Trimestre Été, 2013

Mohamed Lokbani

IFT1155 – Examen Final –

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

Nom : _____| Prénom(s) : _____|

Signature : _____| Code perm : _____|

Date : mardi 16 juillet 2013

Durée : 3 heures (de 17h30 à 20h30)

Local : Z-220 ; 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._____/15 (2.1, 2.2, 2.3, 2.4, 2.5)

3._____/15 (3.1, 3.2, 3.3, 3.4, 3.5)

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

5._____/30 (5.1, 5.2, 5.3, 5.4)

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] La plateforme Android utilise la base de données Oracle, pour le développement?

1.2 [VRAI | FAUX] Les applications Android ne peuvent être distribuées que via la plateforme [Google Play]?

1.3 [VRAI | FAUX] Il est possible d'utiliser l'émulateur pour tester des envois SMS?

1.4 [VRAI | FAUX] Webkit est une librairie native de la plateforme Android?

1.5 [VRAI | FAUX] La constante « CONNECTIVITY_ACTION » nous informe si un changement a eu lieu dans la connectivité du réseau?

1.6 [VRAI | FAUX] Les fichiers ressources sont en mode lecture et écriture?

1.7 [VRAI | FAUX] Pour accéder à un provider, votre application doit demander le plus souvent des permissions spécifiques du fichier manifeste?

1.8 [VRAI | FAUX] Une application Android peut demander des permissions additionnelles dynamiquement au moment de son exécution?

1.9 [VRAI | FAUX] « Google Map » fait partie des paquetages par défaut d'Android?

1.10 [VRAI | FAUX] Android supporte les applications « jar »?

Exercice 2 (15 points) Choisissez la bonne réponse en y incluant une courte explication. Il peut y avoir plusieurs bonnes réponses.

2.1 Quelle est la composante qui n'est pas activée par un intent?

a) Activité
b) Service
c) ContentProvider
d) BroadcastReceiver

2.2 Quand un ContentProvider est-il activé?

a) En utilisant un intent
b) En utilisant SQLite
c) En utilisant ContentResolver
d) Aucune des précédentes réponses

2.3 Quelles sont les déclarations qui permettent d'obtenir la valeur « Integer » sauvegardée avec la clé « mDirection » de l'instance « itest » d'un objet « Bundle »?

a) mDirection = itest.getInteger("mDirection");
b) mDirection = Bundle.getInteger("mDirection");
c) mDirection = itest.getLong("mDirection");
d) mDirection = itest.getInteger("uneCle");

2.4 Lesquelles de ces assertions décrivent le processus de signature d'une application?

a) Toutes les applications doivent être signées avant d'être déployées?
b) Pour un développement local et qui ne nécessite pas l'utilisation de « Google Play », un certificat auto-signé fait l'affaire.
c) Il y a 3 modes de signature : « debug », « test » et « release ».
d) Pour distribuer une application de test à plus de 100 clients, il est nécessaire d'avoir un certificat test émis par Google pour notre application.
e) Un certificat « release » est émis par Google moyennant un paiement unique de 25$ lors de l'inscription en tant que développeur.

2.5 Lesquelles de ces assertions décrivent l'émulateur Android SDK?

a) Un émulateur n'exécute pas en réalité le système Android. Il s'agit plutôt d'un composant logiciel qui interprète les fonctionnalités et les actions de toute l'API d'Android.
b) Les applications déployées avec un certificat « debug » sur l'émulateur doivent-être résignées pour être déployées sur « Google Play ».
c) Un seul émulateur peut-être exécuté à la fois.
d) L'adresse IP de l'émulateur est la même que celle de l'ordinateur qui l'héberge.
e) L'émulateur n'a pas besoin d'être redémarré après chaque déploiement d'une application.

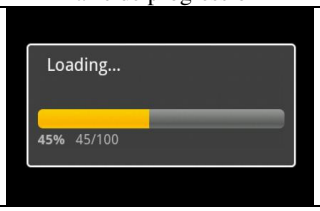
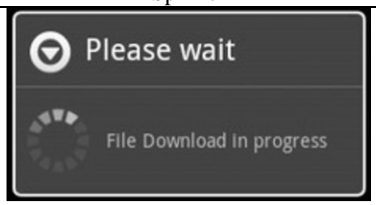
Exercice 3 (15 points) Expliquez ce qui suit.

3.1 Une application qui peut lire et mettre à jour des données fournies à travers un ContentProvider, peut-être mise à jour sans trop de difficultés pour lire et mettre à jour ces mêmes données si elles étaient fournies via une base de données Android SQLite.

3.2 L'API Android fournit un ensemble de classes pour renforcer la notion qu'un seul thread a la possibilité de mettre à jour l'interface graphique (GUI) de l'application.

3.3 Un « AlertDialog » est plus approprié qu'un « Toast » pour informer l'utilisateur que le téléphone va s'éteindre dans 5 minutes, car la batterie est presque vide.

3.4 Pour informer l'utilisateur qui vient de télécharger une grosse image localement sur son appareil Android à partir d'un fichier, il est préférable d'utiliser un « ProgressDialog » du type une barre de progression qu'un « ProgressDialog » du type « spinner ».

Barre de progression	Spinner
	

3.5 Expliquez la relation entre un ContentProvider et un ContentResolver.

Exercice 4 (20 points) Le but de l'exercice est de sauvegarder l'état de la connexion d'un utilisateur à un service donné, et la date de cette connexion. Pour cela, nous allons utiliser deux activités.

Voici une capture d'écran après avoir exécuté successivement les deux activités :

Activité « Login », principale	Activité « Test », secondaire	Test après un clic sur « Menu »

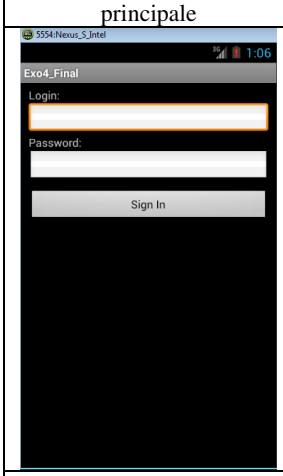
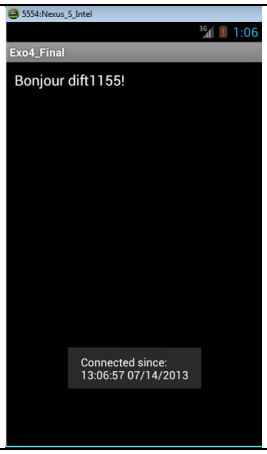
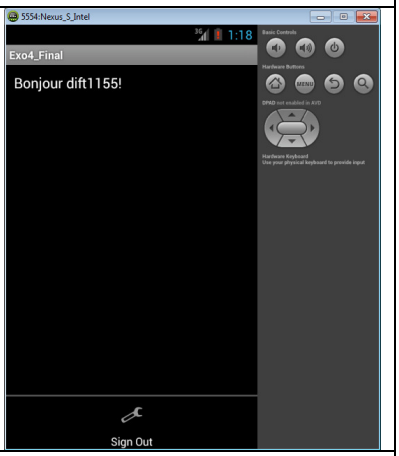

		

Figure -1-	Figure -2-	Figure -3-

Les opérations se déroulent comme suit :

1- Démarrage de l'activité principale, « Login ».
2- L'utilisateur « dift1155 » est-il déjà connecté à sa session?
   a) Non, l'utilisateur ne s'est pas encore connecté. Dans ce cas, nous lui affichons la figure -1-.
      1) L'utilisateur fournit son [Login], « dift1155 », et son [Password], « test »; puis il clique sur le bouton [Login] qui correspond au [Sign in] sur la figure -1-.
      2) Nous préservons l'information comme quoi l'utilisateur s'est connecté à sa session.
      3) Nous sauvegardons aussi la date et l'heure de sa connexion.
      4) Nous affichons la figure -2- : le texte « Bonjour dift1155! » et un « Toast » contenant la date de sa connexion pour cet exemple « 13:06:57 07/14/2013 ».
   b) Oui, l'utilisateur s'est connecté déjà. Nous lui affichons la figure -2-, comme dans « 2.a.4 ».
3- L'utilisateur clique sur l'icône « Maison » (Home), puis décide de relancer son application. Comme il est déjà connecté, il tombe sur le cas « 2.a.b ». Si pour une quelconque raison, Android a décidé de tuer l'application, retour à la case départ, et nous sommes dans le cas « 2.a ».
4- L'utilisateur clique sur « Menu » et choisit de se déconnecter en cliquant sur « Sign out ». On s'assure dans ce cas que la session a bien pris fin et que la date et l'heure ont été remises à zéro.

4.1 Complétez la méthode « onCreate » de la classe « Test », associée à la seconde activité :

```
public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.test);









}
```

4.2 La méthode « onOptionsItemSelected » de la classe « Test », est appelée si l'utilisateur clique sur le bouton « Logout » correspondant au « Sign out » sur la figure -3-.

```
public boolean onOptionsItemSelected(MenuItem item){









}
```

4.3 Complétez la méthode « onCreate » de la classe « Login », associée à la première activité :

```
public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.test);









}
```

Exercice 5 (30 points) Soit « Exo05 », une application Android, composée de 2 boutons et d'un « TextView ». Les boutons sont : « IP » et « TimeStamp ». Le « TextView » affiche, après un traitement donné, le résultat du clic sur l'un des 2 boutons.

Le traitement consiste à faire une requête « http » à un site web donné en faisant appel à la méthode « getJSONFromUrl » fournie en annexe de ce document. Le résultat obtenu est sous la forme « JSON ». Il sera traité par la suite par l'application afin de l'afficher à l'utilisateur via le « TextView ».

Chaque bouton est associé une adresse « url » propre à lui et qui renvoie un résultat « JSON » personnalisé. Ainsi donc, pour les 2 boutons décrits précédemment, nous avons ce qui suit :

Nom du bouton : **IP**
Adresse URL : http://ip.jsontest.com/
Résultat « JSON » :

```
{"ip": "132.204.26.153"}
```

Nom du bouton : **TimeStamp**
Adresse URL : http://date.jsontest.com/
Résultat « JSON »:

```
{
  "time": "03:53:25 AM",
  "milliseconds_since_epoch": 1362196405309,
  "date": "03-02-2013"
}
```

5.1 Quelles sont les permissions qu'il faudra ajouter au fichier « Android Manifest » pour que l'application « Exo05 » puisse fonctionner correctement?

5.2 Est-ce que les réponses ont le même format « JSON ». Précisez lequel (ou lesquels, si le format n'est pas unique).

5.3 Est-ce que le traitement associé à la requête « http » et le traitement de la réponse doivent avoir lieu en dehors du thread UI?

5.4 Nous faisons les hypothèses suivantes :

- Tous les paquetages nécessaires au bon fonctionnement de l'application ont été importés dans votre application;
- Nous sommes intéressés que par la méthode « onCreate » et les méthodes ou classes à ajouter dans l'application pour son bon fonctionnement. Le traitement relatif aux différentes étapes du cycle de vie de l'application, ne nous intéresse pas à ce stade. Il n'est donc pas nécessaire de définir « onDestroy », « onPause » etc.
- Pour les réponses « JSON » obtenues, nous ne sommes intéressés qu'aux champs suivants : « IP », « time », « date ». Il ne faudra pas donc tenir compte du champ « milliseconds_since_epoch ».
- La méthode AsyncTask n'est pas permise.

- **Il sera tenu compte lors de la correction de la clarté et de la compacité de votre programme.**

Annexe

```
public class JSONParser {

	static InputStream is = null;
	static JSONArray jarray = null;
	static String json = "";

	// constructeur
	public JSONParser() {

	}

	public StringBuilder getJSONFromUrl(String url) {

		StringBuilder builder = new StringBuilder();
		HttpClient client = new DefaultHttpClient();
		HttpGet httpGet = new HttpGet(url);
		try {
			HttpResponse response = client.execute(httpGet);
			StatusLine statusLine = response.getStatusLine();
			int statusCode = statusLine.getStatusCode();
			if (statusCode == 200) {
				HttpEntity entity = response.getEntity();
				InputStream content = entity.getContent();
				BufferedReader reader = new BufferedReader(new
                                                    InputStreamReader(content));
				String line;
				while ((line = reader.readLine()) != null) {
					builder.append(line);
				}
			} else {
				Log.e("==>", "Echec lors du telechargement du fichier");
			}
		} catch (ClientProtocolException e) {
			e.printStackTrace();
		} catch (IOException e) {
			e.printStackTrace();
		}

		// retourne un JSON String (un StringBuilder)
		return builder;

	}
}
```