SYM Labo 1

Auteurs

Antoine Drabble & Patrick Djomo

Question 1.

Comment organiser les textes pour obtenir une application multi-langues (français, allemand, italien, anglais, langue par défaut : anglais) ?

Dans le dossier values, il faut créer des fichiers xml contenant des strings en utilisant l'assistant de création et ajouter le qualifier "local" en choisissant la langue. On peut créer autant de fichier qu'on veut pour chaque langue/région et même changer le texte en fonction de l'orientation de l'écran ou autre en utilisant les qualifiers. Android va toujours choisir la traduction la plus spécifique possible pour l'appareil utilisé. Le fichiers strings.xml est le fichier par défault et sera utilisé quand aucun autre language correspond.

Question 2.

Dans l'exemple fourni, sur le dialogue pop-up, nous affichons l'icône android.R.drawable.ic_dialog_alert, disponible dans le SDK Android mais qui n'est pas très bien adaptée visuellement à notre utilisation. Nous souhaitons la remplacer avec notre propre image, veuillez indiquer comment procéder. Dans quel(s) dossier(s) devons-nous ajouter cette image ? Décrivez brièvement la logique derrière la gestion des ressources de type « image » sur Android. Info : Google met à disposition des icônes open source dans le style « Material Design » utilisé actuellement sur Android : https://design.google.com/icons/

Afin de modifier le logo de l'application, on commence par télécharger l'icone. On la place ensuite dans le dossier correspondant (drawable) du projet. Finalement, il faut ouvrir le fichier AndroidManifest.xml et changer la propriété android:icon de l'élément application (android:icon="@drawable/ic_info_black_24dp").

Question 3.

Lorsque le login est réussi, vous êtes censé chaîner une autre Activity en utilisant un Intent. Si je presse le bouton "Back" de l'interface Android, que puis-je constater ? Comment faire pour que l'application se comporte de manière plus logique ?

Quand on clique sur le bouton back, l'application se ferme. Ceci est dû à ce bout de code dans l'activité principale.

```
Intent intent = new Intent(MainActivity.this, SuccessActivity.class);
intent.putExtra(SuccessActivity.emailEntered, mail);
intent.putExtra(SuccessActivity.passwordGiven, passwd);
```

```
startActivity(intent);

Toast.makeText(MainActivity.this, getResources().getString(R.string.good),
Toast.LENGTH_LONG).show();
finish();
```

On voit que dans la dernière ligne on tue l'activité principale, ce qui fait que quand on clique sur back dans la deuxième activité, on arrête la deuxième activité et comme l'activité principale a été terminée on quitte l'application. Si l'on enlève ce finish(), quand on clique sur back on retournera sur l'activité principale.

Ce qu'on voudrait c'est que l'application se mette en arrière plan ou simplement que le bouton back ne fasse rien. Pour que l'application se mette en arrière plan il faut rajouter le code suivant à l'activité:

```
@Override
public void onBackPressed() {
    moveTaskToBack(true);
}
```

Et pour que le bouton back ne fasse rien, il faut rajouter ce code :

```
@Override
public void onBackPressed()
{
   // Your Code Here. Leave empty if you want nothing to happen on back press.
}
```

Question 4.

On pourrait imaginer une situation où cette seconde Activity fournit un résultat (par exemple l'IMEI ou une autre chaîne de caractères) que nous voudrions récupérer dans l'Activity de départ. Comment procéder ?

On a plusieurs possibilités. On peut utiliser les SharedPreferences pour sauvegarder la valeur calculée sur le téléphone. On peut aussi utiliser la méthode setResult qui permet d'envoyer un résultat à l'activité.

Dans l'activité principale, on crée une nouvelle activité.

```
Intent i = new Intent(this,  Activity2.class);
startActivityForResult(i, 1);
```

Dans l'activité crée on calcule la valeur, on la met dans un intent qu'on retourne à l'activité principale.

```
Intent intent = new Intent();
intent.putExtra("edittextvalue","value_here")
setResult(RESULT_OK, intent);
finish();
```

Dans l'activité principale on récupère le résultat.

```
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == 1) {
         if(resultCode == RESULT_OK){
             String stredittext=data.getStringExtra("edittextvalue");
         }
    }
}
```

Question 5.

Dans l'activité de login, en plaçant le téléphone (ou l'émulateur) en mode paysage (landscape), nous constatons que les 2 champs de saisie ainsi que le bouton s'étendent sur toute la largeur de l'écran. Veuillez réaliser un layout spécifique au mode paysage qui permet un affichage mieux adapté et indiquer comment faire pour qu'il soit utilisé à l'exécution.

Afin de changer le layout en mode paysage, il faut faire un clique droit dans le dossier /res/layout et créer un nouveau layout, il faut ensuite lui donner le même nom que le layout auquel on veut ajouter le landscape et dans l'utilitaire de création, il faut ajouter le qualifier d'orientation et selectionner landscape. On devrait maintenant pouvoir mettre l'écran on mode landscape et voir le deuxième template être utilisé.

Question 6.

Le layout de l'interface utilisateur de l'activité de login qui vous a été fourni a été réalisé avec un LinearLayout à la racine. Nous vous demandons de réaliser un layout équivalent utilisant cette fois-ci un RelativeLayout.

Voilà le relative layout équivalent.

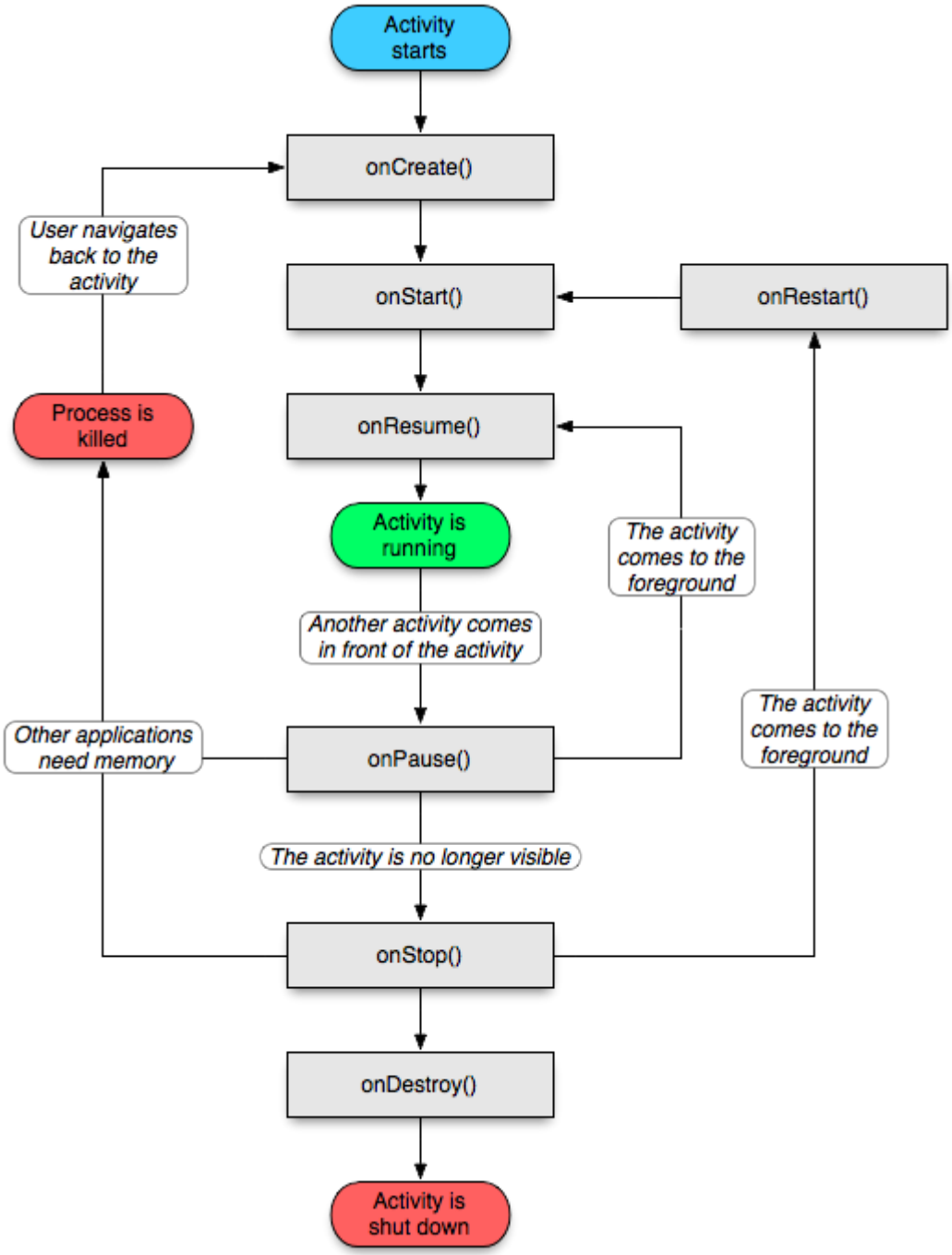
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding = "10dp">


    <EditText
        android:hint="E-mail"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:id="@+id/email"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:inputType="textEmailAddress" />

    <EditText
        android:hint="Password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:ems="10"
        android:layout_below="@+id/email"
        android:layout_alignLeft="@+id/email"
        android:layout_alignStart="@+id/email"
        android:layout_marginTop="12dp"
        android:id="@+id/password" />

    <Button
        android:text="Button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/password"
        android:layout_alignLeft="@+id/password"
        android:layout_alignStart="@+id/password"
        android:id="@+id/buttOk" />

</RelativeLayout>
```

Question 7.

Implémenter dans votre code les méthodes onCreate(), onStart(), onResume(), onPause(), onStop(), etc... qui marquent le cycle de vie d'une application Android, et tracez leur exécution. Décrivez brièvement à quelles occasions ces méthodes sont invoquées. Si vous aviez (par exemple) une connexion Bluetooth (ou des connexions bases de données, ou des capteurs activés) ouverte dans votre Activity, que faudrait-il peut-être faire, à votre avis (nous ne vous demandons pas de code ici) ?

Les explications quant à l'invocation des méthodes de l'image ci-dessous est décrite dans la javadoc de l'application.



Dans le cas de l'utilisation de base de données, on voudrait par exemple ouvrir la connection à la base de données dans le onStart et arrêter la connection dans le onStop. Avec des capteurs, ou aussi avec la base de données, on voudra peut-être démarrer le capteur dans le onResume et l'arrêter dans le cas d'un onPause.

Avec une connexion bluetooth dans notre activité, tout d'abord nous devons ouvrir la connexion bluetooth lors de l'appel des méthode onCreate() ou onStart() afin de permettre par exemple à d'autres mobiles de communiquer avec notre activité. Ensuite lorsque l'activité n'est plus visible par l'utilisateur, nous devons enregistrer son état, couper la connexion dans la méthode onPause().Enfin lorsque l'activité sera à nouveau reprise, nous allons restaurer son état et établir une nouvelle connexion dans la méthode onResume().

Question 8.

Facultatif, s’il vous reste du temps, nous vous conseillons de le consacrer à mettre en place la résolution des permissions au runtime