

Les activités



Sources : <http://www.android.com>, wikipedia...

le cnam

Paris, 28/11/2017

Les Activités

2

- Composant applicatif central de l’interface utilisateur d’une application. Il gère le cycle de vie de l’IHM de l’application et son intégration au système Android.
- C’est le seul des composants applicatifs qui gèrent l’interface graphique utilisateur.
- Toutes les activités définies par le programmeur doivent être déclarées dans le fichier `AndroidManifest.xml` de l’application (balise **activity**). C’est le système qui crée, gère et libère les composants applicatifs.
- Chaque nouvel écran présenté à l’utilisateur est porté par une activité différente (cas particuliers : onglets et fragments)

Les Activités

3

- Une activité occupe la totalité de l’écran (philosophie une seule tâche à la fois).
- Le modèle de comportement est celui d’une page Web :
 - Le système gère la navigation comme un navigateur Web (lancement, bouton «home», touche «back»...)
 - Les actions des widgets de l’interface (boutons, cases à cocher...) se comportent comme des hyperliens et peuvent lancer de nouvelles activités (empilement)
 - L’activité dialogue avec des tâches gérant l’état ou les données de l’application (modèle client-serveur du Web).
 - Si une activité a un état propre sauvegardé en mémoire celui-ci est minimal (*quasi - stateless*)

Les Activités

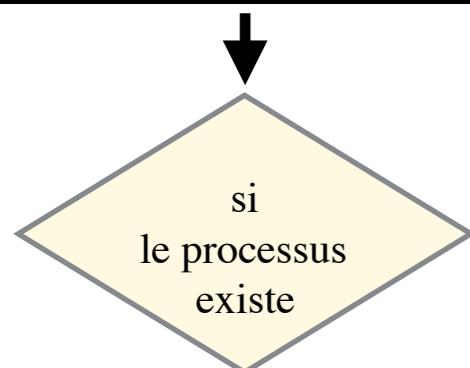
4

- Concrètement il s'agit d'une sous-classe de la classe :
android.app.Activity
- Instanciée par le runtime Android, **l'utilisateur ne crée pas les instances d'une Activité !**
- Le cycle de vie de l'activité est géré par le «runtime» du système Android à travers le fil d'exécution principal de l'application : le **UIThread**
- Comportement général de l'activité est obtenu par redéfinition d'un certain nombre de méthodes (dont le nom commence généralement par `on...`). Dans chacune de ces méthodes, il faudra appeler le comportement de la méthode parente pour «rassurer le système».

Cycle de vie d'une activité

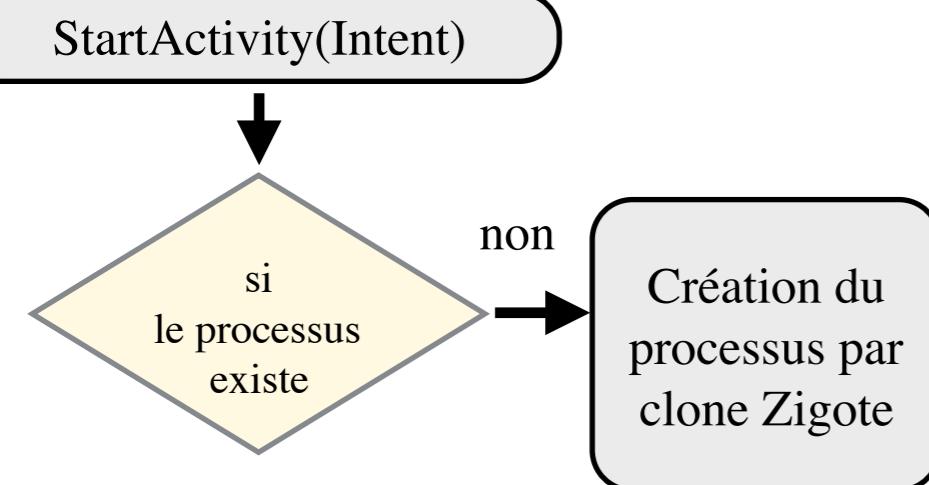
5

StartActivity(Intent)



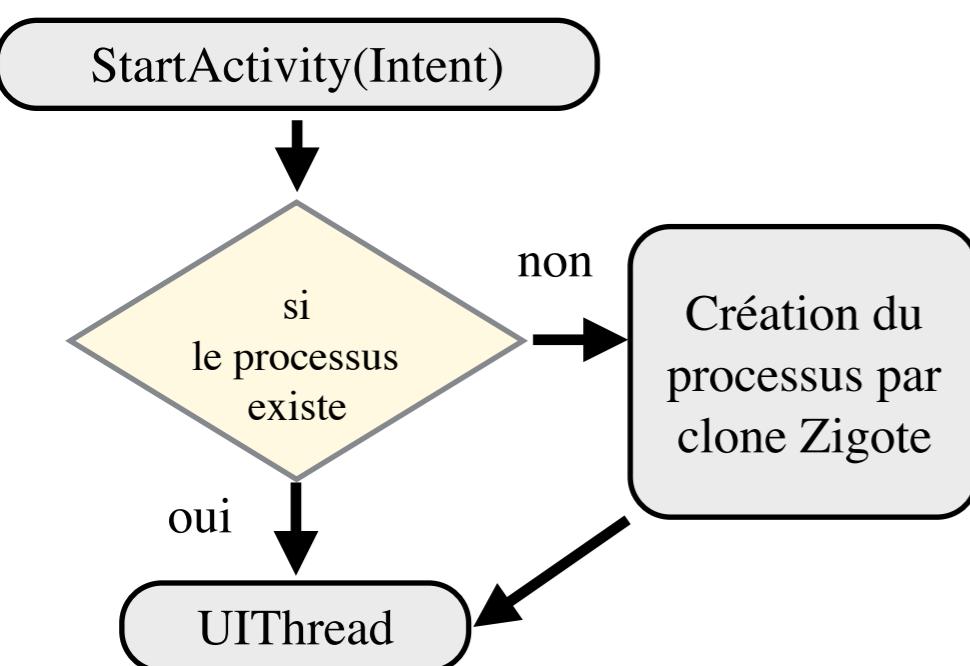
Cycle de vie d'une activité

5



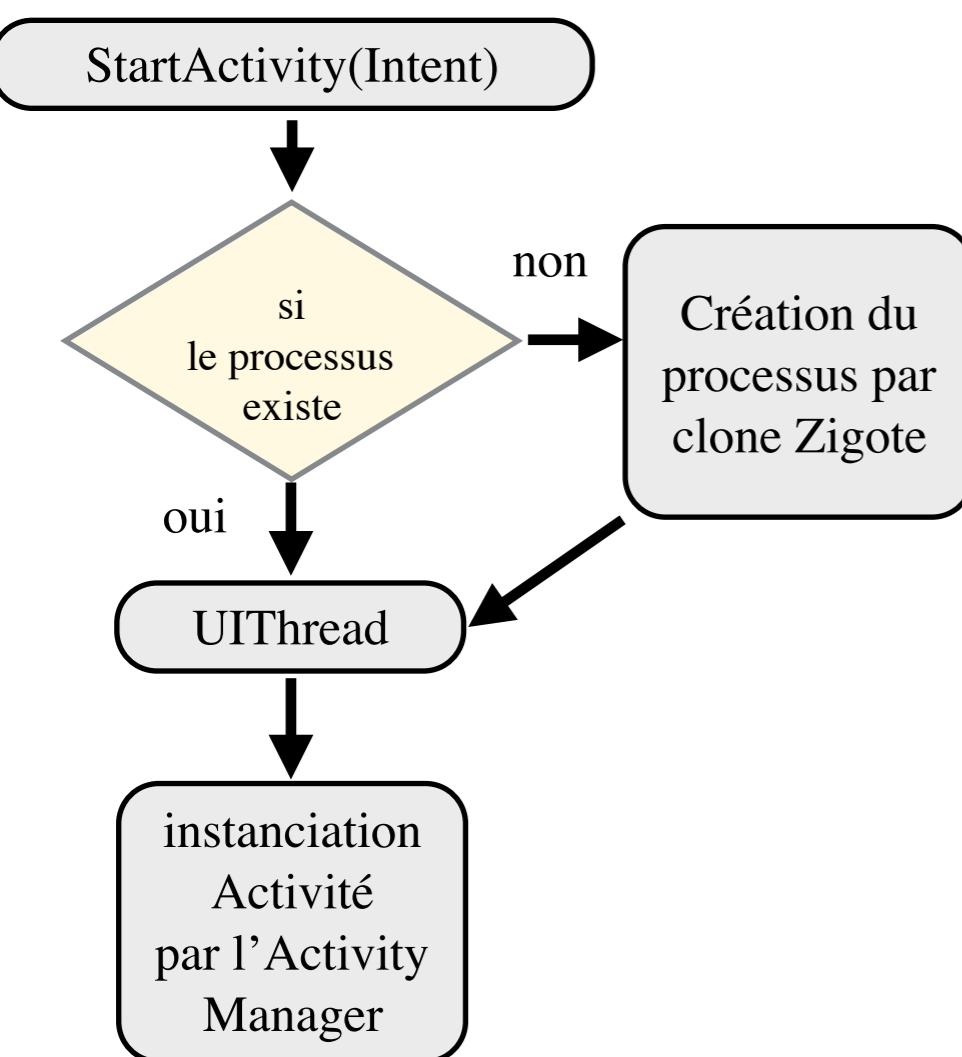
Cycle de vie d'une activité

5



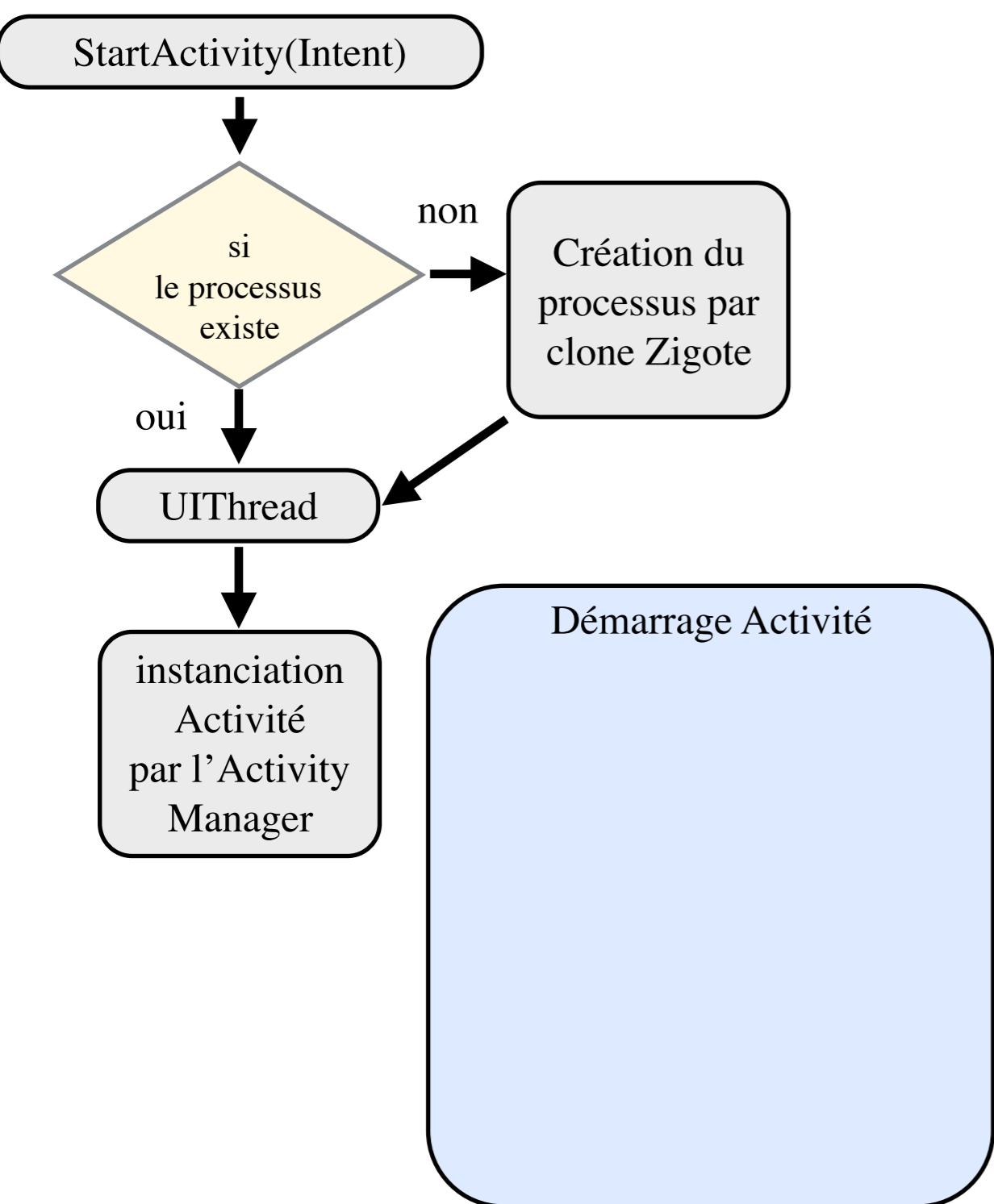
Cycle de vie d'une activité

5



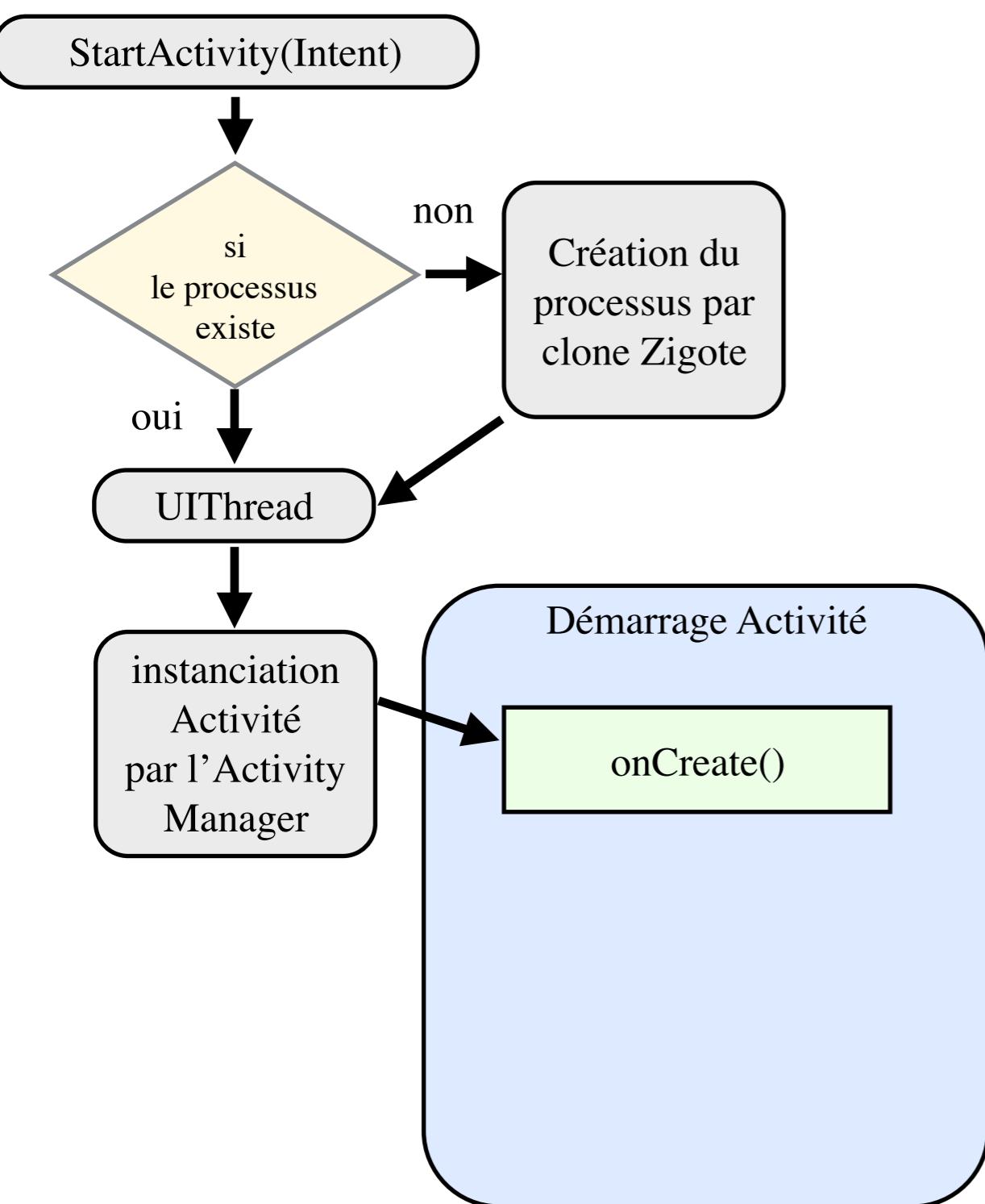
Cycle de vie d'une activité

5



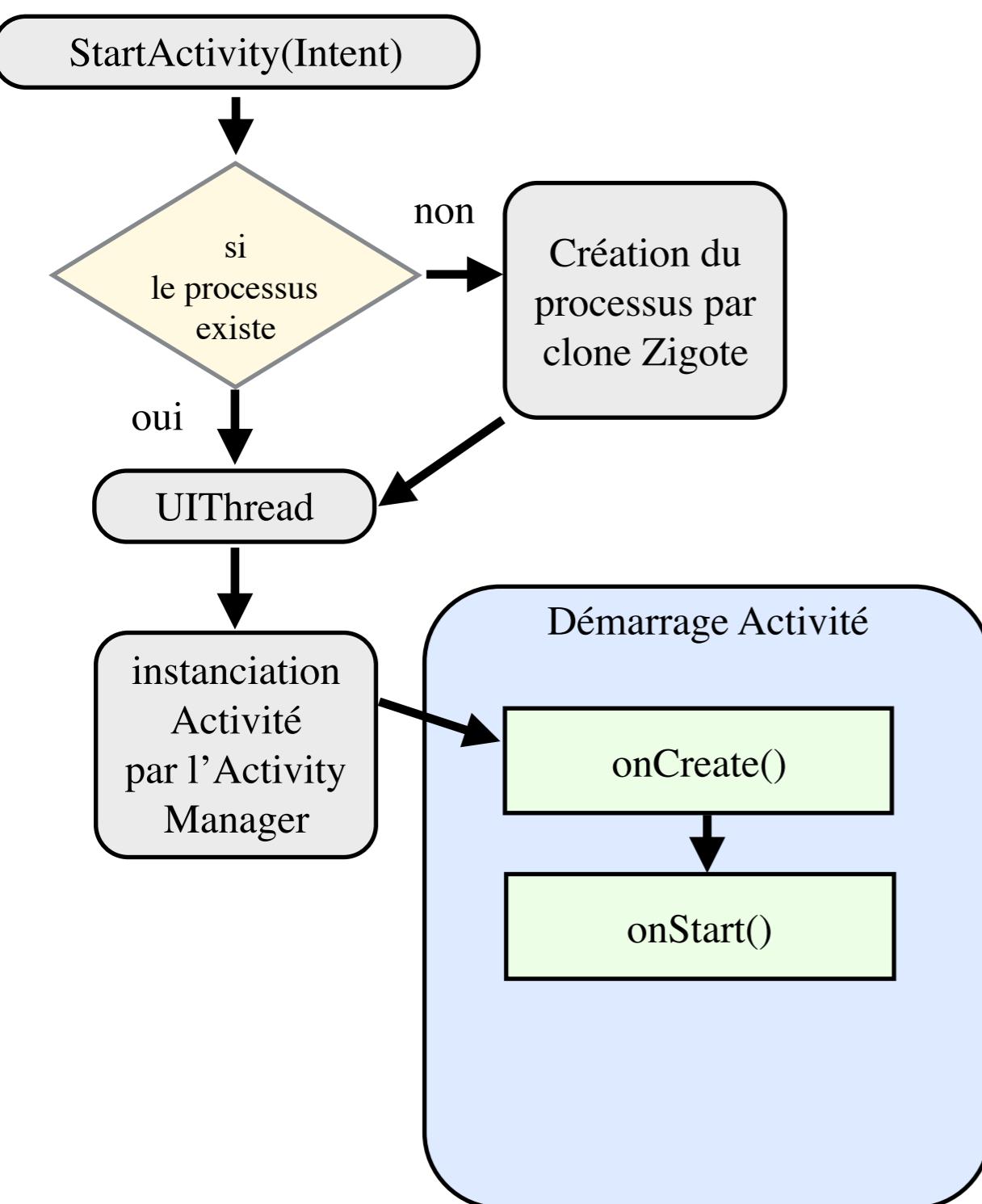
Cycle de vie d'une activité

5



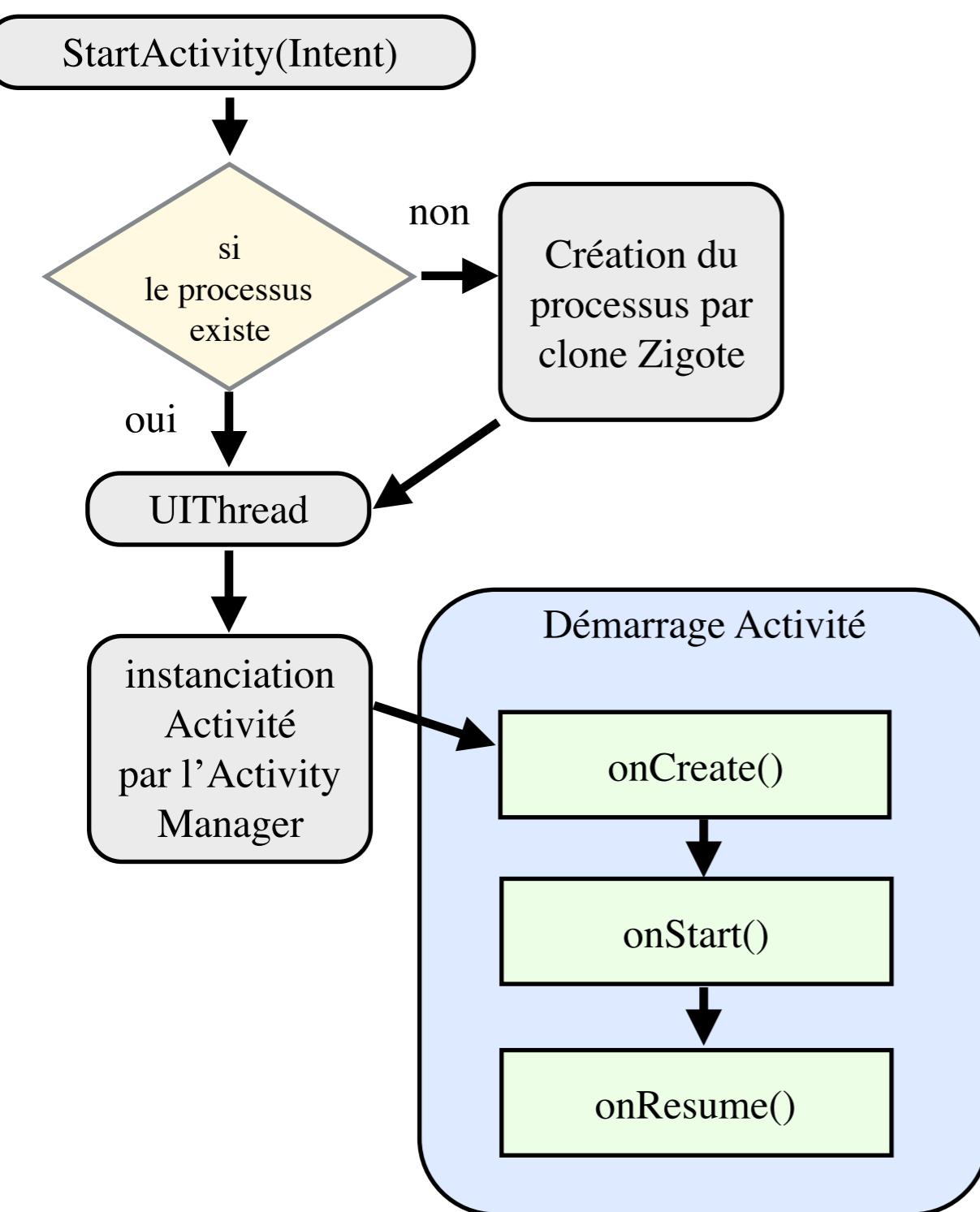
Cycle de vie d'une activité

5



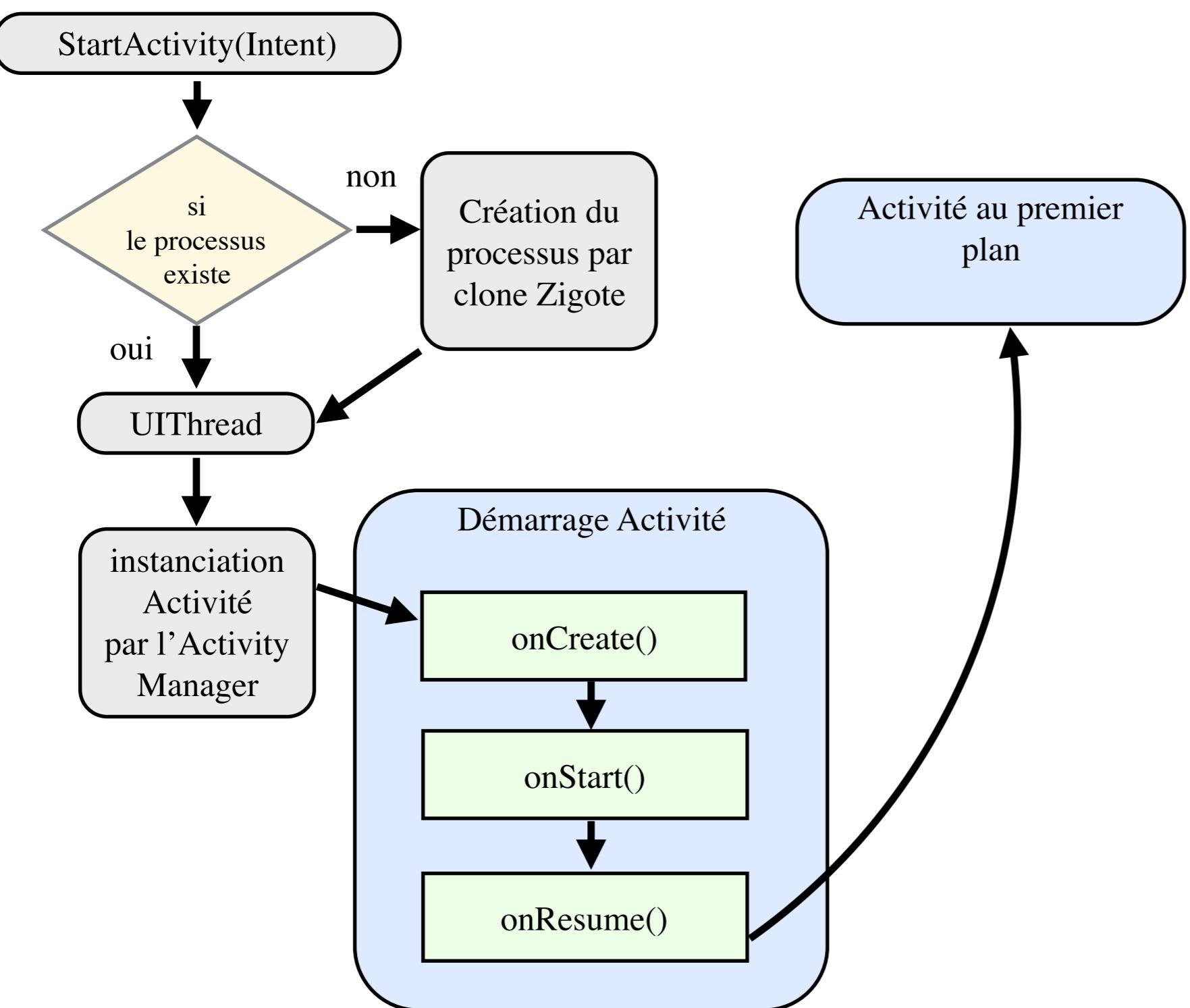
Cycle de vie d'une activité

5



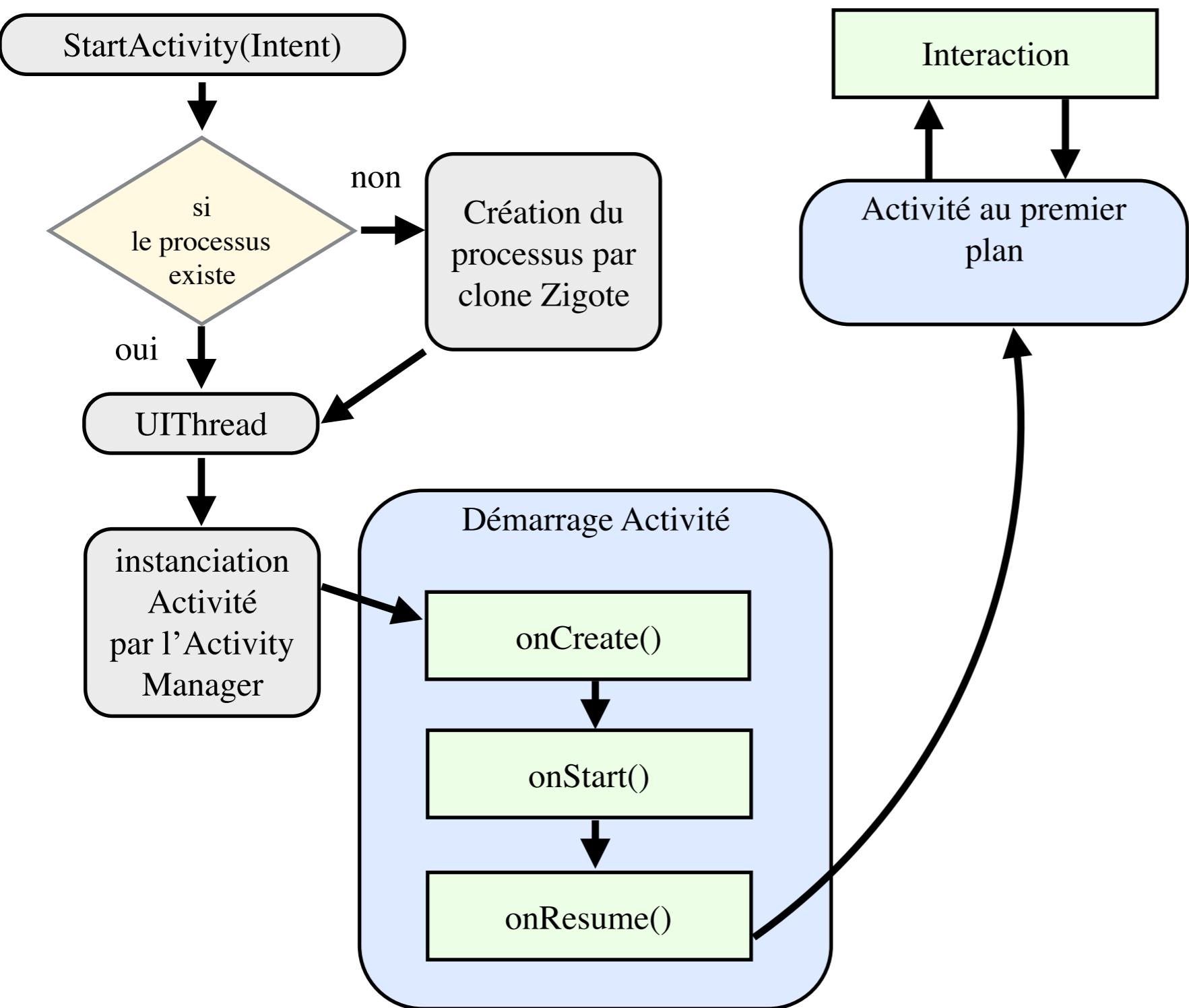
Cycle de vie d'une activité

5



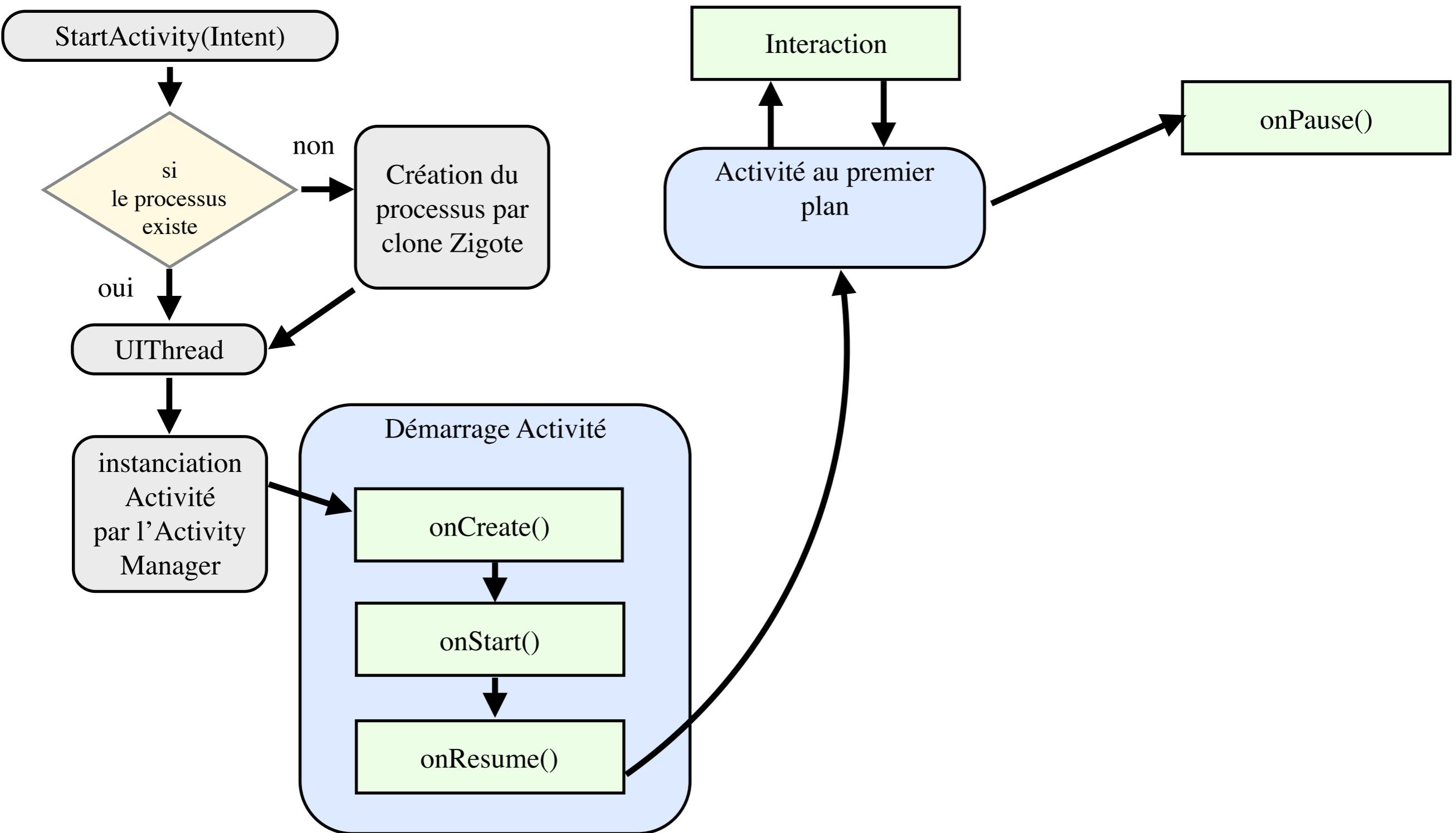
Cycle de vie d'une activité

5



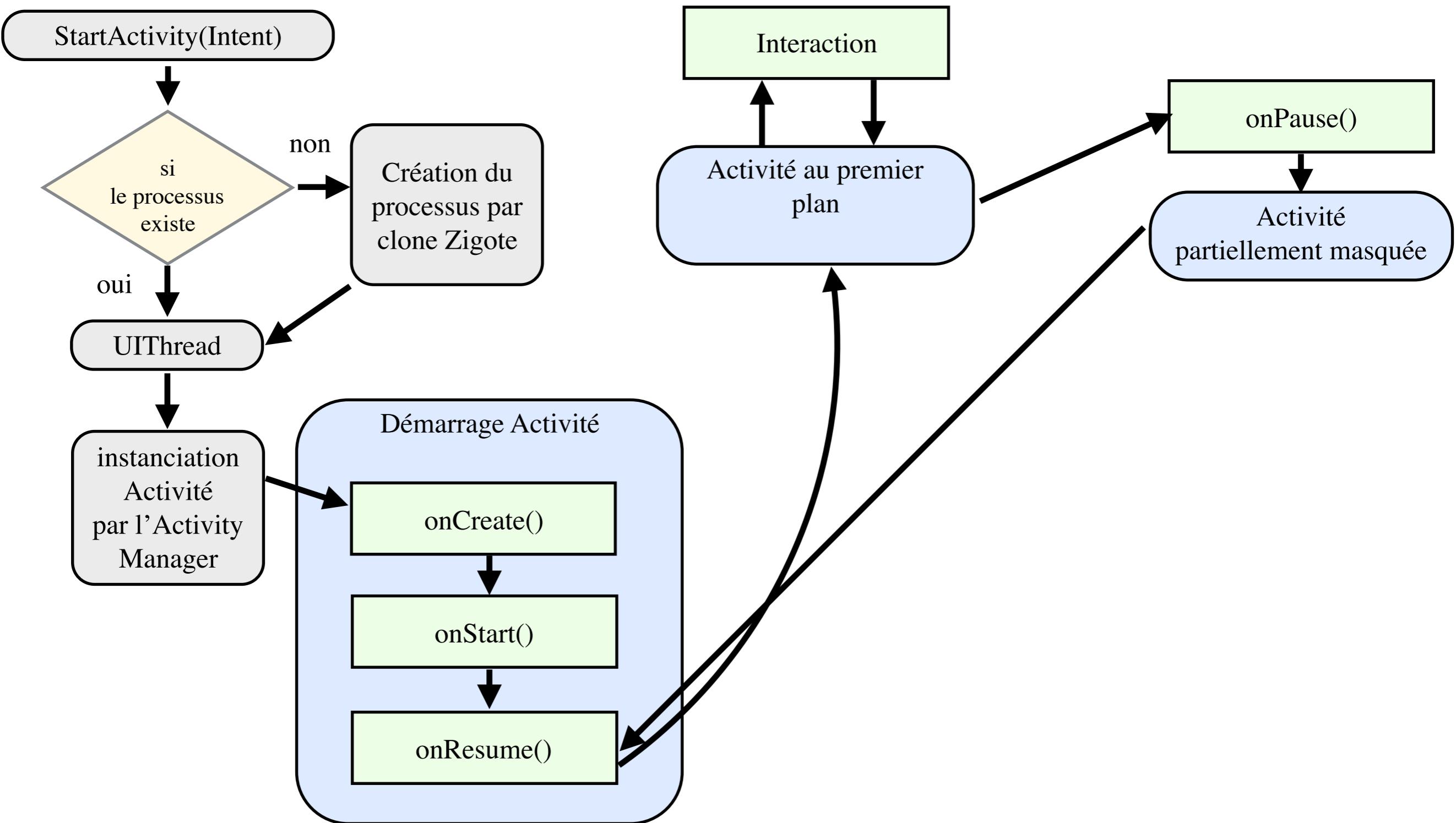
Cycle de vie d'une activité

5



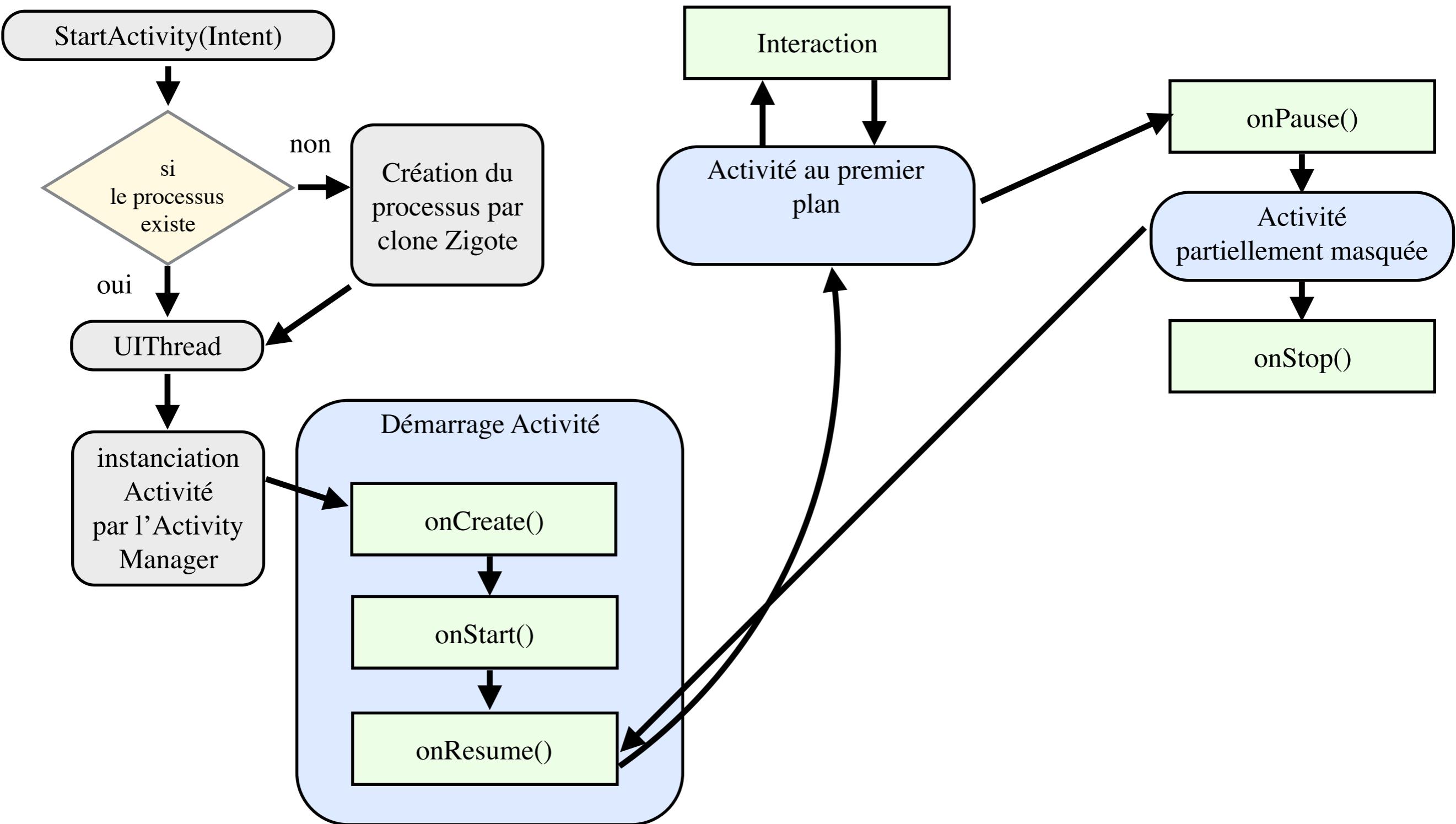
Cycle de vie d'une activité

5



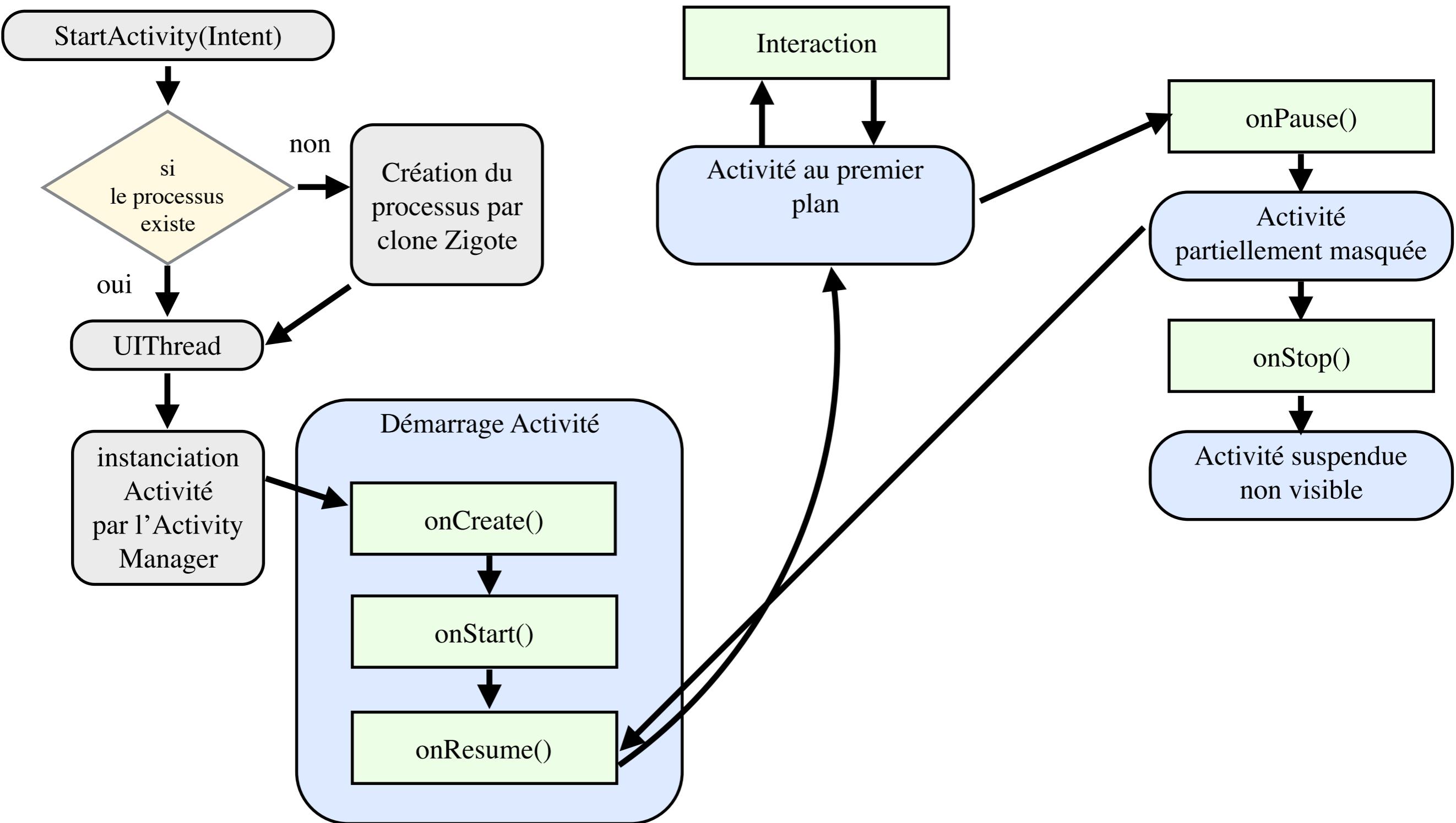
Cycle de vie d'une activité

5



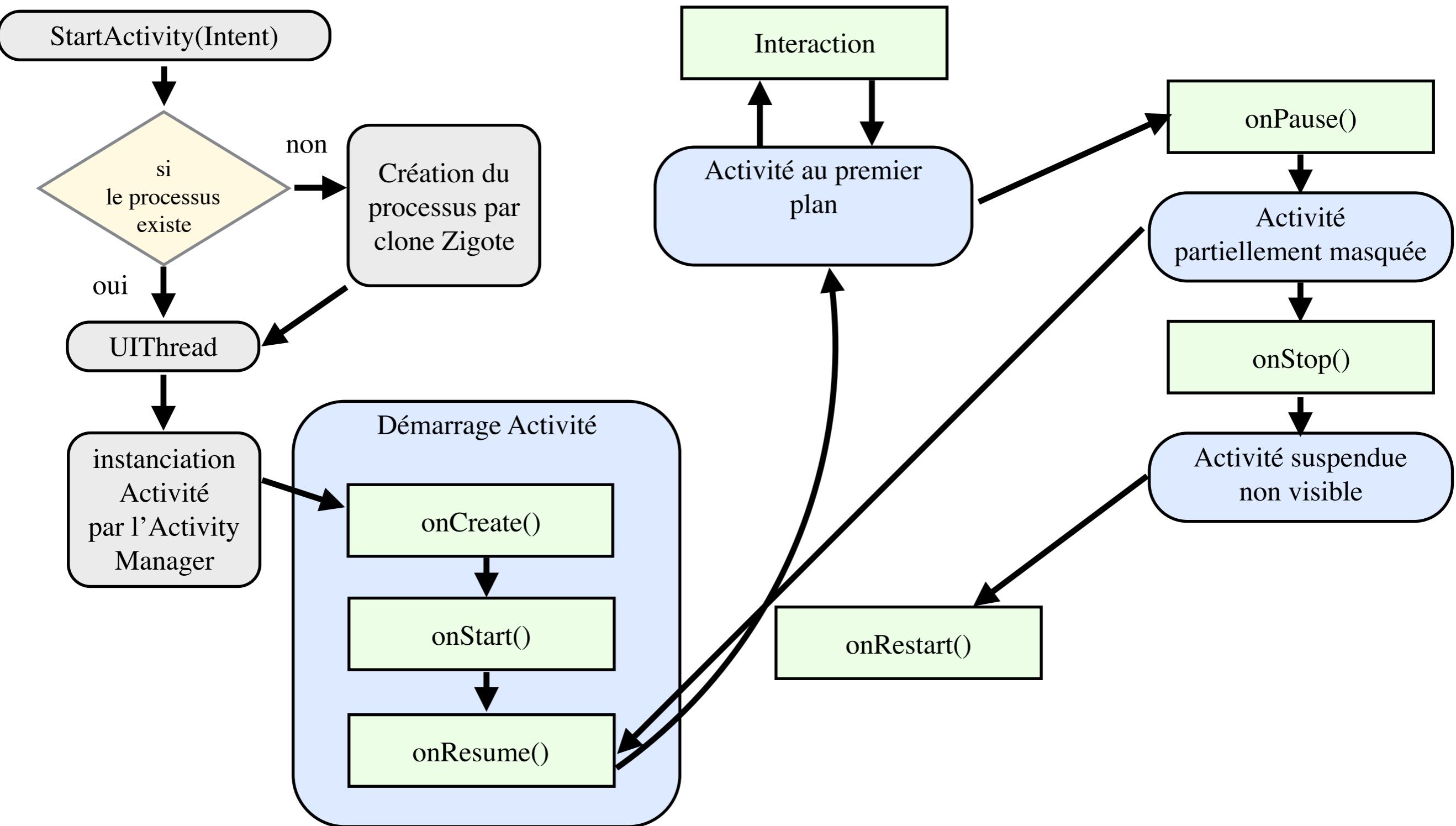
Cycle de vie d'une activité

5



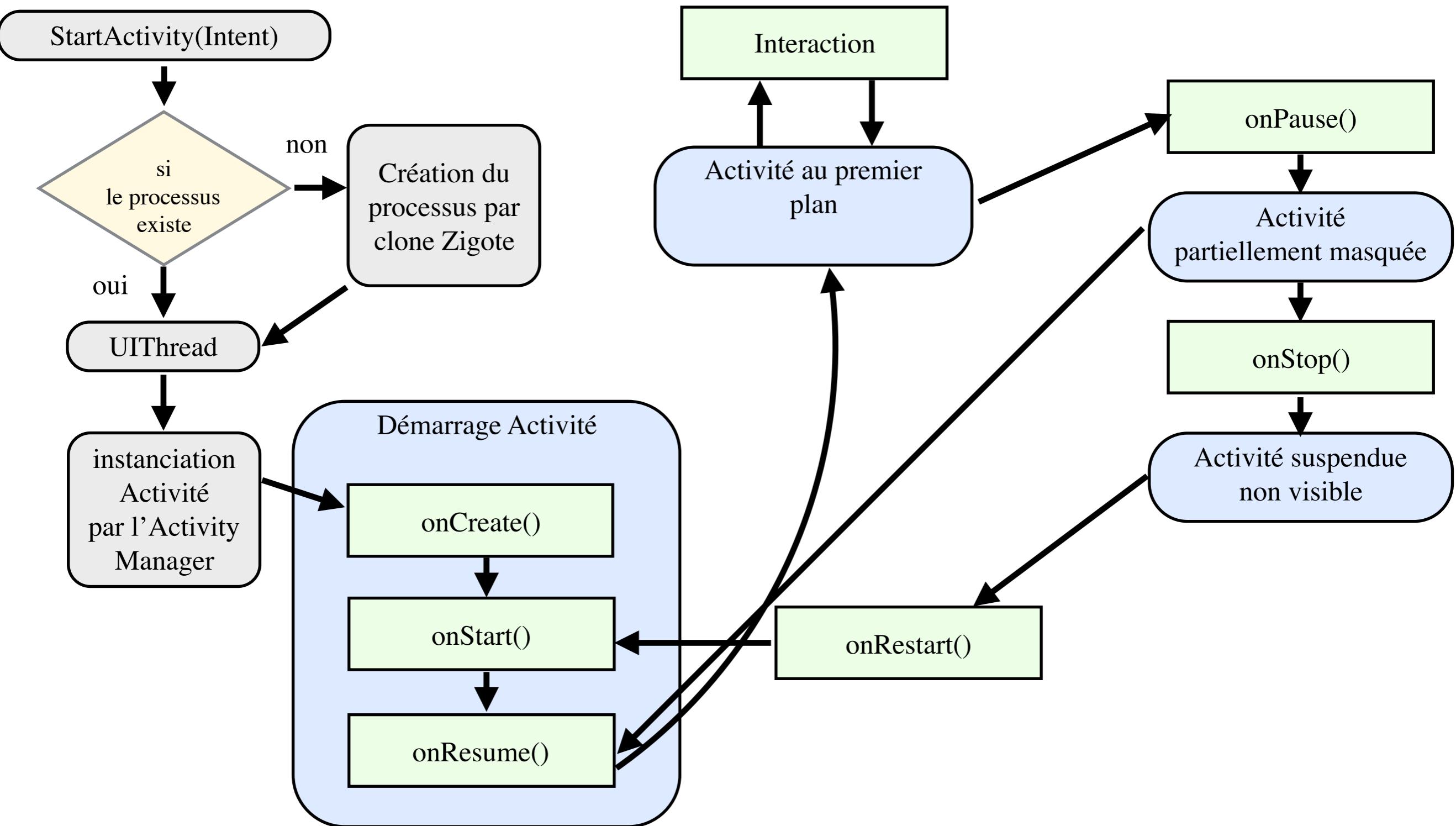
Cycle de vie d'une activité

5



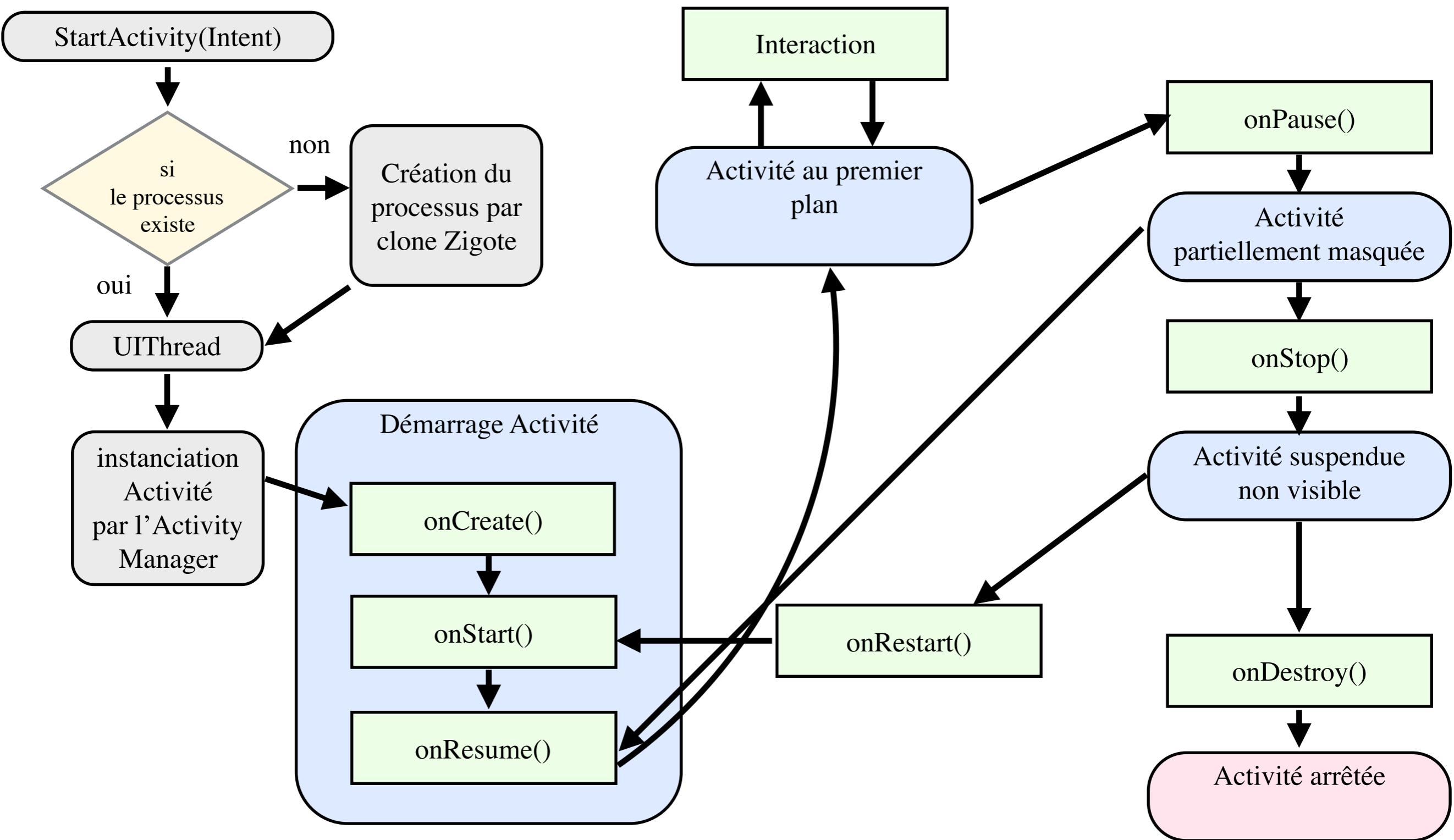
Cycle de vie d'une activité

5



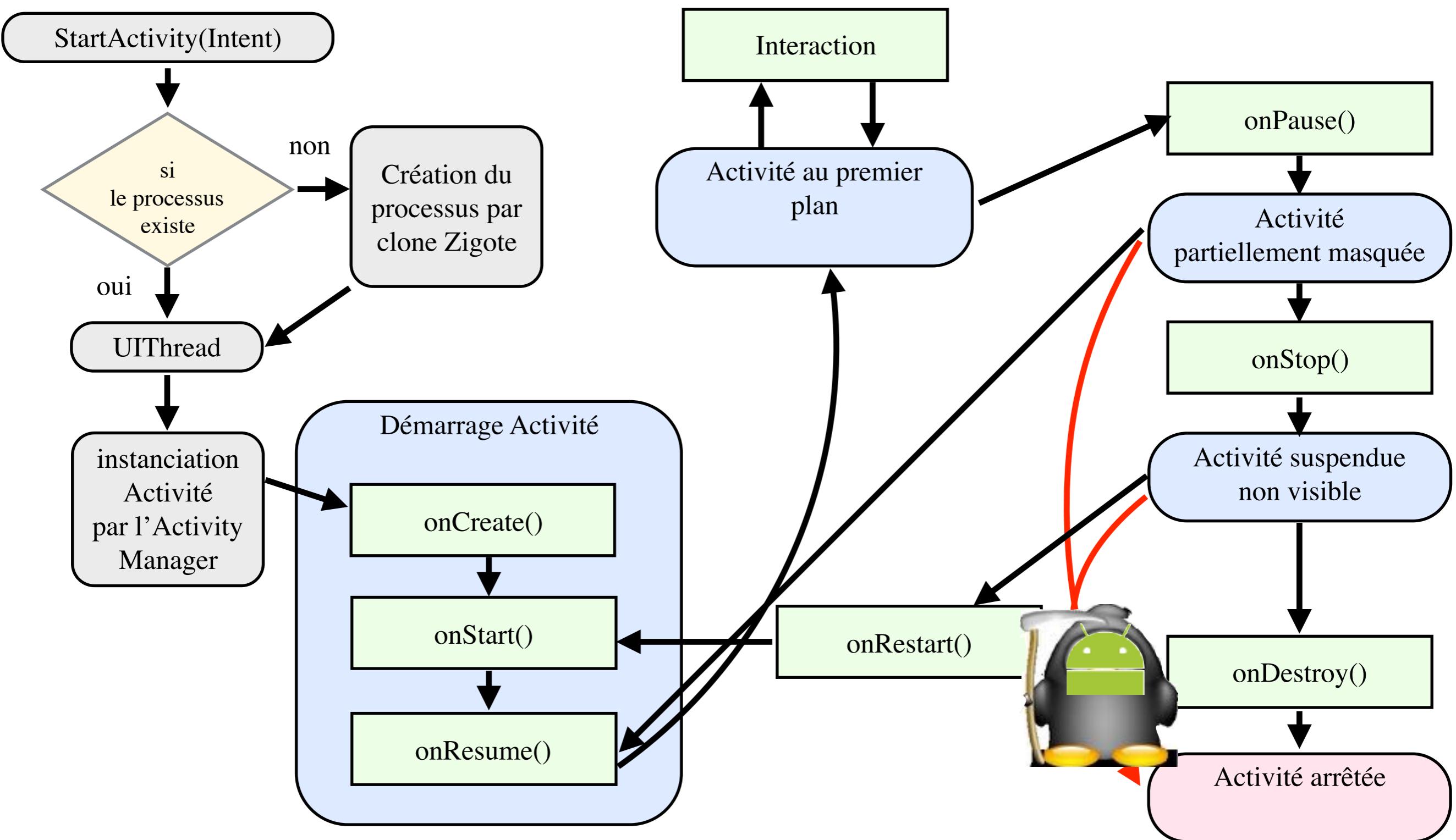
Cycle de vie d'une activité

5



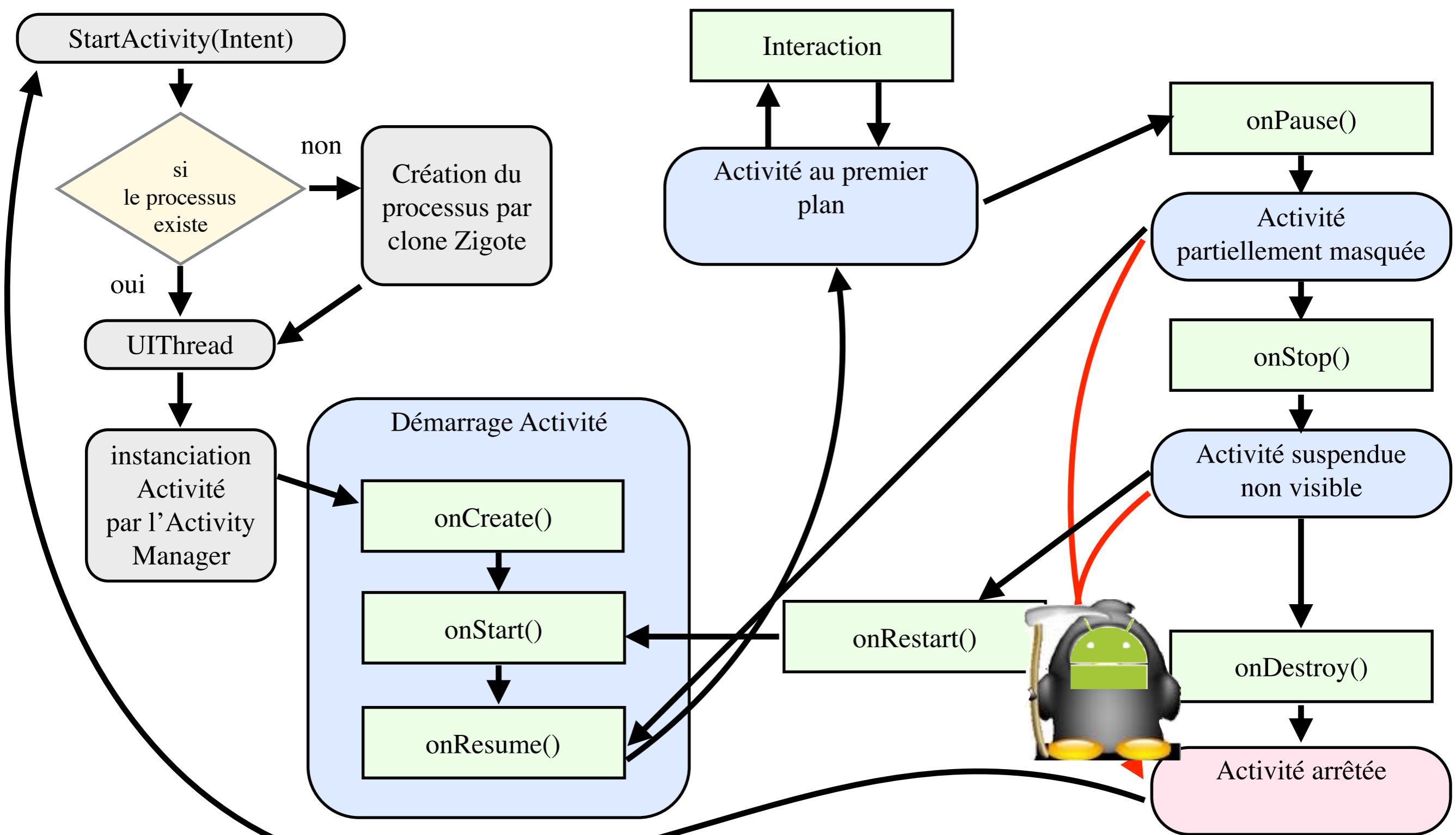
Cycle de vie d'une activité

5



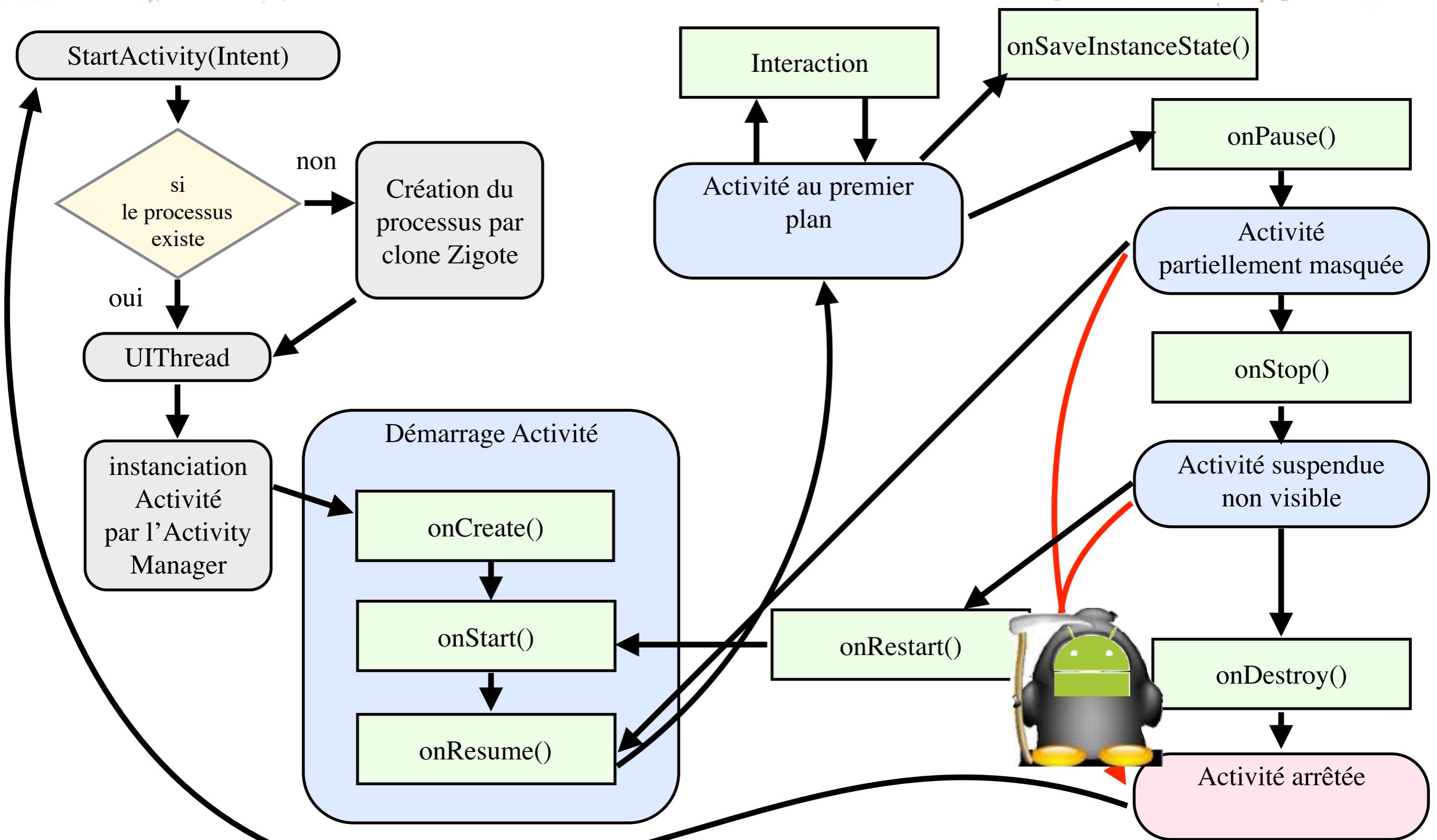
Cycle de vie d'une activité

5



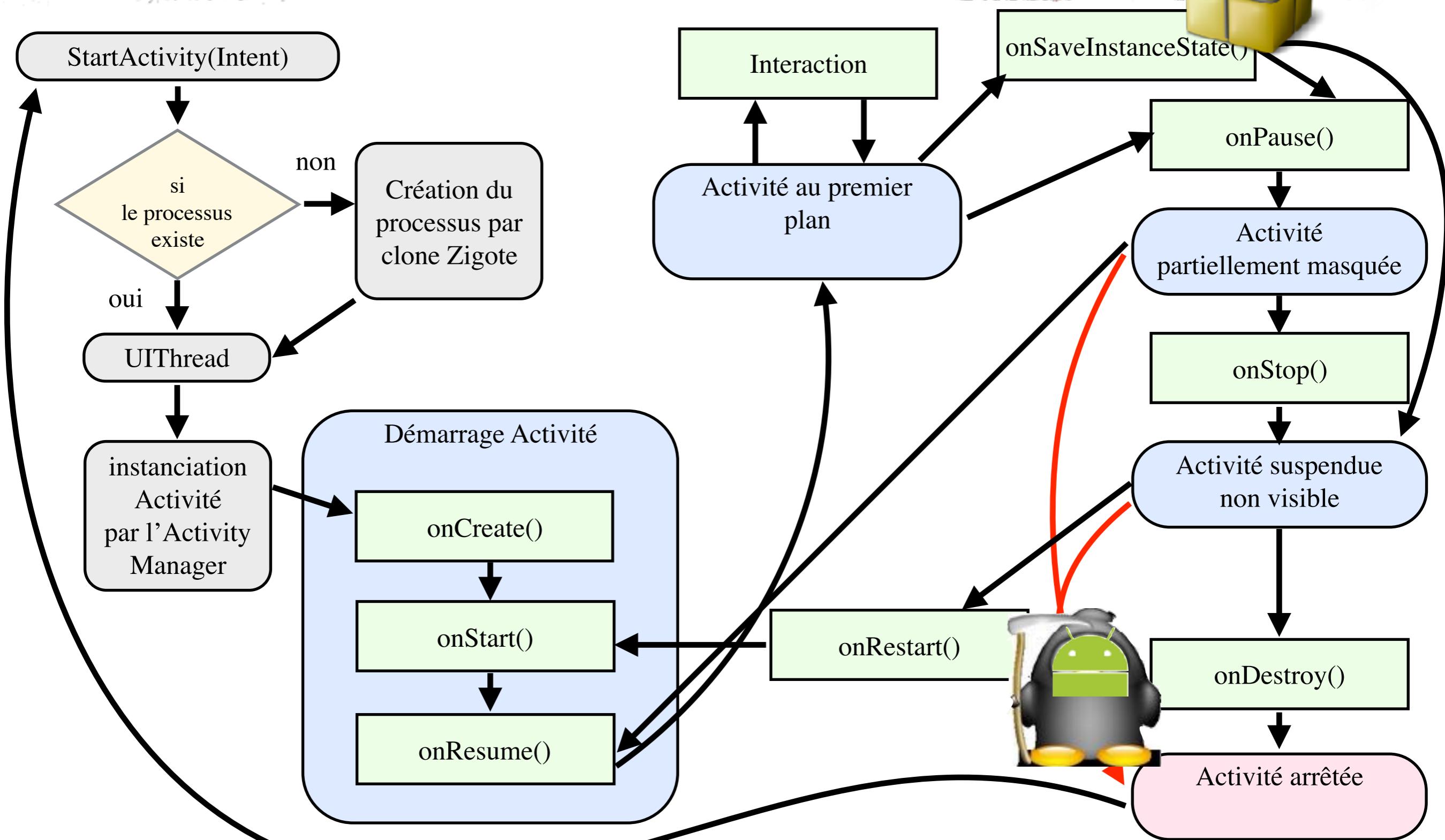
Cycle de vie d'une activité

5



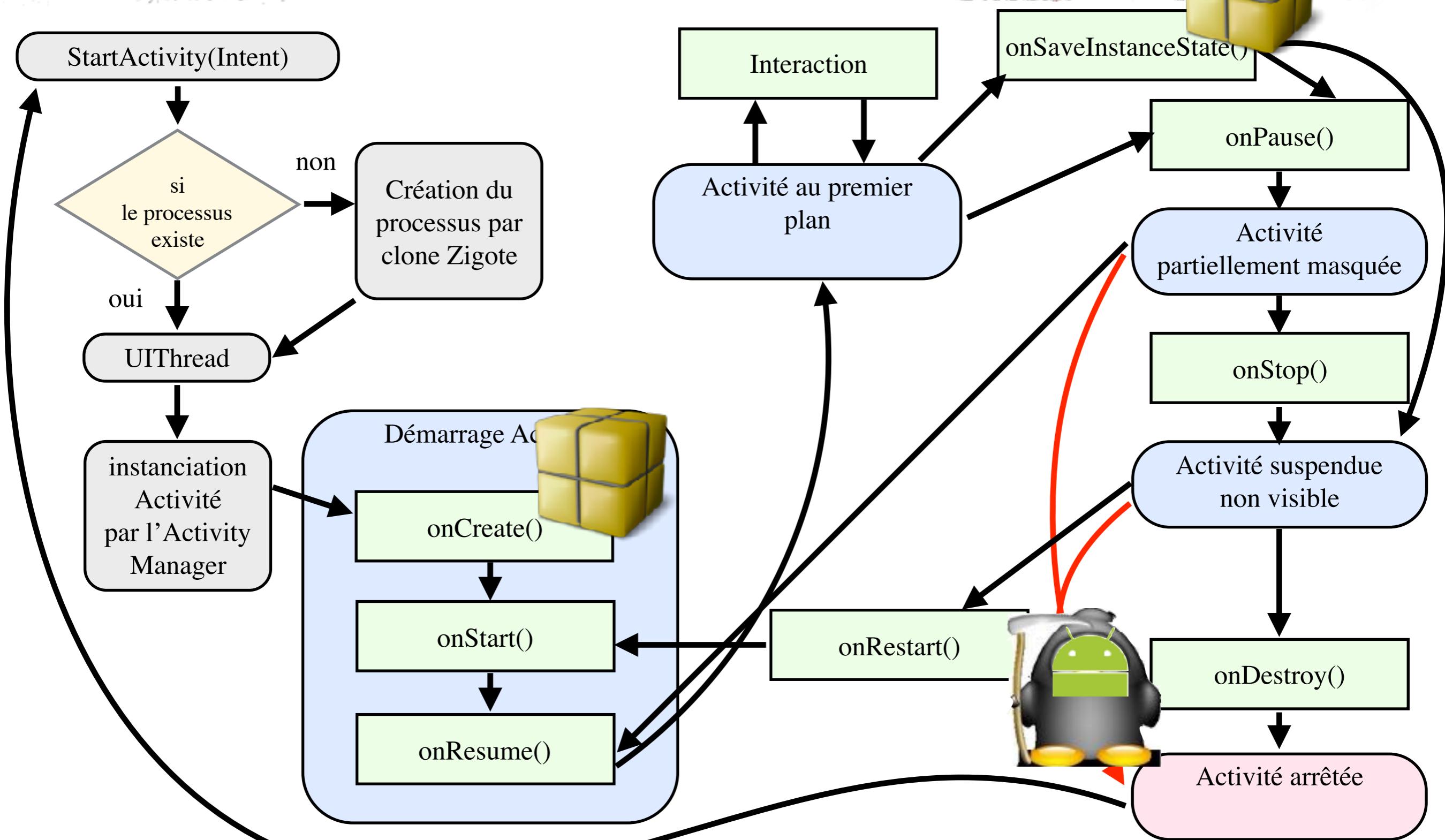
Cycle de vie d'une activité

5



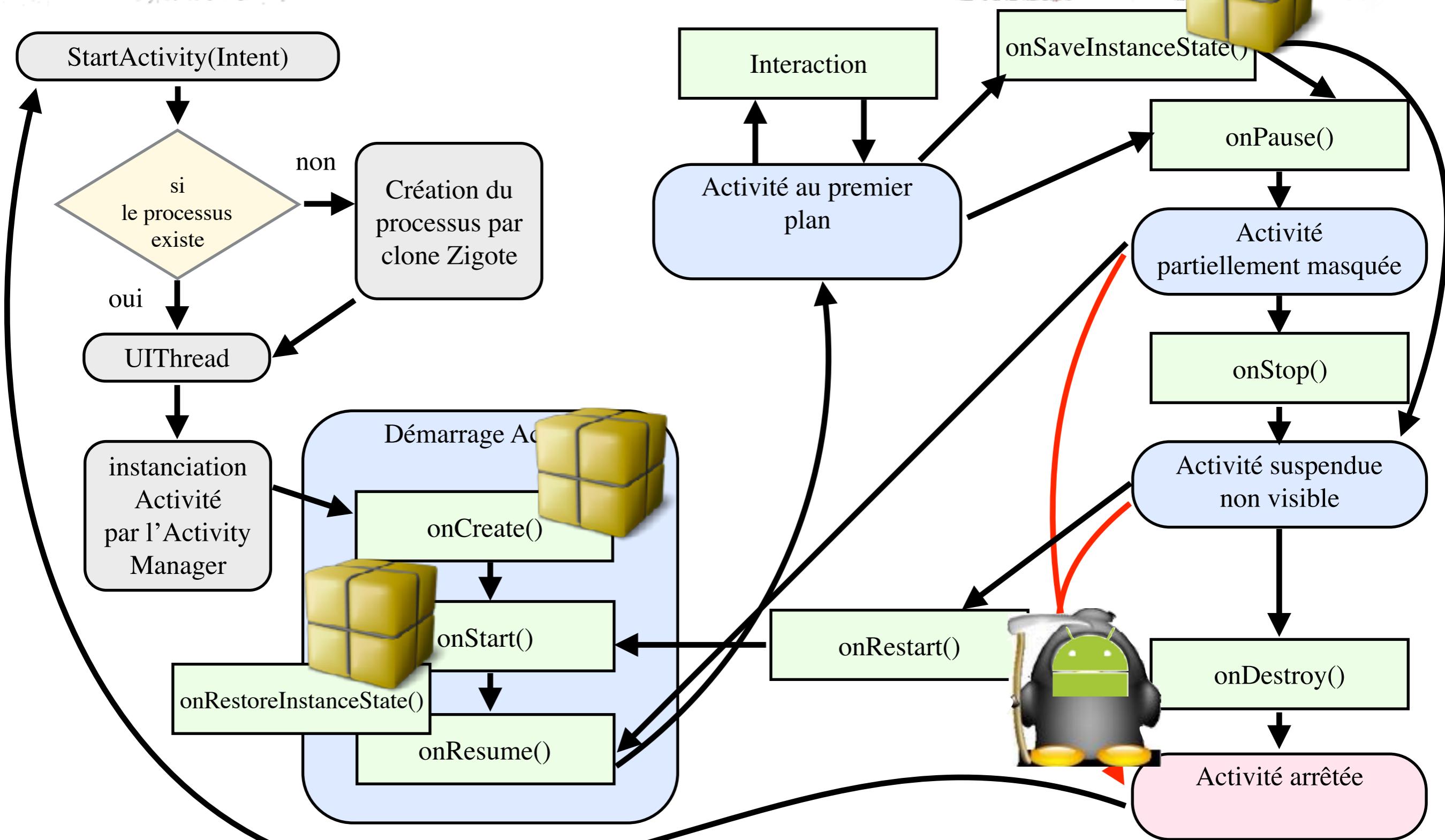
Cycle de vie d'une activité

5



Cycle de vie d'une activité

5



Gestion de l'état d'une activité

- Utilisation d'une ressource Bundle d'android :
android.os.Bundle
 - créé et maintenu par l'**ActivityManager** le **Bundle** est passé au (re) démarrage de l'activité
public void onCreate(Bundle b){...
 - Fonctionne comme une tableau associatif (dictionnaire) dont les clés sont des chaînes
getNombreuxTypes
putNombreuxTypes
 - Propose un mécanisme de “serialization” spécifique:
Parcelable, **Parcel...**

Les Activités

7

- Modèle du thread unique de gestion de l'affichage : le *UIThread* est géré par le système qui implante une boucle événementielle (il traite à chaque itération un élément d'une file d'événements) => toutes les opérations attachées à l'activité doivent-être brèves sous peine de bloquer le thread principale
- Un *timer* supervise une activité => arrêt forcé en cas de non réponse de l'activité à l'issu de la période de test
- Toute tâche complexe doit donc être déléguée à un *thread* spécifique ou un autre composant applicatif non graphique afin de préserver la réactivité du système

Exemple (HelloWorld)

8

- Contenu d'un projet Android :

```
./AndroidManifest.xml  
./ant.properties  
./build.xml  
./gen/fr/cnam/nfa025/BuildConfig.java  
./gen/fr/cnam/nfa025/R.java  
./res/layout/main.xml  
./res/values/strings.xml  
./src/fr/cnam/nfa025/HelloWorld.java
```

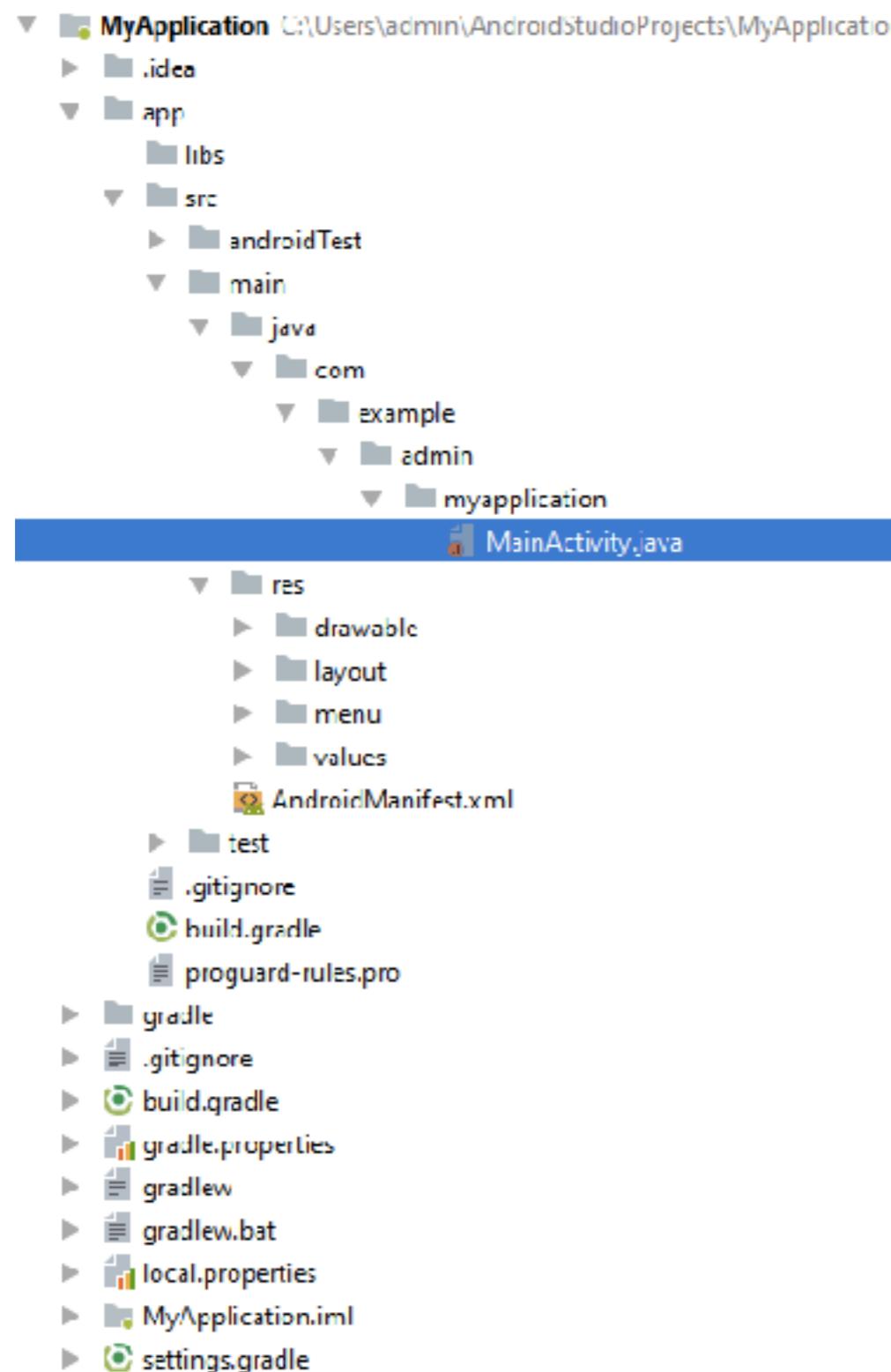
Légende :

En vert les fichiers que le développeurs va généralement éditer

En gris les fichiers automatiquement générés par le processus de compilation, il ne faut pas les éditer sinon les modifications seront perdues à la prochaine compilation.

Exemple (HelloWorld)

8



id :

hl

/BuildConfig.java
/R.java
hl
.xml
/HelloWorld.java

oppeurs va généralement éditer
ment générés par le processus de
diter sinon les modifications seront
tion.

Exemple (HelloWorld)

9

- Le fichier AndroidManifest.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="fr.cnam.nfa025"
4     android:versionCode="1"
5     android:versionName="1.0">
6     <application android:label="@string/app_name" >
7         <activity android:name="HelloWorld"
8             android:label="@string/app_name">
9             <intent-filter>
10                <action android:name="android.intent.action.MAIN"/>
11                <category android:name="android.intent.category.LAUNCHER"/>
12            </intent-filter>
13        </activity>
14    </application>
15 </manifest>
```

Exemple (HelloWorld)

10

- Le fichier main.xml contenant la description d'une interface très simple :

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android
3     ="http://schemas.android.com/apk/res/android"
4     android:orientation="vertical"
5     android:layout_width="fill_parent"
6     android:layout_height="fill_parent"
7     >
8     <TextView
9         android:layout_width="fill_parent"
10        android:layout_height="wrap_content"
11        android:text="Hello World, HelloWorld"
12        />
13 </LinearLayout>
```

Exemple (HelloWorld)

11

- Le fichier XML res/values/string.xml contient des définitions de chaînes de caractères :

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <string name="app_name">HelloWorld</string>
4 </resources>
```

Exemple (HelloWorld)

12

```
1  /* AUTO-GENERATED FILE.  DO NOT MODIFY.
2  *
3  * This class was automatically generated by the
4  * aapt tool from the resource data it found.  It
5  * should not be modified by hand.
6  */
7
8 package fr.cnam.nfa025;
9
10 public final class R {
11     public static final class attr {
12     }
13     public static final class layout {
14         public static final int main=0x7f020000;
15     }
16     public static final class string {
17         public static final int app_name=0x7f030000;
18     }
19 }
```

Exemple (HelloWorld)

13

```
1 import android.app.Activity;
2 import android.os.Bundle;
3 import android.util.Log;
4
5 public class HelloWorld extends Activity
6     public void onCreate(Bundle savedInstanceState){
7         super.onCreate(savedInstanceState);
8         this.setContentView(R.layout.main);
9         Log.i(APP_TAG, "CycleDeVieActivite is created!");
10        if(null != savedInstanceState){
11            Log.i(APP_TAG
12                 , "CycleDeVieActivite is restoring previous state!");
13        }
14    }
15 }
```

L'API du Logcat Android

14

- Le logcat est un service de collecte de messages de *débug* du système Android qui centralise tous les messages en un endroit unique (un module est ajouté spécifiquement au noyau Linux d'Android).
- Il se programme grâce à un outil (API) dérivé du mécanisme de Logging Java : `android.util.Log`.
- Les fonctions de base sont :
 - `Log.v()` : message de trace d'exécution
 - `Log.d()` : message de déverminage
 - `Log.i()` : message d'information
 - `Log.w()` : message d'erreur non critique
 - `Log.e()` : message d'erreur et exceptions