Trimestre Hiver, 2018

Mohamed Lokbani

IFT1155 – Examen Intra –

Inscrivez tout de suite : votre nom et le code permanent.

Nom : _____| Prénom(s) : _____|

Signature : _____| Login : _____|

Date : Vendredi 23 février 2018

Durée : 2 heures (de 16h30 à 18h30)

Local : Z-350, Pavillon Claire McNicoll

Directives :

- Toute documentation est permise.
- Calculatrice **non** permise.

- Répondre directement sur le questionnaire.
- Les réponses **doivent être brèves, précises, claires et nettement présentées.**

1._____/20

2._____/20

3._____/20

4._____/40

Total : _____/100

Directives officielles

* Interdiction de toute communication verbale pendant l'examen.

* Interdiction de quitter la salle pendant la première heure.

* L'étudiant qui doit s'absenter après la première heure remettra sa carte d'étudiant au surveillant, l'absence ne devant pas dépasser 5 minutes. Un seul étudiant à la fois peut quitter la salle.

* Toute infraction relative à une fraude, un plagiat ou un copiage est signalée par le surveillant au directeur de département ou au professeur qui suspend l'évaluation.

F.A.S

Exercice 1 (20 points) Répondez par « vrai » ou « faux » en y incluant une très courte explication.

1.1 [VRAI | FAUX] Nougat est la dernière version d'Android.

1.2 [VRAI | FAUX] Si on veut cibler le maximum d'appareils, on doit développer pour une taille d'écran normal.

1.3 [VRAI | FAUX] On n'a pas besoin de réaliser un traitement « spécial » pour sauvegarder les données sensibles quand une activité passe en mode repos.

1.4 [VRAI | FAUX] Il n'y a qu'une seule manière pour accéder aux gestionnaires de logs d'Android, « logcat ».

1.5 [VRAI | FAUX] Dans un terminal ou une console « DOS », la commande pour lister tous les appareils Android connectés à votre ordinateur est « android --list ».

1.6 [VRAI | FAUX] Quand plusieurs appareils sont connectés à votre ordinateur, on ne peut pas ouvrir une invitée de commandes (shell) spécifique pour chaque appareil.

1.7 [VRAI | FAUX] On lance d'abord une activité basique sur l'émulateur. Par la suite, on clique sur le bouton « retour » (back). L'activité passe uniquement par les deux étapes « onPause » et « onStop ».

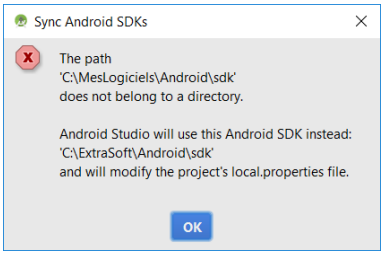
1.8 [VRAI | FAUX] L'activité principale est démarrée à l'aide d'un « intent ».

1.9 [VRAI | FAUX] « RelativeLayout » permet de disposer les éléments en 1 ligne ou 1 colonne dans l'ordre où ils sont définis dans le fichier « XML ».

1.10 [VRAI | FAUX] Pour chaque taille à supporter, il faut un fichier « layout » de même nom.

Exercice 2 (20 points)

2.1 Expliquer le pourquoi de ce message et dans quelle situation il peut se produire.



2.2 Un projet Android contient les 3 fichiers « gradle » suivants: « build.gradle » au niveau du projet, « build.gradle » au niveau du module et «settings.gradle » à la racine du projet. Expliquer l'utilité de ces 3 fichiers.

2.3 Dans le fichier « build.gradle » de votre module, à quoi sert ce bloc et quel est l'utilité de l'élément qu'il contient.

```
dependencies {
  implementation 'com.android.support:support-v4:27.0.2'
}
```

2.4 Commencer par nous expliquer l'utilité de « lint » par la suite, expliquer l'effet des instructions ci-dessus. À noter que ce bloc est défini dans le fichier « build.gradle » de votre module.

```
lintOptions {
  disable 'AllowBackup'
  baseline file("AndroidManifest.xml")
}
```

Exercice 3 (20 points) dessiner l'interface graphique à partir du fichier XML ci-dessous dont la syntaxe est correcte, en prenant soin d'inclure un résumé de votre démarche.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  tools:context=".MainActivity">
  <TextView
    android:id="@+id/TextView0"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:text="TITRE"
    android:background="#00FF00"
    android:layout_margin="10dp"/>
  <LinearLayout
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal"
    android:background="#FFFF00">
    <TextView
      android:id="@+id/TextView1"
      android:layout_height="100dp"
      android:layout_width="0dp"
      android:layout_weight="1"
      android:text="Texte1"
      android:background="#FF0000"
      android:layout_margin="10dp"/>
    <TextView
      android:id="@+id/TextView2"
      android:layout_height="100dp"
      android:layout_width="0dp"
      android:layout_weight="1"
      android:text="Texte2"
      android:background="#00FF00"
      android:layout_margin="10dp"/>
    <TextView
      android:id="@+id/TextView3"
      android:layout_height="100dp"
      android:layout_width="0dp"
      android:layout_weight="1"
      android:text="Texte3"
      android:background="#0000FF"
      android:layout_margin="10dp"/>
  </LinearLayout>
  <LinearLayout
    android:id="@+id/LinearLayout2"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:background="#FFFFFF"
    android:orientation="horizontal">
    <TextView
      android:id="@+id/TextView4"
      android:layout_height="100dp"
      android:layout_width="wrap_content"
      android:text="Texte4"
      android:background="#FF0000"
      android:layout_margin="10dp"/>
    <TextView
      android:id="@+id/TextView5"
      android:layout_height="100dp"
      android:layout_width="wrap_content"
      android:text="Texte5"
      android:background="#0000FF"
      android:layout_margin="10dp"/>
```

Code des couleurs

vert : 00FF00

jaune : FFFF00

rouge : FF0000

vert : 00FF00

bleu : 0000FF

blanc : FFFFFF

rouge : FF0000

bleu : 0000FF

```
    <LinearLayout
      android:id="@+id/LinearLayout3"
      android:layout_width="0dp"
      android:layout_height="100dp"
      android:layout_weight="1"
      android:background="#000000"
      android:orientation="vertical"
      android:layout_margin="10dp">
      <TextView
        android:id="@+id/TextView6"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:text="Texte6"
        android:background="#FF0000"
        android:layout_margin="10dp"/>
      <TextView
        android:id="@+id/TextView7"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:text="Texte7"
        android:background="#0000FF"
        android:layout_margin="10dp"/>
    </LinearLayout>
  </LinearLayout>
</LinearLayout>
```

noir : 000000

rouge : FF0000

bleu : 0000FF

Exercice 4 (40 points) nous allons examiner l'état de la batterie d'un appareil Android. Pour cela, nous allons utiliser les différentes méthodes et constantes définies dans la classe « BatteryManager ». Cette classe est décrite en annexe.

4.1 Écrire le code de la méthode « **String getPlugTypeString(int)** ». Cette méthode retourne la nature de la connexion de la batterie. La variable de retour peut contenir l'un des types mentionnés en annexe : « AC », « USB », « Inconnue », « Wireless ».

4.2 Écrire le code de la méthode « **String getHealthString(int)** ». Cette méthode retourne l'état de santé de la batterie. En fonction de la valeur passée en argument, la variable de retour peut contenir l'un des types mentionnés en annexe : « Inconnu », «Morte», « Gelée », «Bonne», « Surchargée », « Surchauffée », ou « Échec ».

4.3 Écrire le code de la méthode « **String getStatusString(int,int)** ». Le premier argument représente l'état de la batterie, le second son niveau (un calcul en %). Cette méthode retourne le niveau de charge de la batterie. En fonction de la valeur passée en argument, la variable de retour peut contenir l'un des types mentionnés en annexe : « Inconnu », «Pas de charge », «Pleine», « Charge en cours xyz% », « Décharge en cours xyz% ». La 2e figure en annexe donne un exemple avec une charge à 50%.

4.4 Écrire le contenu de la méthode « onCreate » de la classe « MainActivity ». Le layout a l'identifiant « **activity_main** », le bouton est « **buttonBatteryStatus »** et le textview est « **textViewBatteryStatus »**. À noter que cette méthode définit aussi tous les éléments nécessaires au bon fonctionnement du bouton ainsi que du BroadcastReceiver. Ce dernier sera déclaré dans la question qui suit (jetez d'abord un coup d'oeil à la question suivante).

4.5 Nous allons utiliser un « BroadcastReceiver » pour surveiller l'état de la batterie. On vous demande de compléter le contenu de la méthode « OnReceive ». Cette méthode va se charger de collecter les différentes informations nécessaires aux méthodes 4.1 à 4.3; d'afficher dans un « textview » l'état de la batterie (le résultat de 4.2), la nature du branchement (le résultat de 4.1), le statut (le résultat de 4.3), la technologie utilisée, la température et le voltage. Une capture d'écran de l'affichage demandée est fournie à titre d'exemple en annexe. Cette collecte ne peut avoir lieu que si la batterie est présente.

```
private BroadcastReceiver batteriereceiver = new BroadcastReceiver() {
   @Override
   public void onReceive(Context context, Intent intent) {









}
```

4.5 Après le lancement de l'application, il n'y a aucun message alarmant dans les logs (Logcat) de l'application. On clique sur le bouton « back », oups, on obtient ce message alarmant :

« 02-15 21:28:37.424 4585-4585/ca.umontreal.iro.ift1155.getbatteryinfoapp E/ActivityThread: Activity ca.umontreal.iro.ift1155.getbatteryinfoapp.MainActivity has leaked IntentReceiver ca.umontreal.iro.ift1155.getbatteryinfoapp.MainActivity$2@ff6fa37 that was originally registered here. Are you missing a call to unregisterReceiver()? »

Expliquer le message. Puis, écrivez le code nécessaire permettant de corriger le problème. Ce code va s'insérer dans la classe « MainActivity »,

4.6 Tout en développant votre réponse, est-ce qu'il faut enregistrer le « BroadcastReceiver » dans le fichier manifeste de l'application?

Annexes

BatteryManager

public class BatteryManager
extends Object

java.lang.Object
↳ android.os.BatteryManager

The BatteryManager class contains strings and constants used for values in the ACTION_BATTERY_CHANGED Intent, and provides a method for querying battery and charging properties.

Instances of this class must be obtained using Context.getSystemService(Class) with the argument BatteryManager.class or Context.getSystemService(String) with the argument Context.BATTERY_SERVICE.

Summary

Constants

String	ACTION_CHARGING Sent when the device's battery has started charging (or has reached full charge and the device is on power).
String	ACTION_DISCHARGING Sent when the device's battery may be discharging, so apps should avoid doing extraneous work that would cause it to discharge faster.
int	BATTERY_HEALTH_COLD
int	BATTERY_HEALTH_DEAD
int	BATTERY_HEALTH_GOOD
int	BATTERY_HEALTH_OVERHEAT
int	BATTERY_HEALTH_OVER_VOLTAGE
int	BATTERY_HEALTH_UNKNOWN
int	BATTERY_HEALTH_UNSPECIFIED_FAILURE
int	BATTERY_PLUGGED_AC Power source is an AC charger.
int	BATTERY_PLUGGED_USB Power source is a USB port.
int	BATTERY_PLUGGED_WIRELESS Power source is wireless.
int	BATTERY_PROPERTY_CAPACITY Remaining battery capacity as an integer percentage of total capacity (with no fractional part).
int	BATTERY_PROPERTY_CHARGE_COUNTER Battery capacity in microampere-hours, as an integer.
int	BATTERY_PROPERTY_CURRENT_AVERAGE Average battery current in microamperes, as an integer.
int	BATTERY_PROPERTY_CURRENT_NOW Instantaneous battery current in microamperes, as an integer.
int	BATTERY_PROPERTY_ENERGY_COUNTER Battery remaining energy in nanowatt-hours, as a long integer.
int	BATTERY_PROPERTY_STATUS Battery charge status, from a BATTERY_STATUS_* value.
int	BATTERY_STATUS_CHARGING
int	BATTERY_STATUS_DISCHARGING
int	BATTERY_STATUS_FULL

int	BATTERY_STATUS_NOT_CHARGING
int	BATTERY_STATUS_UNKNOWN
String	EXTRA_HEALTH Extra for ACTION_BATTERY_CHANGED: integer containing the current health constant.
String	EXTRA_ICON_SMALL Extra for ACTION_BATTERY_CHANGED: integer containing the resource ID of a small status bar icon indicating the current battery state.
String	EXTRA_LEVEL Extra for ACTION_BATTERY_CHANGED: integer field containing the current battery level, from 0 to EXTRA_SCALE.
String	EXTRA_PLUGGED Extra for ACTION_BATTERY_CHANGED: integer indicating whether the device is plugged in to a power source; 0 means it is on battery, other constants are different types of power sources.
String	EXTRA_PRESENT Extra for ACTION_BATTERY_CHANGED: boolean indicating whether a battery is present.
String	EXTRA_SCALE Extra for ACTION_BATTERY_CHANGED: integer containing the maximum battery level.
String	EXTRA_STATUS Extra for ACTION_BATTERY_CHANGED: integer containing the current status constant.
String	EXTRA_TECHNOLOGY Extra for ACTION_BATTERY_CHANGED: String describing the technology of the current battery.
String	EXTRA_TEMPERATURE Extra for ACTION_BATTERY_CHANGED: integer containing the current battery temperature.
String	EXTRA_VOLTAGE Extra for ACTION_BATTERY_CHANGED: integer containing the current battery voltage level.

Captures d'écran d'un émulateur

L'interface obtenue après le lancement de l'application sur un émulateur	Le résultat obtenu après avoir cliqué sur le bouton « CLIQUER POUR DES INFOS SUR LA BATTERIE »