Trimestre Hiver, 2015

Mohamed Lokbani

IFT1155 – Examen Final –

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

Nom : _____| Prénom(s) : _____|

Signature : _____| Login : _____|

Date : Mercredi 15 avril 2015

Durée : 3 heures (de 18h30 à 20h30)

Local : Z-200, 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 (1.1 à 1.10)

2._____/20 (2.1, 2.2, 2.3, 2.4, 2.5)

3._____/20 (3.1, 3.2, 3.3, 3.4, 3.5)

4._____/20 (4.1, 4.2, 4.3, 4.4, 4.5)

5._____/20 (5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7)

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] Pour Andoird, « APK » est l'acronyme de « l'Agence de Protection des Kongourous »?

1.2 [VRAI | FAUX] La méthode « onResume » est appelée quand votre application est passée en avant plan?

1.3 [VRAI | FAUX] Il est nécessaire d'avoir l'autorisation de Google avant de pouvoir utiliser GoogleMaps?

1.4 [VRAI | FAUX] Il n'est pas possible de figer (bloquer) l'orientation d'une application?

1.5 [VRAI | FAUX] Android détermine qui doit répondre à un intente donné?

1.6 [VRAI | FAUX] Il n'est pas possible de sauvegarder des données en cache?

1.7 [VRAI | FAUX] Il n'est pas possible de modifier un objet « SharedPreferences »?

1.8 [VRAI | FAUX] Il est possible de savoir si une base de données est en lecture seule?

1.9 [VRAI | FAUX] Il est plus simple d'utiliser « AsyncTask » que « Thread »?

1.10 [VRAI | FAUX] Effectuer des requêtes HTTP dans le thread UI va soulever une exception?

Exercice 2 (20 points)

2.1 Pourquoi Android est-il si populaire? La question est-elle subjective?

2.2 Quelles sont les informations dont vous allez avoir besoin avant de coder une application Android pour un client donné?

2.3 Quelle est l'utilité de « WebView », pour Android?

2.4 Définissez les différents « Context » disponibles dans Android.

2.5 Quelles sont les boites de dialogues supportées par Android? Décrivez-les brièvement.

Exercice 3 (20 points)

3.1 Une application Android doit accéder aux données de l'appareil, par exemple les messages « SMS » ou la caméra. À quel moment, l'utilisateur d'une telle application, doit-il autoriser un tel accès?

3.2 Quelle est la différence entre « permission » et « uses-permissions » dans Android?

3.3 Est-ce qu'il est conseillé de changer le nom de l'application après son déploiement sur le magasin virtuel?

3.4 Comment résoudre les problèmes de compatibilité quand il faut déployer une application Android pour un téléphone et pour une tablette?

3.5 Comment analyser une application qui plante? Quelles sont les techniques disponibles pour le débogage afin de corriger le plantage de l'application?

Exercice 4 (20 points)

4.1 Comment deux applications Android peuvent-elles partager le même identificateur Linux (id) et fonctionner dans la même machine virtuelle?

4.2 Par défaut, le processus associé à une application a combien de threads? Qui aura la tâche de créer ce nombre de threads?

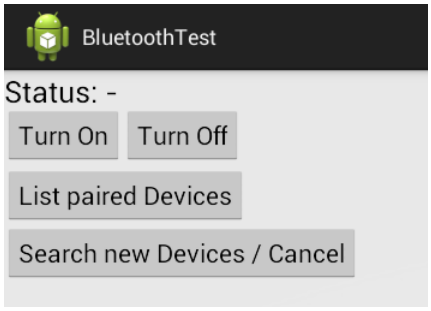
4.3 Quelle est la différence entre un « thread » et un « handler thread »?

4.4 On crée un thread dans l'activité, puis on ferme cette activité. Est-ce que ce thread va continuer à fonctionner en arrière-plan ou est-ce qu'il sera tué automatiquement quand l'activité est fermée? S'il va continuer à fonctionner en arrière-plan, quand sera-t-il tué?

4.5 Comment créer un service ayant un seul thread? Doit-on utiliser un « IntentService » ou un « AsyncTask »?

Exercice 5 (20 points)

Nous allons nous intéresser à l'utilisation du « Bluetooth » dans une application Android. Au départ, l'application est représentée par la figure suivante :



- « Status » : il permet de donner l'état du Bluetooth. Est-il disponible sur l'appareil en question? S'il est disponible, est-il activé ou désactivé?
- « Turn On » : active le Bluetooth (s'il est disponible sur l'appareil).
- « Turn Off » : désactive le Bluetooth (s'il est disponible sur l'appareil).
- « List paired Device » : liste les appareils connectés par Bluetooth.
- « Search new devices / Cancel » : recherche via Bluetooth les appreils disponible. Ce bouton permet aussi d'annuler cette recherche.

Pour la suite de l'exercice :

```
private static final int REQUEST_ENABLE_BT = 1;
private Button onBtn;
private Button offBtn;
private Button listBtn;
private Button findBtn;
private TextView text;
private BluetoothAdapter monBluetoothAdapter;
private Set<BluetoothDevice> pairedDevices;
private ListView myListView;
private ArrayAdapter<String> BTArrayAdapter;

// prend une instance de BluetoothAdapter - Bluetooth radio
monBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
```

5.1 Quelles sont les permissions qu'il faudra ajouter dans le fichier manifeste de l'application pour pouvoir utiliser et manipuler le Bluetooth de l'appareil?

5.2 Si le Bluetooth n'est pas supporté par votre appareil, l'instance « monBluetoothAdapter » prend la valeur « null ». Nous allons mettre dans ce cas à « false » les boutons et nous allons afficher « Status : not supported ». En plus de cela, nous allons afficher un « toast » avec le message « Your device does not support Bluetooth ». Écrivez ce fragment de code.

Pour la suite de l'exercice, nous allons supposer que votre appareil est compatible Bluetooth. Voir la fin du document pour une liste complète des constances et méthodes disponibles dans la classe « BluetoothAdapter ».

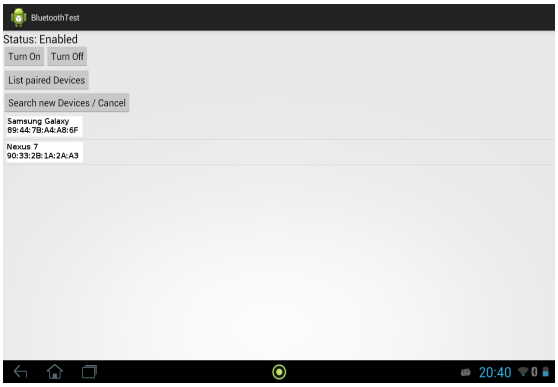
5.3 Placez-vous à la place de l'utilisateur et cliquez sur le bouton « Turn On ». Que se passe-t-il?

5.4 Placez-vous à la place de l'utilisateur et cliquez sur le bouton « Turn Off ». Que se passe-t-il?

5.5 Écrivez le code nécessaire pour le bon fonctionnement du bouton « onBtn ».
Images obtenues après un clic sur le bouton « Turn On » dans 2 cas de figure.

5.6 Écrivez le code nécessaire pour le bon fonctionnement du bouton « offBtn ».

5.7 Écrivez le code nécessaire pour le bon fonctionnement des boutons « findBtn » et « listBtn ».



BluetoothAdapter

Constants		
String	ACTION_CONNECTION_STATE_CHANGED	Intent used to broadcast the change in connection state of the local Bluetooth adapter to a profile of the remote device.
String	ACTION_DISCOVERY_FINISHED	Broadcast Action: The local Bluetooth adapter has finished the device discovery process.
String	ACTION_DISCOVERY_STARTED	Broadcast Action: The local Bluetooth adapter has started the remote device discovery process.
String	ACTION_LOCAL_NAME_CHANGED	Broadcast Action: The local Bluetooth adapter has changed its friendly Bluetooth name.
String	ACTION_REQUEST_DISCOVERABLE	Activity Action: Show a system activity that requests discoverable mode.
String	ACTION_REQUEST_ENABLE	Activity Action: Show a system activity that allows the user to turn on Bluetooth.
String	ACTION_SCAN_MODE_CHANGED	Broadcast Action: Indicates the Bluetooth scan mode of the local Adapter has changed.
String	ACTION_STATE_CHANGED	Broadcast Action: The state of the local Bluetooth adapter has been changed.
int	ERROR	Sentinel error value for this class.
String	EXTRA_CONNECTION_STATE	Extra used by ACTION_CONNECTION_STATE_CHANGED This extra represents the current connection state.
String	EXTRA_DISCOVERABLE_DURATION	Used as an optional int extra field in ACTION_REQUEST_DISCOVERABLE intents to request a specific duration for discoverability in seconds.
String	EXTRA_LOCAL_NAME	Used as a String extra field in ACTION_LOCAL_NAME_CHANGED intents to request the local Bluetooth name.
String	EXTRA_PREVIOUS_CONNECTION_STATE	Extra used by ACTION_CONNECTION_STATE_CHANGED This extra represents the previous connection state.
String	EXTRA_PREVIOUS_SCAN_MODE	Used as an int extra field in ACTION_SCAN_MODE_CHANGED intents to request the previous scan mode.
String	EXTRA_PREVIOUS_STATE	Used as an int extra field in ACTION_STATE_CHANGED intents to request the previous power state.
String	EXTRA_SCAN_MODE	Used as an int extra field in ACTION_SCAN_MODE_CHANGED intents to request the current scan mode.
String	EXTRA_STATE	Used as an int extra field in ACTION_STATE_CHANGED intents to request the current power state.
int	SCAN_MODE_CONNECTABLE	Indicates that inquiry scan is disabled, but page scan is enabled on the local Bluetooth adapter.
int	SCAN_MODE_CONNECTABLE_DISCOVERABLE	Indicates that both inquiry scan and page scan are enabled on the local Bluetooth adapter.
int	SCAN_MODE_NONE	Indicates that both inquiry scan and page scan are disabled on the local Bluetooth adapter.
int	STATE_CONNECTED	The profile is in connected state
int	STATE_CONNECTING	The profile is in connecting state
int	STATE_DISCONNECTED	The profile is in disconnected state
int	STATE_DISCONNECTING	The profile is in disconnecting state
int	STATE_OFF	Indicates the local Bluetooth adapter is off.
int	STATE_ON	Indicates the local Bluetooth adapter is on, and ready for use.

int	STATE_TURNING_OFF	Indicates the local Bluetooth adapter is turning off.
int	STATE_TURNING_ON	Indicates the local Bluetooth adapter is turning on.

Public Methods (quelques méthodes utiles)	
boolean	cancelDiscovery() Cancel the current device discovery process.
static boolean	checkBluetoothAddress(String address) Validate a String Bluetooth address, such as "00:43:A8:23:10:F0" Alphabetic characters must be uppercase to be valid.
void	closeProfileProxy(int profile, BluetoothProfile proxy) Close the connection of the profile proxy to the Service.
boolean	disable() Turn off the local Bluetooth adapter—do not use without explicit user action to turn off Bluetooth.
boolean	enable() Turn on the local Bluetooth adapter—do not use without explicit user action to turn on Bluetooth.
String	getAddress() Returns the hardware address of the local Bluetooth adapter.
BluetoothLeAdvertiser	getBluetoothLeAdvertiser() Returns a BluetoothLeAdvertiser object for Bluetooth LE Advertising operations.
BluetoothLeScanner	getBluetoothLeScanner() Returns a BluetoothLeScanner object for Bluetooth LE scan operations.
Set<BluetoothDevice>	getBondedDevices() Return the set of BluetoothDevice objects that are bonded (paired) to the local adapter.
synchronized static BluetoothAdapter	getDefaultAdapter() Get a handle to the default local Bluetooth adapter.
String	getName() Get the friendly Bluetooth name of the local Bluetooth adapter.
int	getProfileConnectionState(int profile) Get the current connection state of a profile.
boolean	getProfileProxy(Context context, BluetoothProfile.ServiceListener listener, int profile) Get the profile proxy object associated with the profile.
int	getScanMode() Get the current Bluetooth scan mode of the local Bluetooth adapter.
int	getState() Get the current state of the local Bluetooth adapter.
boolean	isDiscovering() Return true if the local Bluetooth adapter is currently in the device discovery process.
boolean	isEnabled() Return true if Bluetooth is currently enabled and ready for use.
boolean	isMultipleAdvertisementSupported() Return true if the multi advertisement is supported by the chipset
boolean	isOffloadedFilteringSupported() Return true if offloaded filters are supported
boolean	isOffloadedScanBatchingSupported() Return true if offloaded scan batching is supported
boolean	setName(String name) Set the friendly Bluetooth name of the local Bluetooth adapter.
boolean	startDiscovery() Start the remote device discovery process.