le cnam
Grand Est

TD1 USSI26 : Développement mobile

Sujet : Cycle de vie et composants d'application Android

1. QCM

- **Détaillez vos réponses.**
- **Aucune ou plusieurs réponses possibles.**

Question 1 : Qu'est-ce que Android ?

A : un langage de programmation.
Non.

B : un système d'exploitation mobile.
Oui.

C : une marque de Smartphone.
Non.

D : un framework de programmation.
Non.

Question 2 : Que peut-on retrouver dans le fichier « AndroidManifest.xml »?

A : Les états du cycle de vie de l'application
Non, le cycle de vie d'une application est géré par Android.

B : Les composants de l'application
Oui, on indique les activités, les services, les Broadcast Receivers, les Content Providers, etc.

C : Les ressources de l'application
Non, les ressources (images, vidéo, etc.) sont inclues dans le dossier « res/ » du projet Android.

D : La version minimum d'Android nécessaire pour exécuter l'application
Oui, cela permet de filtrer les smartphones/tablettes compatibles sur Google Play Store.

Question 3 : Parmi ces états, lesquelles sont des états possibles pour l'unité d'exécution d'une application Android ?

A : Created.
Non, cet état n'existe pas dans Android.

B : Active ou Running.
Oui, l'application est au premier plan.

C : Paused.
Oui, l'application n'a plus le focus et encore non arrêtée.

D : Canceled.

Non, cet état n'existe pas dans Android.

Question 4 : Android est basé sous un noyau Linux…

A : pour son abstraction matérielle.
Oui, principalement.

B : parce que chaque application s'exécute dans une machine virtuelle séparée.
NON ! C'est Android qui ajoute cette amélioration pour mobile.

C : parce qu'il est développé en Java.
Rien n'avoir

D : parce qu'il est développé en C.
Rien n'avoir

Question 5 : Justifiez ou non l'usage du NDK par chaque situation.

A : On souhaite développer une application sous Android et iPhone.
Oui, si un souhaite réaliser du code commun en C++ par exemple.

B : On souhaite utiliser le langage .NET pour le développement multiplateforme.
Surement pas du .NET

C : On souhaite réaliser une application hybride.
Ce n'est nullement justifier, mais si une partie de l'application est commune à une version iPhone et Android par exemple, pourquoi pas.

D : On souhaite réaliser une application web.
La non, si c'est web, c'est plus natif ou hybride.

Question 6 : Dalvik…

A : permet de construire des fichiers APK.
Aapt oui, Dalvik non.

B : permet de construire des fichiers PAK.
PAK ne correspond à rien sous Android

C : permet de construire des fichiers AKP.
AKP ne correspond à rien sous Android

D : est une machine virtuelle.
Oui, en cours de remplacement par ART.

Question 7 : Une application Android...

A : est exécuté en totalité dans le Thread principal UIThread.
Non, ce n'est pas le cas si on démarre un service avec un Thread spécifique.

B : qui dispose d'un long travail, doit le séparer du Thread principal UIThread.
Oui, c'est une bonne pratique pour éviter l'apparition du message « ANR ».

C : peut accéder à l'interface graphique hors du Thread principal UIThread.
Non, c'est surtout ce qu'il faut éviter, l'interface graphique ne doit être accessible que par le UIThread.

D : n'a pas la possibilité d'exécuter une tâche répétitive.
Non, pour cela on dispose de la classe *TimerTask*.

Question 8 : Un composant …

A : de type « Activity » possède une interface utilisateur
En résumé : une activité c'est ce qui est disponible à l'écran

B : de type « BroadCast Receiver » permet de gérer une base de données
Non, il écoute et réagit à certains messages.

C : de type « Service » peut être appelé depuis une activité pour fonctionner en arrière-plan.
Oui

D : de type « Content Provider » ne permet pas de partager à d'autres applications.
Si c'est le cas !

Question 9 : Une activité …

A : contient un seul fragment.
Non, elle peut contenir plusieurs fragments.

B : contient plusieurs fragments.
Non, elle peut contenir un seul fragment.

C : dispose d'un cycle de vie.
Oui, tout comme une activité avec quelques méthodes supplémentaires

D : dispose d'une seule mise en page.
Non, une activité peut avoir plusieurs mises en page différente en fonction de la zone d'affichage, type d'écran, orientation etc.

Question 10 : Un fragment …

A : est très pratique pour l'agencement sur tablette.
Oui mais pas que.

B : est réutilisable.
Oui, on essaye de faire en sorte

C : dispose d'un cycle de vie.
Oui, comme l'activité

D : dispose ou non d'une interface.
Oui, même si souvent elle est présente.

Question 11 : une intention permet de …

A : démarrer une activité.
Oui avec startActivity() par exemple

B : démarrer un service.
Oui avec startService() par exemple

C : démarrer une interface utilisateur.
Oui car une activité possède une interface utilisateur

D : démarrer un récepteur de services.
Un récepteur de services ???

Question 12 : Une intention…

A : implicite permet de démarrer une activité grâce à la méthode *startActivity()*.
Non, c'est plutôt le cas pour une intention explicite.

B : qui est appelée avec la méthode *startActivityForResult()* permet d'attendre un retour de l'activité lancée.
Oui, c'est fait pour ça, contrairement à *startActivity()* où aucun retour n'est attendu.

C : sera nécessairement exécutée directement et ne pourra pas être mis en attente.
Non, on a pour ça la classe *PendingIntent*.

D : explicite sera exécutée si le composant dispose du bon filtre dans la balise *<intent-filter>*.
Non, c'est plutôt le cas pour une intention implicite.

Question 13 : Un filtre d'intention…

A : se déclare uniquement dans le manifest.
Pas pour les BroadcastReceiver.

B : est utilise pour les intentions explicites
Non, pas besoin de filtre dans ce cas.

C : est utilise pour les intentions implicites
Oui, totalement.

D : contient toujours une seule action.
Non, il peut y avoir plusieurs actions, comme plusieurs catégories et plusieurs données.

Question 14 : Un *BroadcastReceiver*…

A : passera par la méthode *onCreate()* lors de son lancement la première fois.
Non, un *BroadcastReceiver* ne dipose que de la méthode *onReceive()*.

B : ne pourra s'enregistrer à une action que grâce à un *<intent-filter>* dans le fichier manifest.
Non, c'est possible grâce à la méthode *registrerReceiver()* en Java.

C : sera désinscrit à l'aide de la méthode *unregistrerReceiver()*.

Oui, si l'enregistrement s'est fait dynamiquement dans le code Java.

D : peut s'enregistrer à des actions personnalisées.
Oui, il est possible de s'enregistrer à sa propre action personnalisée.

Question 15 : un service local…

A : est dit démarrer.
Oui.

B : se lance avec *startService()*.
Oui.

C : peut être lié.
Oui, à la fois démarré et lié c'est possible mais il ne faudra pas oublier de l'arrête même si plus personne n'y est attaché.

D : s'arrête avec *stopService()*.
Oui ou avec stopSelf().

Question 16 : Vrai ou faux ?

A : *IntentService* exige la gestion d'un thread.
Non, cette classe gère déjà un thread.

B : *Service* gère une file d'attente.
Non, c'est *IntentService*

C : *IntentService* gère une file d'attente.
Oui.

D : *Service* ne permet pas l'exécution simultanée.
Si, il faut bien penser à threader correctement pour éviter les ANR.

Question 17 : Un service…

A : créé à partir de la classe *IntentService* sera exécuté dans le thread principal de l'application (*UIThread*).
Non, c'est plutôt le cas pour la classe *Service*, un *IntentService* sera automatiquement exécuté dans un thread à part.

B : local démarrera directement avec l'appel de la méthode *onStartCommand()*.
Non, la première méthode appelée est *onCreate()* dans le cycle de vie du service local.

C : distant peut-être appelé par un composant d'une autre application.
Oui, grâce à la méthode *bindService()*.

D : distant doit être détaché si on en n'a plus besoin.
Oui, c'est de la bonne pratique en utilsant la méthode *unbind()*.

Question 18 : Le store applications Google Play Store…

A : permet la distribution gratuite d'applications pour Android.
Oui, c'est l'inscription qui est payante.

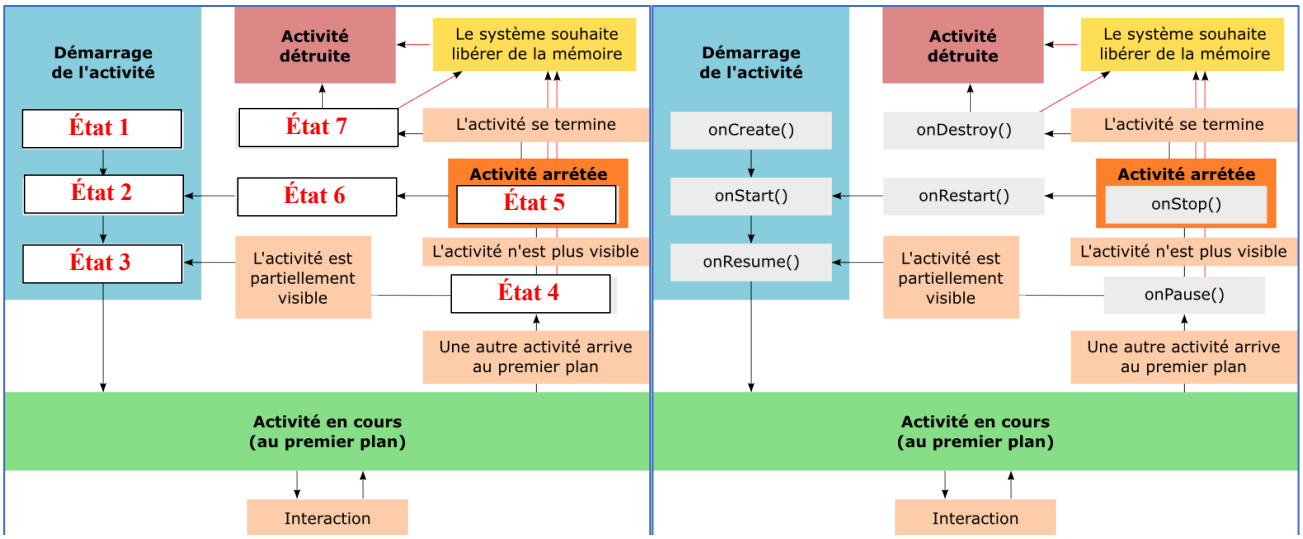
B : permet la publication immédiate d'applications Android.
Oui, c'est le cas pour Google Play Store, mais il y a une étape de vérification pour d'Apple.

C : Android permet à tous les smartphones d'accéder à toutes les applications disponibles.
Non, ils filtrent sur les applications compatibles (version de l'OS, capteurs indispensables, etc.).

D : Android permet de gérer les mises à jour, statistiques et les achats intégrés.
Oui, c'est fait en grande partie pour ça.

2. Cycle de vie d'application Android

Question 19 : Complétez les noms des différents états du cycle de vie d'une application Android.



Soit l'extrait du code suivant d'une classe JAVA :

Extrait de la classe MainActivity

```
public class MainActivity extends Activity {

    final String TAG = "Cycle de vie";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.i(TAG,"onCreate");
    }
```

```
    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.i(TAG,"onDestroy");
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.i(TAG,"onCreate");
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.i(TAG,"onStop");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.i(TAG,"onPause");
    }

    @Override
    protected void onResume() {
        super.onResume();
        Log.i(TAG,"onResume");
    }

    @Override
    public void onBackPressed() {
        super.onBackPressed();
        Log.i(TAG,"onBackPressed");
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Log.i(TAG,"onRestart");
    }
...
```

Question 20 : Écrivez dans l'ordre les messages affichés dans le LogCat au premier démarrage de mon application ?

```
Cycle de vie : onCreate
Cycle de vie : onStart
Cycle de vie : onResume
```

Question 21 : Quelle manipulation devrais-je faire pour obtenir les messages suivants ?

```
Cycle de vie : onBackPressed
```

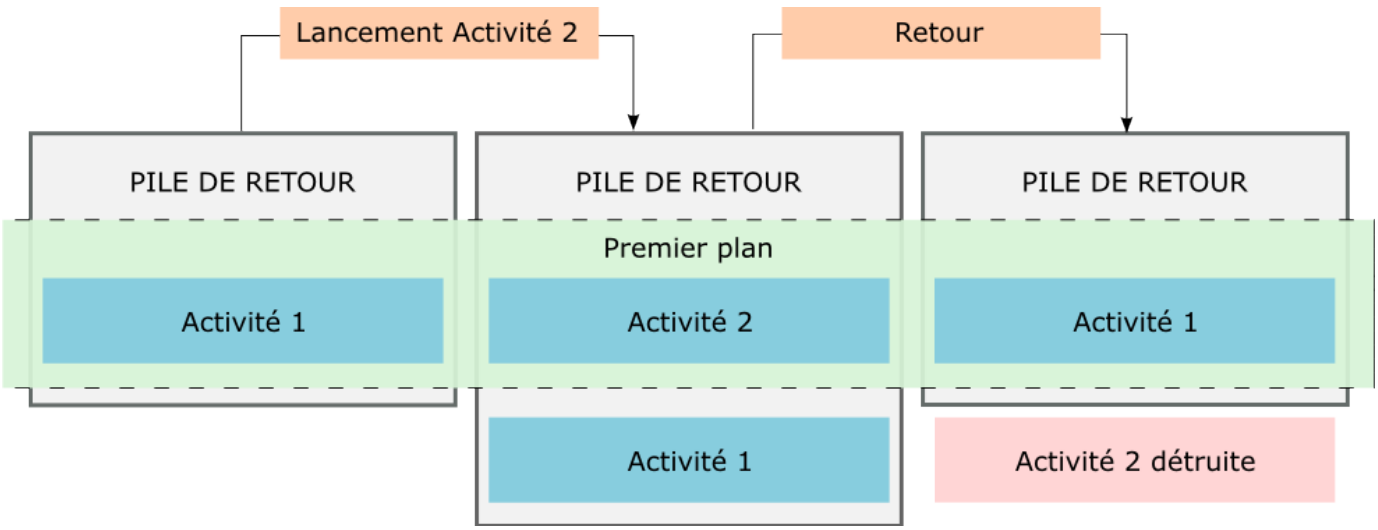
Réponse : Appuyez sur le bouton retour en arrière de l'appareil. (flèche arrière)

```
Cycle de vie : onPause
Cycle de vie : onStop
Cycle de vie : onRestart
Cycle de vie : onStart
Cycle de vie : onResume
Cycle de vie : onBackPressed
Cycle de vie : onPause
Cycle de vie : onStop
Cycle de vie : onDestroy
```

Réponse : Après avoir lancé la première fois l'application. Un appui sur la touche « Home » permet d'obtenir les *onPause* puis *onRestart*. Ensuite, une réouverture de l'application puis un appui la touche retour permet d'obtenir les messages suivants (Attention, ici le Log dans le onStart est volontairement erroné !)

Question 22 : A quoi correspond `@Override` ?
A la surcharge des méthodes du cycle de vie d'une application Android

Question 23 : Schématisez la pile de retour Android (2 activités suffiront pour l'illustration)



3. Les intentions dans Android

Question 24 : Quels sont les différents types d'intentions ? Donnez leurs définitions.

Réponse :
1)- Les intentions explicites : ces intentions précisent explicitement le nom du composant que l'application souhaite démarrer. Ce nom est le nom complet de la classe correspondante.
2)- Les intentions implicites : ces intentions ne précisent pas explicitement le nom de la classe mais une action générale à effectuer. Ainsi, une autre application pourra correspondre à votre demande et y répondre pour la gérer.

Question 25 : Écrivez le code d'un Intent qui démarre une activité qui s'appelle « SecondActivity ».

```
Intent intention = new Intent(this,SecondActivity.class);
startActivity(intention);
```

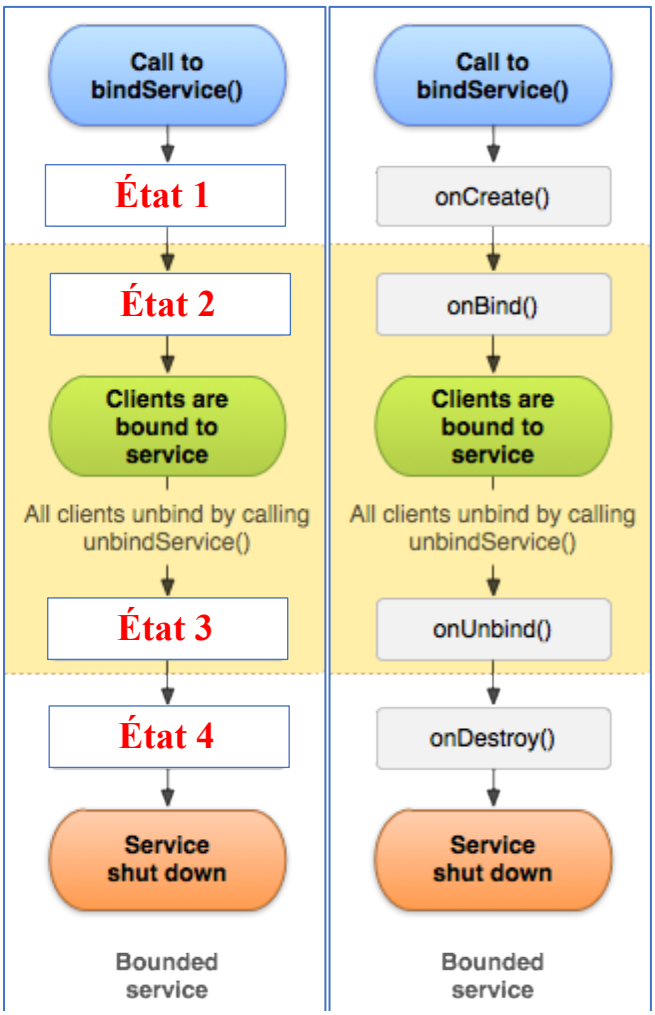
4. Les services

Question 26 : Décrivez ces différents attributs disponibles pour la déclaration d'un service dans le fichier manifest.

Attribut	Description
Enabled	- Le service peut être instancié par le système - True par défaut
Exported	- Autorise le démarrage depuis une autre application - True par défaut
Icon	- l'icône du service - référence à une ressource graphique (drawable)
IsolatedProcess	- exécuter le service dans un processus isolé du reste du système
Label	- nom du service affiché à l'utilisateur - référence à une « String ressource »
Name	- nom de la sous-classe service - Obligatoire

Permission	- Liste les autorisations nécessaires. - Si aucune, les autorisations de l'application sont appliquées.
Process	- le nom du processus où le service est exécuté. - Généralement c'est le processus courant. - Exemple *android:process= «:nom_du_nouveau_processus»*

Question 27 : Compléter le cycle de vie d'un service distant.



Question 28 : Écrivez l'exemple d'un code de démarrage d'un service local « MyIntentService » grâce à un *Intent*.

```
Intent intent = new Intent(Activity.this, MyIntentService.class);
startService(intent);
```

Question 29 : Écrivez l'exemple d'un code permettant de de se lier à un service distant « CnamService » grâce à un *Intent*.

```
Intent intent = new Intent(this, CnamService.class);
bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
```

5. Les récepteurs d'intentions

Question 30 : Écrivez le code minimal d'une classe de type *BroadcastReceiver* qui s'appelle *DateReceiver*.

```
public class DateReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        // Ici on met le code à éxécuter lorsque le récépteur est appelé.
    }
}
```

Question 31 : On souhaite inscrire ce *BroadcastReceiver* à l'action « *ACTION_DATE_CHANGED* ». Écrivez le code d'inscription dans le fichier manifest.

```
<receiver
    android:name=".DateReceiver"
    android:enabled="true"
    <intent-filter>
        <action android:name="android.intent.action.ACTION_DATE_CHANGED" />
    </intent-filter>
</receiver>
```

Question 32 : On souhaite faire appel à ce *BroadcastReceiver* depuis l'extérieur de l'application, que faut-il modifier à cette inscription dans le fichier manifest?

```
<receiver
    android:name=".DateReceiver"
    android:enabled="true"
    android:exported="true"
    <intent-filter>
        <action android:name="android.intent.action.ACTION_DATE_CHANGED" />
    </intent-filter>
</receiver>
```

Question 33 : A la place de le faire dans le fichier manifest, on souhaite faire l'inscription dans le code Java. Écrivez le code de la déclaration, l'inscription et la désinscription du même *BroadcastReceiver*?

```
// Dans la méthode onCreate :

DateReceiver receiver = new DateReceiver();

IntentFilter intentFilter = new IntentFilter("android.net.conn.ACTION_DATE_CHANGED");

// Inscription :

@Override
protected void onResume() {
    super.onResume();
```

```
    registerReceiver(receiver, intentFilter);
}

// Désinscription :

@Override
protected void onPause() {
    super. onPause();
    unregisterReceiver(receiver);
}
```

6. Exemple d'application :

Énoncé : Vous êtes en charge de la conception d'une application Smartphone (Android) de consultation de biens immobiliers. Une agence immobilière dispose d'un annuaire d'offres de biens (location et vente) et souhaite la mettre à disposition sur terminal smartphone. La recherche des offres doit se faire par ville ET à partir de la position courante (géolocalisation à partir de la position connue du GPS et sur un rayon à définir par l'utilisateur). L'utilisateur doit pouvoir s'abonner à un service de notifications PUSH, télécharger les photos des biens dans son téléphone et contacter l'agent en charge de ce bien (mail et téléphone).

Question 34 : Organisez sur papier l'architecture de l'application.

Réponse :

1)- Répertoriez les Activités / les Services / Les intents / Les content providers :

1. ACTIVITES :
    a. Recherche de biens
    b. Consulter les biens
    c. Consulter un bien
    d. S'abonner à un service de notifications
2. SERVICES : Recherche de biens en temps réel
3. INTENTIONS (INTENTS) : Notification PUSH
4. CONTENT PROVIDERS : Stockage de photos

2)- Définissez les différentes vues :

1. Recherche de biens
2. Consulter les biens
3. Consulter un bien
4. S'abonner à un service de notifications

3)- Définissez les fonctionnalités utilisées.

1. GPS
2. Internet
3. Appels téléphoniques

4)- Définissez le stockage des données : != types de données

- Celles qui ne bougent pas (contact agence, les photos, les textes..etc)
- Celles qui bougent
    - les biens -> base de données
    - les abonnements -> préférences