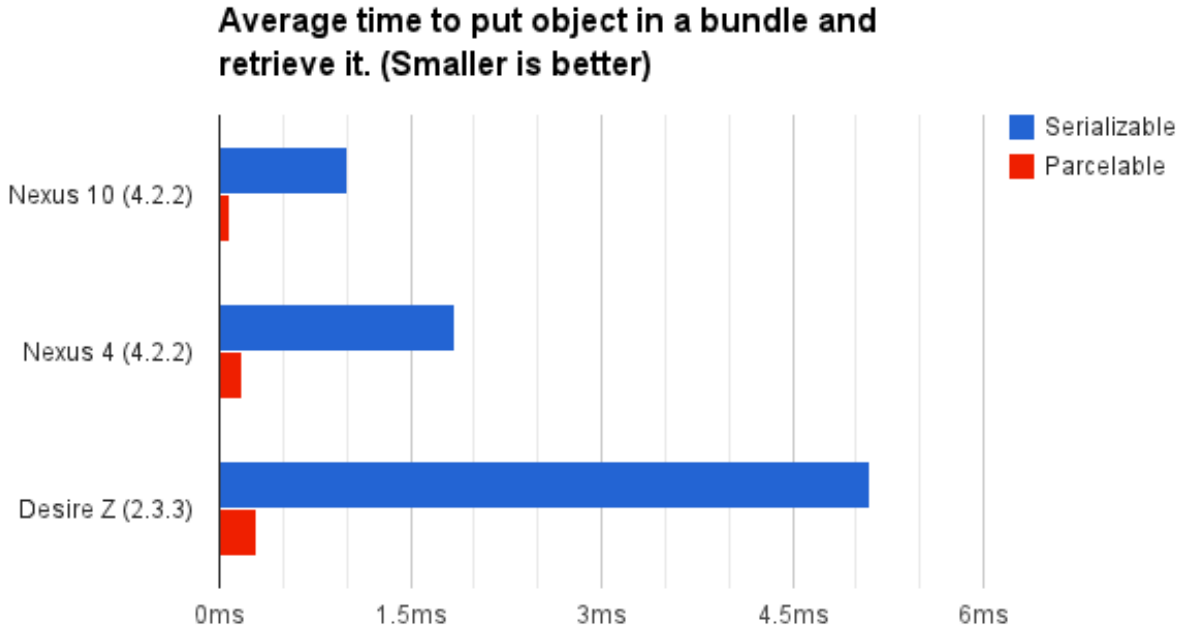
PARCELABLE

- Interface presente na SDK do Android
- Feita pra não usar *reflection*
- Implementação manual

SERIALIZABLE VS PARCELABLE – FIGHT!

THE DEVELOPER'S CONFERENCE

- Performance



Parcelable wins

SERIALIZABLE VS PARCELABLE – FIGHT!

THE DEVELOPER'S CONFERENCE

- Esforço

```
data class Endereco(
    val logradouro: String,
    val numero: String,
    val cep: String,
    val bairro: String,
    val cidade: String,
    val uf: String
)
```

Entidade

```
import java.io.Serializable

data class Endereco(
    val logradouro: String,
    val numero: String,
    val cep: String,
    val bairro: String,
    val cidade: String,
    val uf: String
) : Serializable
```

Serializable

```
import android.os.Parcel
import android.os.Parcelable

data class Endereco(
    val logradouro: String,
    val numero: String,
    val cep: String,
    val bairro: String,
    val cidade: String,
    val uf: String
) : Parcelable {
    constructor(parcel: Parcel) : this(
        parcel.readString().toString(),
        parcel.readString().toString(),
        parcel.readString().toString(),
        parcel.readString().toString(),
        parcel.readString().toString(),
        parcel.readString().toString()
    )
    override fun writeToParcel(parcel: Parcel, flags: Int) {
        parcel.writeString(logradouro)
        parcel.writeString(numero)
        parcel.writeString(cep)
        parcel.writeString(bairro)
        parcel.writeString(cidade)
        parcel.writeString(uf)
    }
    override fun describeContents(): Int { return 0 }
    companion object CREATOR : Parcelable.Creator<Endereco> {
        override fun createFromParcel(parcel: Parcel): Endereco {
            return Endereco(parcel)
        }
        override fun newArray(size: Int): Array<Endereco?> {
            return arrayOfNulls(size)
        }
    }
}
```

Parcelable

Serializable wins

E ASSIM FICAMOS POR ANOS...

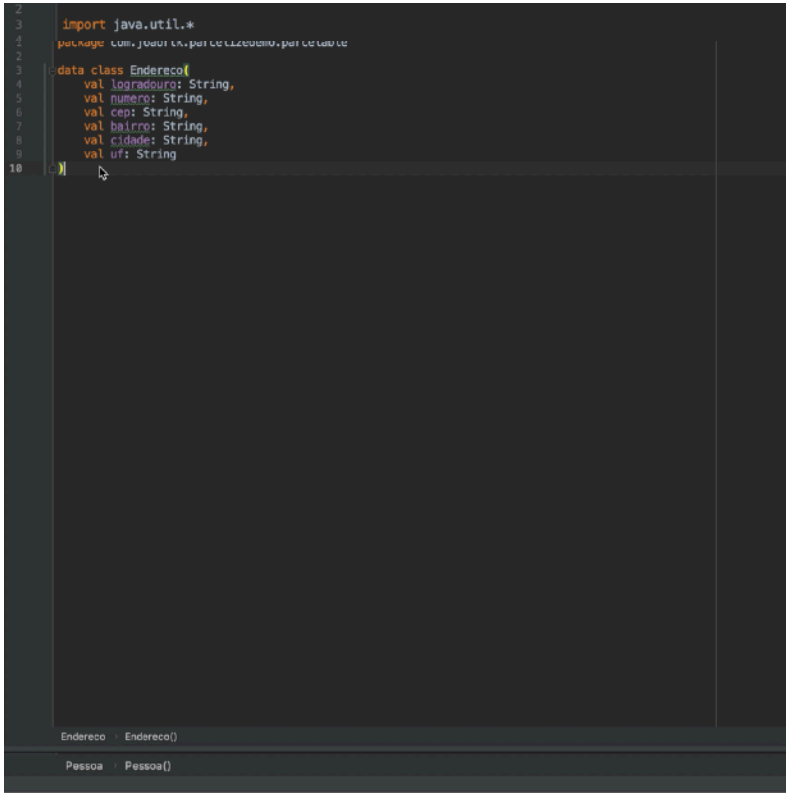
THE DEVELOPER'S CONFERENCE

GOOGLE TENTANDO DAR UMA MÃOZINHA...

THE DEVELOPER'S CONFERENCE

Android Studio ajuda bastante...

...mas não resolve tudo sozinho



EIS QUE SURGE O PARCELIZE

PARCELIZE - PRA ACABAR COM A DISCUSSÃO

- Plugin oficial criado pela JetBrains com suporte do Google
- Faz parte do Kotlin Android Extensions
- Gera código eficiente
- Não gera classes extras
- Disponível como *stable* a partir do Kotlin extension 1.3.40 e suportado como não-experimental no AS a partir da 1.3.60

PARCELIZE - SETUP

THE DEVELOPER'S CONFERENCE

- Kotlin Gradle Plugin atualizado

```
ext.kotlin_version = '1.3.60'
dependencies {
    classpath “org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version”
[…]
}
```

- Adicionar kotlin extensions

```
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: ‘kotlin-android-extensions’
```

- Adicionar kotlin extensions

```
androidExtensions {
    features = ["parcelize"]
}
```

PARCELIZE – COMO USAR

THE DEVELOPER'S CONFERENCE

- Basta adicionar a anotação *@Parcelize* à classe que estende de *Parcelable*

```
import android.os.Parcelable
import kotlinx.android.parcel.Parcelize

@Parcelize
data class Endereco(
    val logradouro: String,
    val numero: String,
    val cep: String,
    val bairro: String,
    val cidade: String,
    val uf: String
) : Parcelable
```

PARCELIZE - FUNCIONAMENTO

THE DEVELOPER'S CONFERENCE

- Tipos básicos

Entidade

```
@Parcelize
class Basic(val aByte: Byte,
            val aShort: Short,
            val anInt: Int,
            val aLong: Long,
            val aFloat: Float,
            val aDouble: Double,
            val aBoolean: Boolean,
            val aString: String) : Parcelable
```

Código gerado

```
public void writeToParcel(@NotNull Parcel parcel,
                          int flags) {
   parcel.writeByte(this.aByte);
   parcel.writeInt(this.aShort);
   parcel.writeInt(this.anInt);
   parcel.writeLong(this.aLong);
   parcel.writeFloat(this.aFloat);
   parcel.writeDouble(this.aDouble);
   parcel.writeInt(this.aBoolean);
   parcel.writeString(this.aString);
}
```

PARCELIZE - FUNCIONAMENTO

THE DEVELOPER'S CONFERENCE

- Enums

Entidade

```
enum class State {
    ON, OFF
}

@Parcelize
class PowerSwitch(var state: State) : Parcelable
```

Código gerado

```
public void writeToParcel(@NotNull Parcel parcel,
                          int flags)
{
   parcel.writeString(this.state.name());
}
```

PARCELIZE – FUNCIONAMENTO

THE DEVELOPER'S CONFERENCE

- Tipos suportados automaticamente:
  - Primitivos
  - *String e Charsequence*
  - *Exception*
  - *Serializables e Parcelables*
  - Coleções *(List, Set, Map)*
  - *Arrays* de tipos suportados
  - *Nullable* de tipos suportados

PARCELIZE - FUNCIONAMENTO

THE DEVELOPER'S CONFERENCE

- Para demais tipos, um *Parceler* deve ser criado:

```
object DateParceler : Parceler<Date> {
    override fun create(parcel: Parcel) = Date(parcel.readLong())
    override fun Date.write(parcel: Parcel, flags: Int) = parcel.writeLong(time)
}
```

PARCELIZE - FUNCIONAMENTO

THE DEVELOPER'S CONFERENCE

- Para demais tipos, um *Parceler* deve ser criado:

```
@Parcelize
@TypeParceler<Date, DateParceler>
class Session(val title: String,
              val startTime: Date,
              val endTime: Date): Parcelable
```

```
@Parcelize
class Session(
    val title: String,
    val startTime: @WriteWith<DateParceler> Date,
    val endTime: @WriteWith<DateParceler> Date
) : Parcelable
```

PARCELIZE – INFORMAÇÕES ADICIONAIS

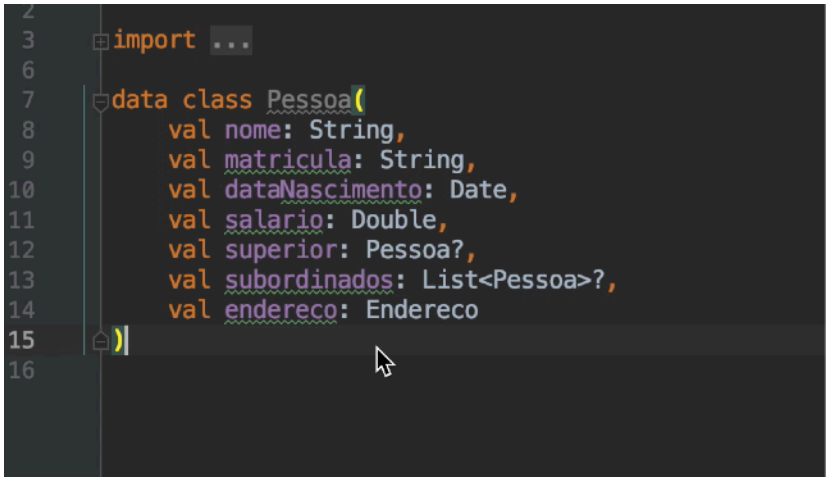
THE DEVELOPER'S CONFERENCE

- Todas propriedades precisam ser declaradas no construtor principal
- Podem ser ***var*** ou ***val***, ***public*** ou ***private***
- Não suporta **classes abstratas** nem ***sealed classes***
- É possível ignorar propriedades com ***@IgnoredOnParcel***

PRA ENCERRAR

THE DEVELOPER'S CONFERENCE

- Acabaram as desculpas pra não usar *Parcelables!*



THE DEVELOPER'S CONFERENCE

Obrigado!

João Rutkoski

joaortk@gmail.com

linkedin.com/in/joaortk

github.com/joaortk

REFERÊNCIAS

- https://kotlinlang.org/docs/tutorials/android-plugin.html
- https://medium.com/@BladeCoder/a-study-of-the-parcelize-feature-from-kotlin-android-extensions-59a5adcd5909
- https://pt.wikipedia.org/wiki/Serializa%C3%A7%C3%A3o