Les Intent & Activités

Hakim MOKEDDEM
École nationale Supérieure d'Informatique

Plan

- Utilisation des Intent
  - Lancement des activités
  - Lancement des applications externes
  - Récupération du résultat d'un Intent
- Cycle de vie d'une activité
- Gestion du changement de configuration

Plan

- Utilisation des Intent
  - Lancement des activités
  - Lancement des applications externes
  - Récupération du résultat d'un Intent
- Cycle de vie d'une activité
- Gestion du changement de configuration

Qu'est ce qu'un Intent ?

Un composant Android pour lancer des opérations

Lancer une activité

Lancer une application externe

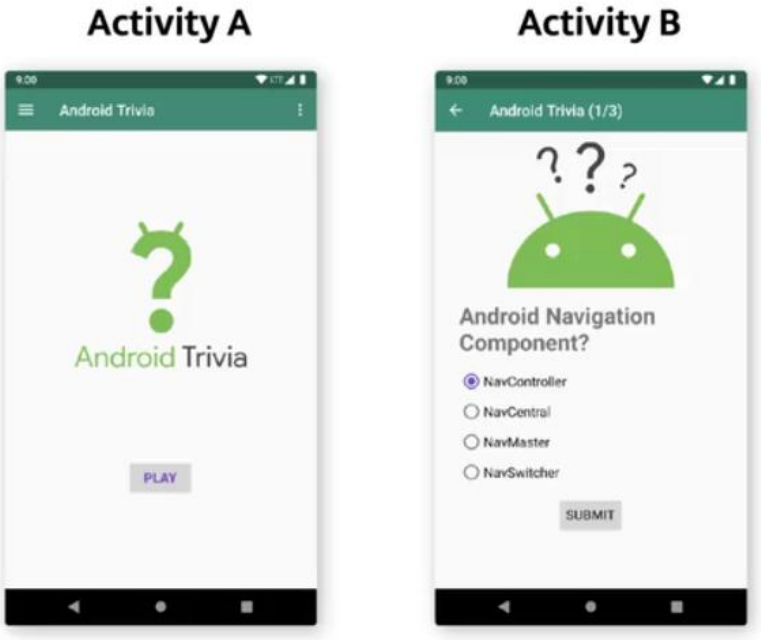
Plan

- Utilisation des Intent
  - Lancement des activités
  - Lancement des applications externes
  - Récupération du résultat d'un Intent
- Cycle de vie d'une activité
- Gestion du changement de configuration

Lancement d'une activité

Un Intent permet:

- De naviguer entre les différentes activités.
- D'envoyer des données entre les différentes activités.



Lancement d'une activité

Cas 1. Sans envoi de données.

```
val intent = Intent(this,OtherActivity::class.java)
startActivity(intent)
```

Cas 2. Envoi de données de type primitif.

```
val intent = Intent(this,OtherActivity::class.java)
intent.putExtra("id", 5)
intent.putExtra("name", "Bill")
startActivity(intent)
```

Lancement d'une activité

Cas 3. Envoi de données de type objet (La classe de l'objet doit implémenter l'interface *Serializable*).

Exemple

data class Person(val id:Int,val name:String):Serializable

Au niveau de l'activité

val intent = Intent(this,OtherActivity::class.java)

intent.putExtra("pr",Person(5,"Bill Gates"))

startActivity(intent)

Récupération des données envoyées

- **Lancement de Intent**

1. *val intent = Intent(this,Activity2::class.java)*
2. *intent.putExtra("id",5)*
3. *intent.putExtra("pr",Person(5,"Bill Gates"))*
4. *startActivity(intent)*

- **Récupération de Intent dans Activity2**

1. *val id= intent.getIntExtra("id",0) //valeur par défaut 0*
2. *val person = intent.getSerializableExtra("pr") as Person*

Plan

- Utilisation des Intent
  - Lancement des activités
  - Lancement des applications externes
  - Récupération du résultat d'un Intent
- Cycle de vie d'une activité
- Gestion du changement de configuration

Lancement d'une application externe

Un Intent permet de lancer une application installée sur le dispositif mobile.

Ouvrir l'interface d'appel

Lancer la caméra

Ouvrir l'interface d'envoi de SMS

Afficher un lieu sur Google Maps

Créer une alarme

Ouvrir une page web

Lancement d'une application externe

1. Ouvrir l'interface d'appel

Exemple

```
val uri = Uri.parse("tel:021212121")
val intent = Intent(Intent.ACTION_DIAL, uri)
if (intent.resolveActivity(packageManager) != null) {
startActivity(intent)
}
```

Lancement d'une application externe

2. Afficher un lieu sur Google Maps

Exemple

```
val latitude = ....
val longitude = ....
val geoLocation = Uri.parse("geo:$latitude,$longitude")
val intent = Intent(Intent.ACTION_VIEW,geoLocation)
if (intent.resolveActivity(packageManager) != null) {
startActivity(intent)
}
```

Lancement d'une application externe

3. Ouvrir une page web

Exemple

```
val url = “https://www.google.com”
val intent = Intent(Intent.ACTION_VIEW, Uri.parse(url))
startActivity(intent)
```

Plan

- Utilisation des Intent
  - Lancement des activités
  - Lancement des applications externes
  - Récupération du résultat d'un Intent
- Cycle de vie d'une activité
- Gestion du changement de configuration

Récupération du résultat d'un Intent

Problème. Comment récupérer le résultat d'un Intent au niveau de l'activité ?

Exemple

- Lancer la caméra, prendre une photo et l'afficher sur l'interface de l'activité.
- Le résultat dans ce cas est la photo prise par la caméra.

Récupération du résultat d'un Intent

Solution. Utilisation de *startActivityForResult()* et *onActivityResult()*.

- *startActivityForResult()* : utilisée pour lancer Intent
- *onActivityResult()* : utilisée pour ajouter le traitement après la réception du résultat.

Récupération du résultat d'un Intent

Exemple. Prendre une photo et l'afficher sur l'interface

1. Lancement de la caméra

```
val REQUEST_IMAGE = 100 // utilisé pour récupérer le résultat

fun openCamera() {
val intent =Intent( MediaStore.ACTION_IMAGE_CAPTURE)
if(intent.resolveActivity(getPackageManager()) !=null) {
startActivityForResult(intent,REQUEST_IMAGE);
}
}
}
```

Récupération du résultat d'un Intent

Exemple. Prendre une photo et l'afficher sur l'interface

2. Récupération et affichage du résultat

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data:
      Intent?) {
  super.onActivityResult(requestCode, resultCode, data)
if (requestCode == REQUEST_IMAGE && resultCode ==
      RESULT_OK) {
 if (data != null && data.extras != null) {
// Affichage dans ImageView
val imageBitmap = data?.extras?.get("data") as Bitmap
mImageView.setImageBitmap(imageBitmap)
          } }
}
```

Récupération du résultat d'un Intent

Exemple. Prendre une photo et l'afficher sur l'interface

3. Ajouter la persmission caméra dans le fichier *manifest*

```
<uses-permission
    android:name="android.permission.CAMERA" />
```

Plan

- Utilisation des Intent
  - Lancement des activités
  - Lancement des applications externes
  - Récupération du résultat d'un Intent
- Cycle de vie d'une activité
- Gestion du changement de configuration

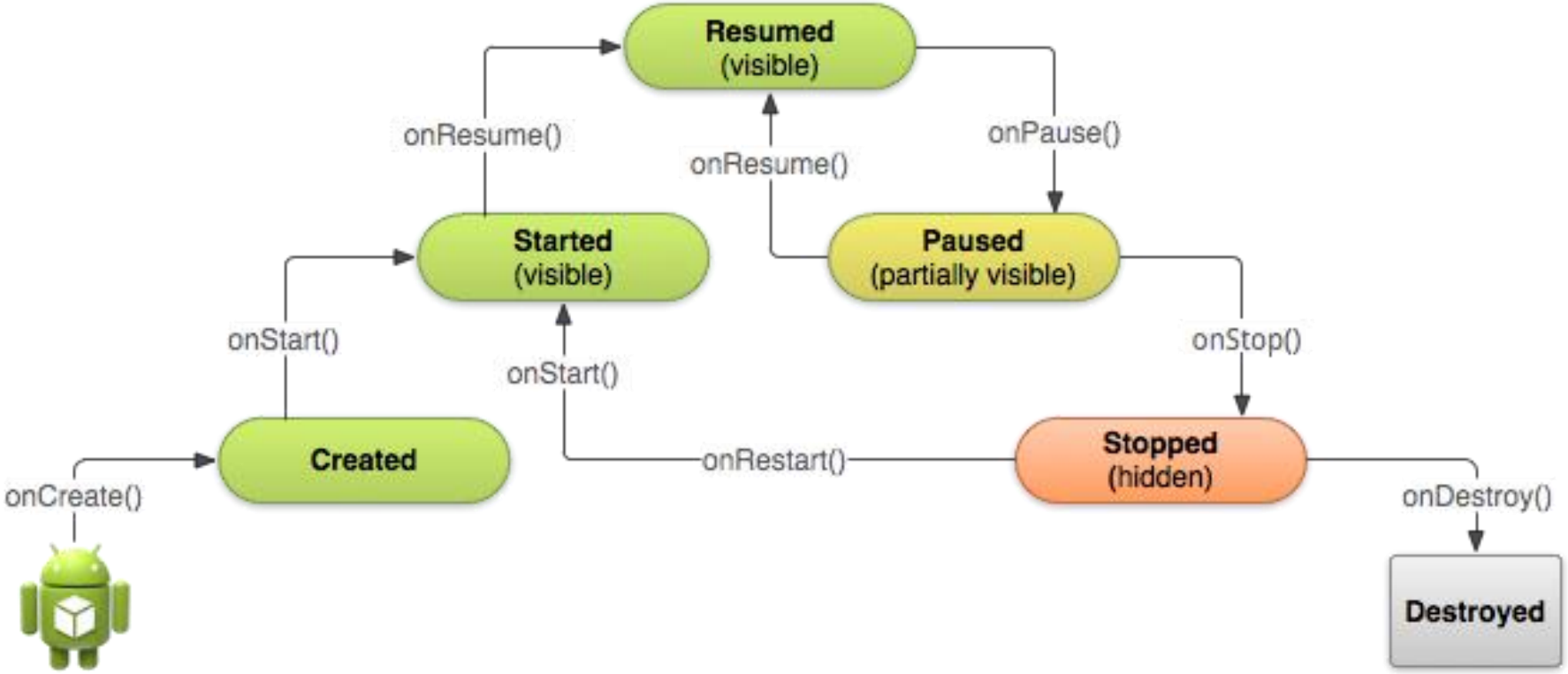
Cycle de vie d'une activité

- Le cycle de vie d'une activité représente les différents états de l'activité: de sa création à sa destruction.
- La compréhension du cycle de vie facilite au développeur la gestion des données en mémoire.

Exemple.

La perte des données affichées sur l'activité après la rotation de l'écran.

Cycle de vie d'une activité



Cycle de vie d'une activité

Les états d'une activité

1. *Created*.

L'activité est créée mais pas visible à l'utilisateur.

2. *Started*

L'activité est visible mais l'utilisateur ne peut pas interagir avec l'interface. Cet état est utilisé par le système *Android* mais rarement utilisé en développement.

3. *Resumed*.

L'activité est visible et l'utilisateur peut interagir avec l'interface.

Cycle de vie d'une activité

Les états d'une activité

4. *Paused.*

L'activité est visible partiellement à l'utilisateur et les données sont toujours en mémoire.

5. *Stopped.*

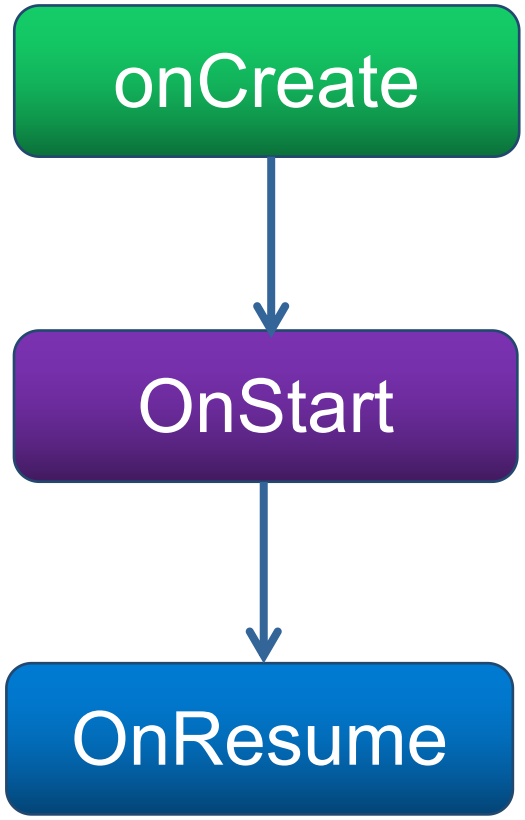
L'activité n'est pas visible à l'utilisateur mais les données sont toujours en mémoire.

6. *Destroyed.*

L'activité est supprimée de la mémoire et les données sont perdues.

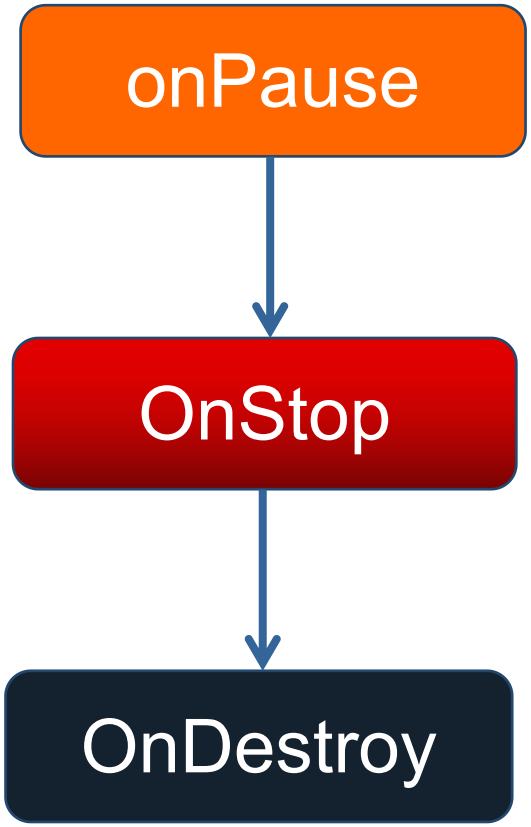
Cycle de vie d'une activité

Cas 1. L'utilisateur ouvre une activité.



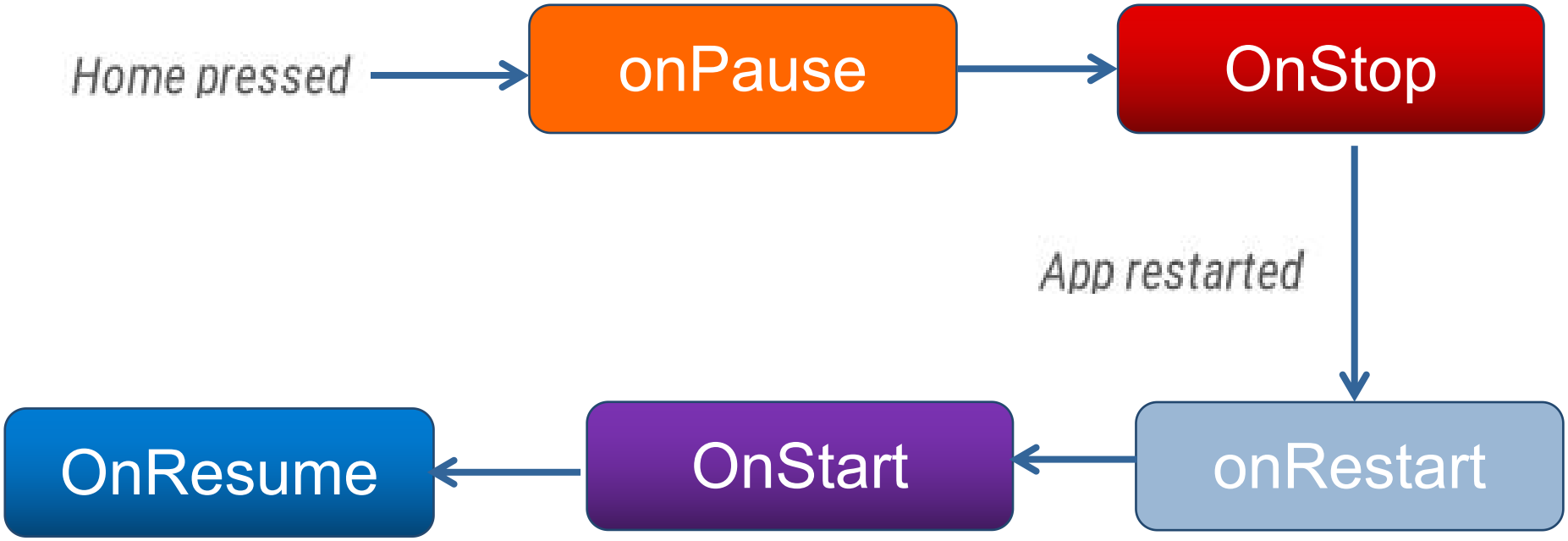
Cycle de vie d'une activité

Cas 2. L'utilisateur quitte l'activité avec le bouton retour.



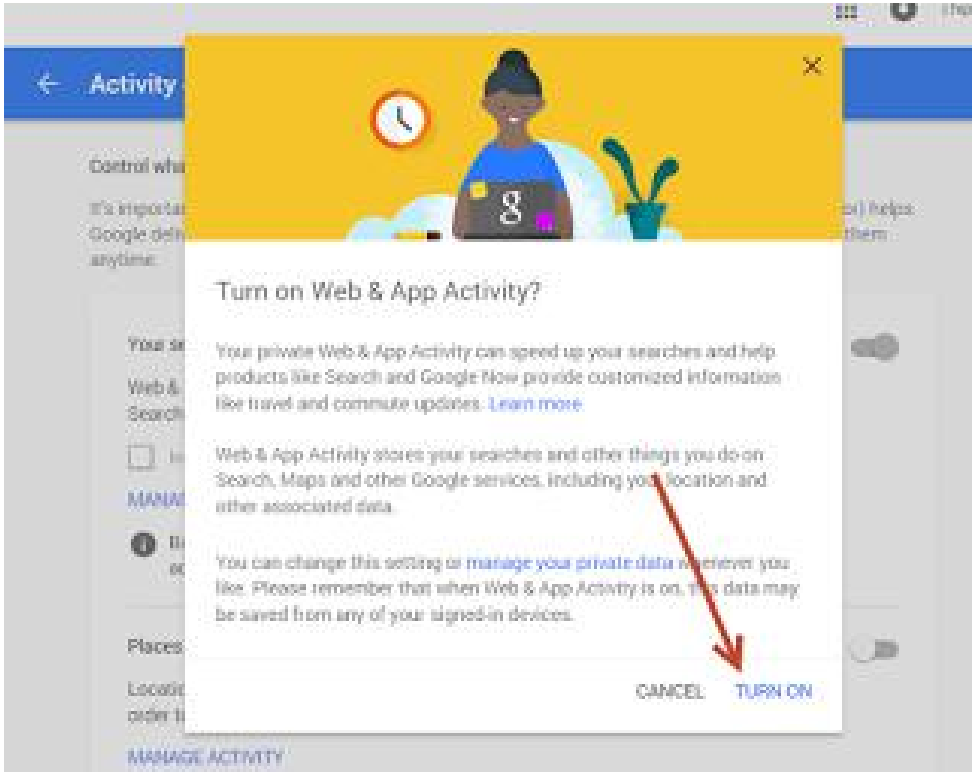
Cycle de vie d'une activité

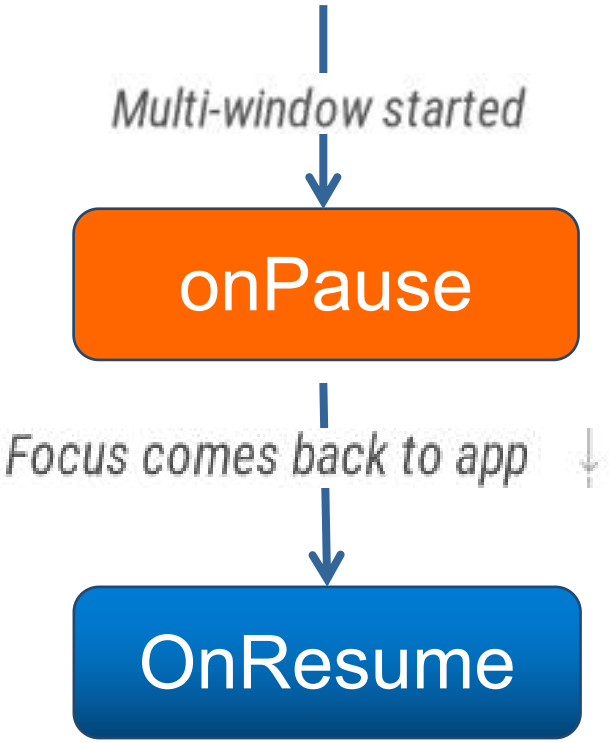
Cas 3. L'utilisateur quitte l'activité avec le bouton *Home,* répond à un appel ou ouvre une autre activité, etc. Puis retourne à l'activité.



Cycle de vie d'une activité

Cas 4. L'activité est ouverte, une boite de dialogue s'affiche.





Cycle de vie d'une activité: les méthodes utilisées

1. ***OnCreate***

Charger et intialiser l'interface utilisateur.

2. ***OnPause***

Libérer les ressources: GPS, caméra,etc.

3. ***OnResume***

Récupérer les ressources libérées dans *onPause*.

4. ***finish***

Détruire une activité.

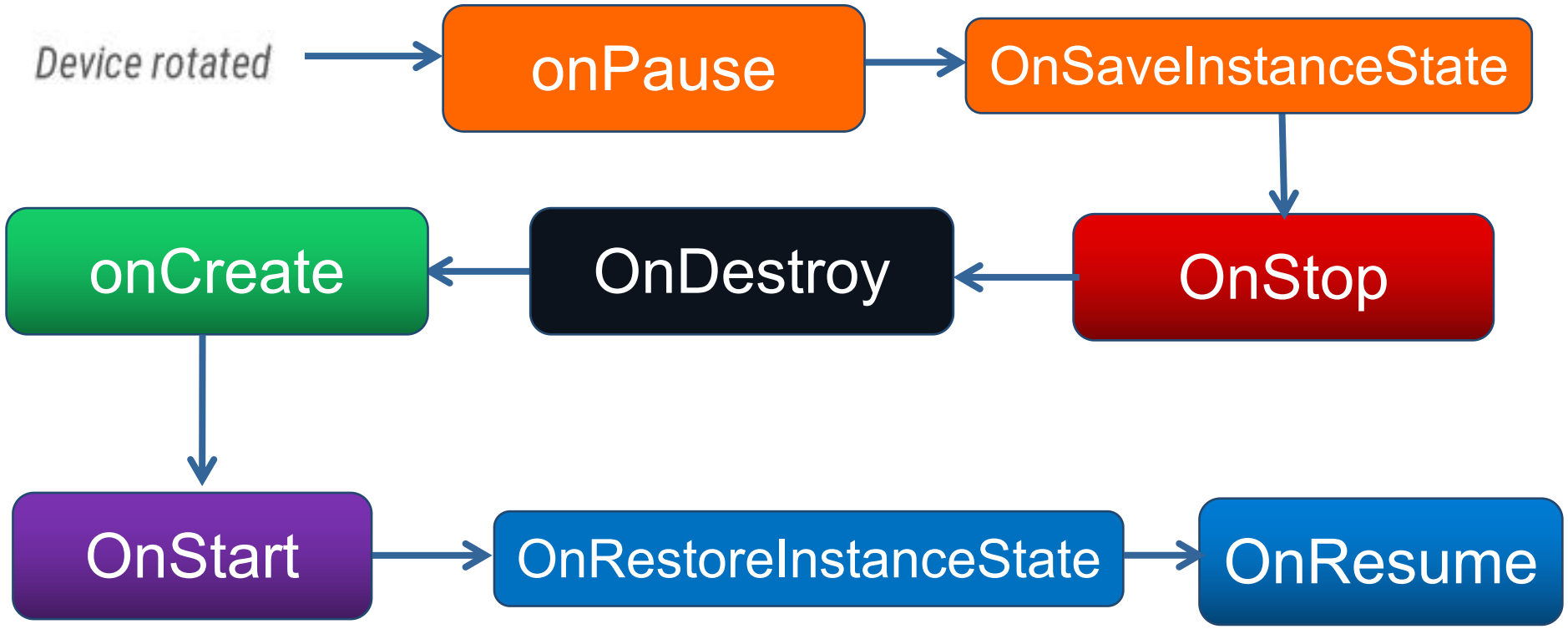
Plan

- Utilisation des Intent
  - Lancement des activités
  - Lancement des applications externes
  - Récupération du résultat d'un Intent
- Cycle de vie d'une activité
- Gestion du changement de configuration

Gestion du changement de configuration

Changement de configuration (Rotation de l'écran).

Activité détruite puis recréer par le système



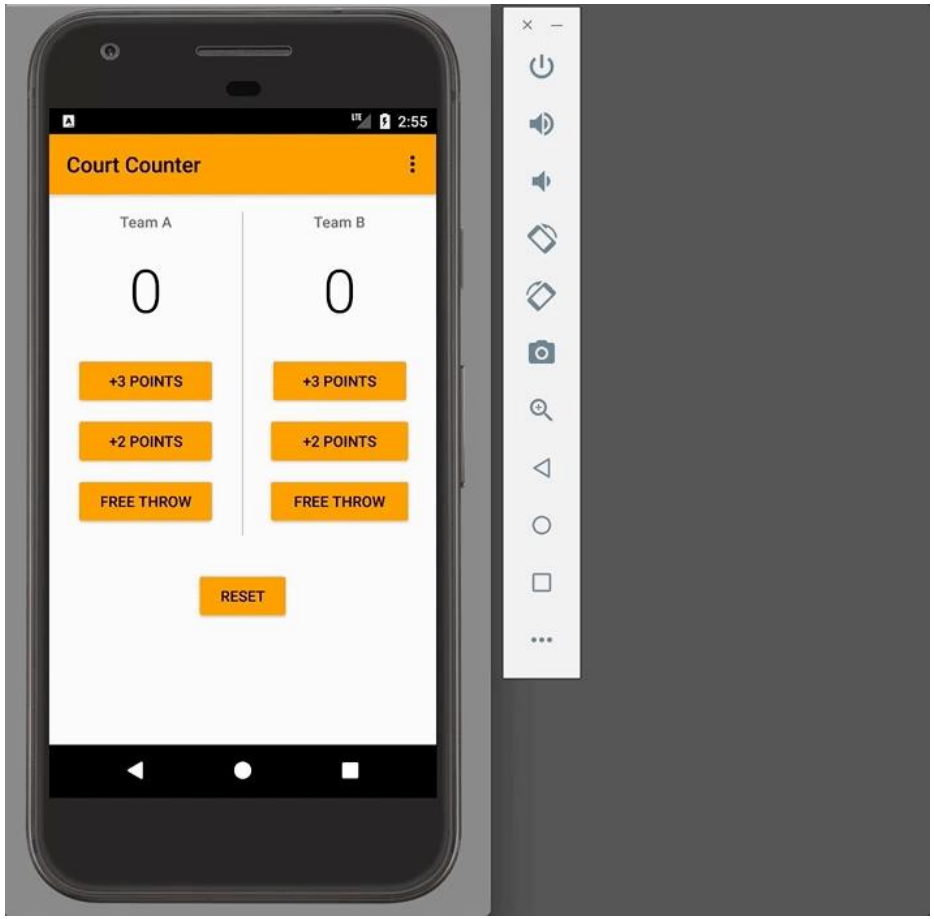
Gestion du changement de configuration

Exemple

- Un jeu de plusieurs niveaux.
- Le score est affiché à l'utilisateur.
- Sauvegarde du score dans une base de données à la fin de chaque niveau.

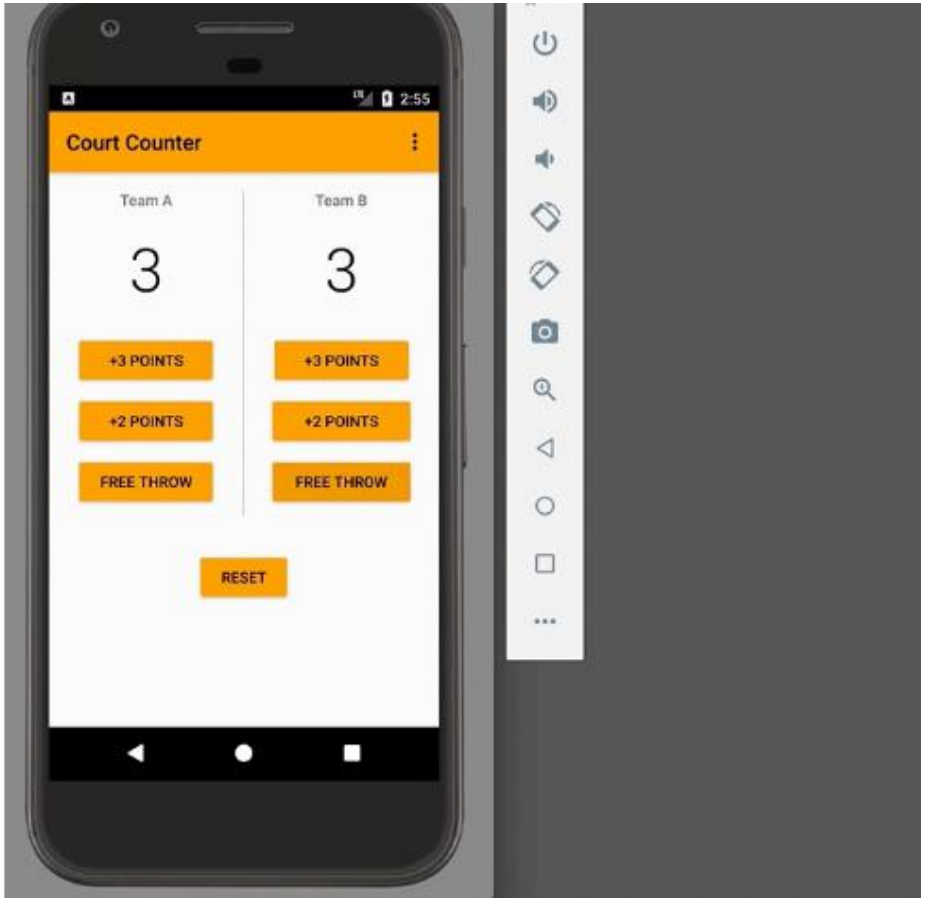
Gestion du changement de configuration

Problème. Le score est réinitialisé après la rotation de l'écran.



Gestion du changement de configuration

Problème. Le score est réinitialisé après la rotation de l'écran.



Gestion du changement de configuration

Problème. Le score est réinitialisé après la rotation de l'écran.

Gestion du changement de configuration

Solutions. Deux solutions existent pour gérer le changement de configuration.

onSaveInstanceState() &
onRestoreInstanceState()

ViewModel

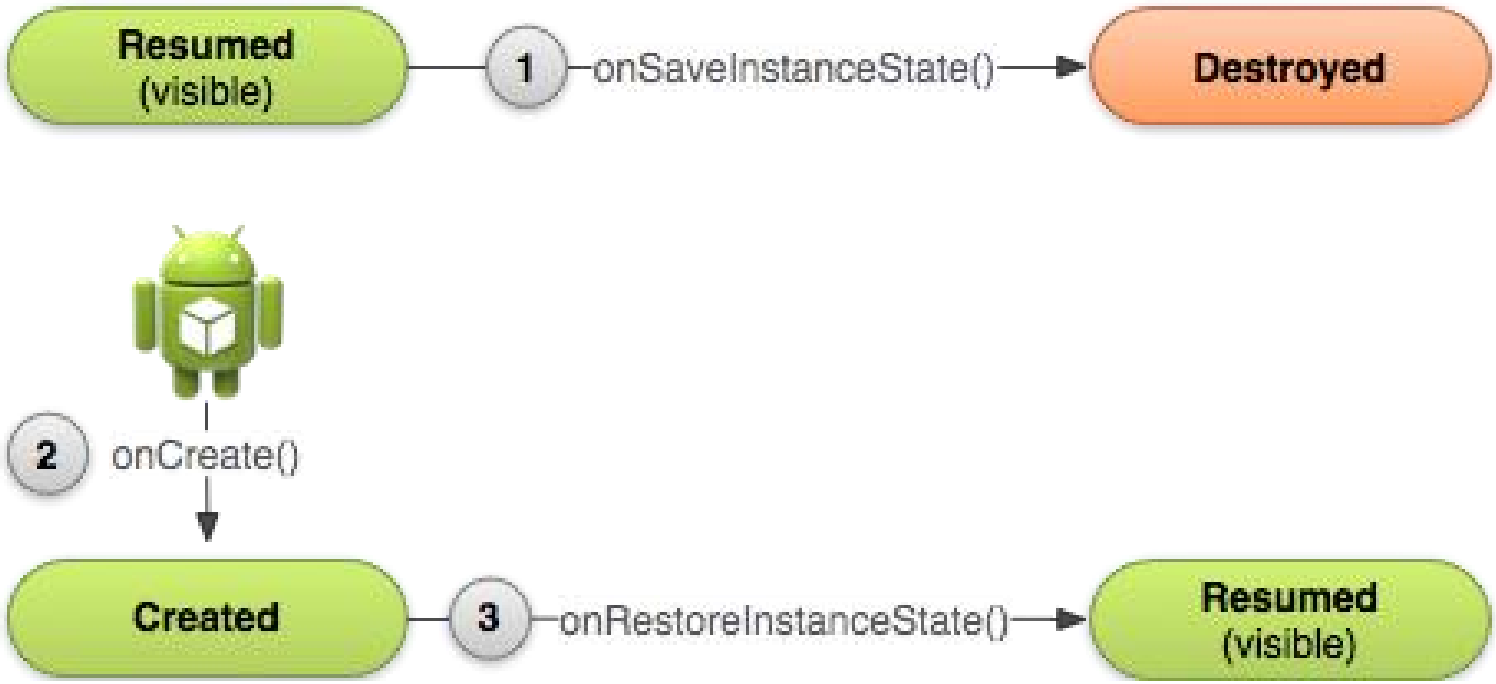
Gestion du changement de configuration

onSaveInstanceState() &
onRestoreInstanceState()

ViewModel

Gestion du changement de configuration

Solution 1. Utilisation de *onSaveInstanceState()* & *onRestoreInstanceState()*.



Gestion du changement de configuration

La méthode ***onSaveInstanceState()***

- Appelée avant de détruire l'activité sans la demande de l'utilisateur (espace mémoire insuffisant, etc.).
- Utilisée pour sauvegarder l'état de l'activité.
- Utilisée par le système pour sauvegarder les états des vues (valeur d'un EditText, la position du scroll, etc.).

Gestion du changement de configuration

La méthode ***onRestoreInstanceState()***

Appelée après la re-création de l'activité.

Utilisée pour restaurer les états sauvegardés dans *onSaveInstanceState()*.

Utilisée par le système pour restaurer l'état des vues après la recréation de l'activité.

Gestion du changement de configuration

Exemple. Sauvegarde et restauration d'une valeur variable d'un TextView après la rotation de l'écran

1. **Sauvegarde de la valeur du TextView**

```
override fun onSaveInstanceState(outState: Bundle) {
    super.onSaveInstanceState(outState)
    val i = textView.text.toString().toInt()
    outState.putInt("value",i)
}
```

Gestion du changement de configuration

Exemple. Sauvegarde et restauration d'une valeur variable d'un TextView après la rotation de l'écran

2. Restauration de la valeur du TextView

```
override fun onRestoreInstanceState(savedInstanceState: Bundle?) {
    super.onRestoreInstanceState(savedInstanceState)
    val i =  savedInstanceState?.getInt("i")
    textView.text = i.toString()
}
```

Gestion du changement de configuration

onSaveInstanceState() &
onRestoreInstanceState()

ViewModel

OnSaveInstanceState()
Les limites

Problème

La sauvegarde d'une liste d'objets est plus complexe.

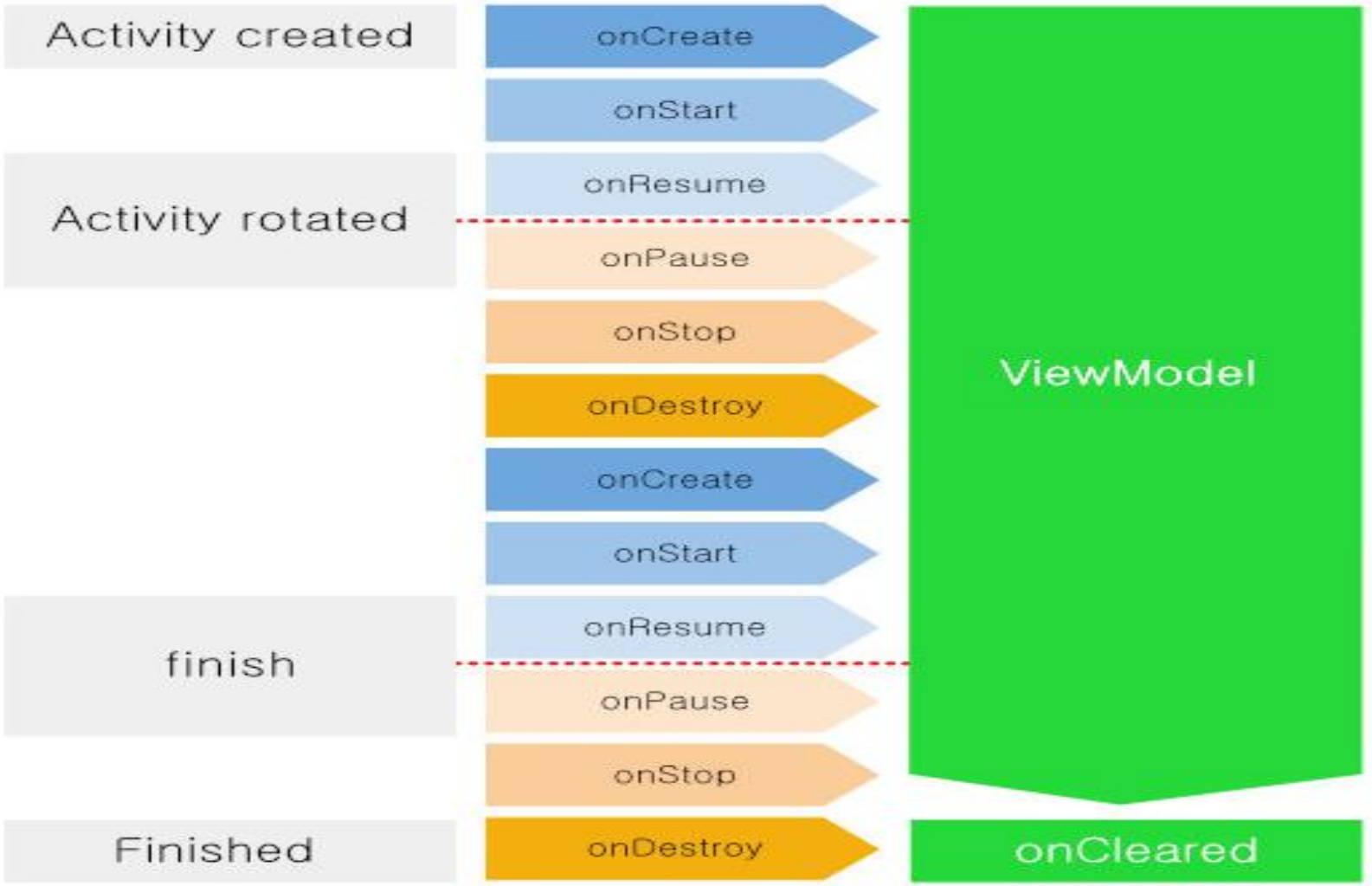
Solution 2

Sauvegarder les éléments de la liste dans un objet ayant un cycle de vie indépendant du cycle de vie de l'activité: *ViewModel*.

Android ViewModel

- Une classe permettant de faciliter la sauvegarde des données après la changement de configuration.
- Une instance de *ViewModel* est toujours liée à une activité.
- Le cycle de vie de *ViewModel* est indépendant du cycle de vie de l'activité.

Cycle de vie de ViewModel



Implémentation de ViewModel

1. Installation

implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'

2. Création d'une sous-classe de *ViewModel* et définition des attributs à sauvagarder

```
class MyModel:ViewModel() {
val list by lazy {
    getDoctors()
}
```

Implémentation de ViewModel

3. Création de l'objet dans l'activité

val vm= ViewModelProvider(this).get(MyModel::class.java)

4. Utilisation des attributs du *ViewModel* dans l'activité

val adapter = Custom Adapter(this,vm.list)

listDoctors.adapter = adapter

Références

1. https://developer.android.com/guide/components/intents-common
2. http://developer.android.com/training/basics/activity-lifecycle/index.html
3. https://medium.com/google-developers/the-android-lifecycle-cheat-sheet-part-i-single-activities-e49fd3d202ab
4. https://developer.android.com/guide/components/activities/state-changes.html