Université Paris Diderot - Paris 7
Master 1 Informatique, Programmation de composants mobiles
10 janvier 2018, Groupe 1 (matin), durée : 2h30

Documents et matériel autorisés :

- tous les documents papier sauf livres,
- documentation android `https://developer.android.com/index.html`,
- documentation java `https://docs.oracle.com/javase/7/docs/api/index.html`
- tous les documents sur moodle du cours,
- vos propres programmes de TP et votre projet,
- vos appareils android pour tester vos programmes.

Il est strictement **interdit** d'utiliser d'autres adresse WWW que mentionnées dessus, en particulier :

- il est interdit d'utiliser google ou n'importe quel autre moteur de recherche (vous devez aller directement sur la page de la documentation d'android et faire les recherches dans la documentation depuis le site d'android , sans passer par google),
- d'utiliser les forums de développeurs,
- utiliser l'ordinateur, tablettes, smartphones à d'autres fins que le développement et le test de vos programmes.

Le sujet comporte 5 pages.

Consignes

Pour chaque exercice vous devez compresser le répertoire **VotreAppli/app/src/main** qui contient les sources de votre application :

```
tar cvf VotreNom_VotrePrenom_Exoi.tar  VotreAppli/app/src/main
gzip VotreNom_VotrePrenomExoi.tar
```

où i sera remplacé par le numéro de l'exercice et `VotreNom` par votre nom.
Déposez le fichier tar.gz qui contient la solution de l'exercice sur moodle (cours Android) dans le dépôt dans la section Examen. Vous pouvez ajouter un fichier txt pour expliquer quelles questions sont implémentées.
J'accepte aussi les archives zip. **Par d'archives rar.** Il est prudent de sauvegarder au moins la dernière version qui marche avant de passer à la question suivante, sinon vous risquez de vous retrouver avec un programme où rien ne marche plus.

Avant de commencer les exercices récupérez sur moodle (section Examen) les fichiers layout pour toutes les activités et le fichier *.java.
Pour les fichier layout récupérés sur moodle il vous faut adapter au moins la ligne `tools:context` pour qu'elle soit conforme à votre projet. Vous pouvez modifier à votre guise tous les fichiers qu'on vous donne (ou même fabriquer vos propres layout).

Dans les fichiers `java` fournis pour le dernier exercice (la bases de données et ContentProvider) vous pouvez faire vos modifications tant que cela ne change pas la structure de la table. (Mais à priori je ne vois aucune raison de modifier les fichiers `*.java` fournis sur moodle.)

Vos programmes seront testés avec un logiciel de détection de plagiat.
Si vous travaillez sur les machines de l'ufr c'est à vous de protéger votre répertoire de travail contre les visites non autorisées.

Barème indicatif : l'exo 1 : 2-3 points, les exos 2 et 3 : 8-9 points par exo.

Exercice 1 Vous devez implémenter une application composée de deux activités, l'activité principal A et l'activité B.
L'activité A (fichier layout `activity_list`) est composée d'une `ListView`. Vous devez alimenter le `ListView` avec de Strings de votre choix (au moins 6 Strings différents). On préfère que le contenu du tableau de String qui servira pour alimenter le ListView soit défini dans un fichier xml de ressources.
L'utilisateur clique sur un item de ListView. L'activité envoie A envoie le String sélectionné par l'utilisateur vers l'activité B.
L'activité B (fichier layout `activity_afficher.xml` possède un TextView dans lequel il faut mettre le String envoyé par l'activité A.
Indication. `OnItemClickListener`, le denier paramètre de la méthode donne l'indice de l'élément sélectionné par l'utilisateur.

Exercice 2 [Devinette] Dans cet exercice vous développerez une application possédant une seule activité dont le fichier layout est `activity_portes.xml`. En regardant le fichier layout vous constaterez que cette activité est composée d'un `TextView` contenant un String fixe `"choisissez une porte"`, de trois boutons au milieu, d'un autre TextView (`id=gagner`) qui initialement est vide (donc invisible) et d'un bouton `rejouer` en bas, voir l'image.
Les trois bouton au milieu représentent trois portes. On suppose que derrière une porte il y a une voiture de luxe (une Bentley) et derrière les deux autres portes une chèvre. L'utilisateur ouvre une porte en cliquant sur un bouton et découvre son récompense. La récompense s'affiche sur le bouton et également dans le TextView au-dessous des portes.

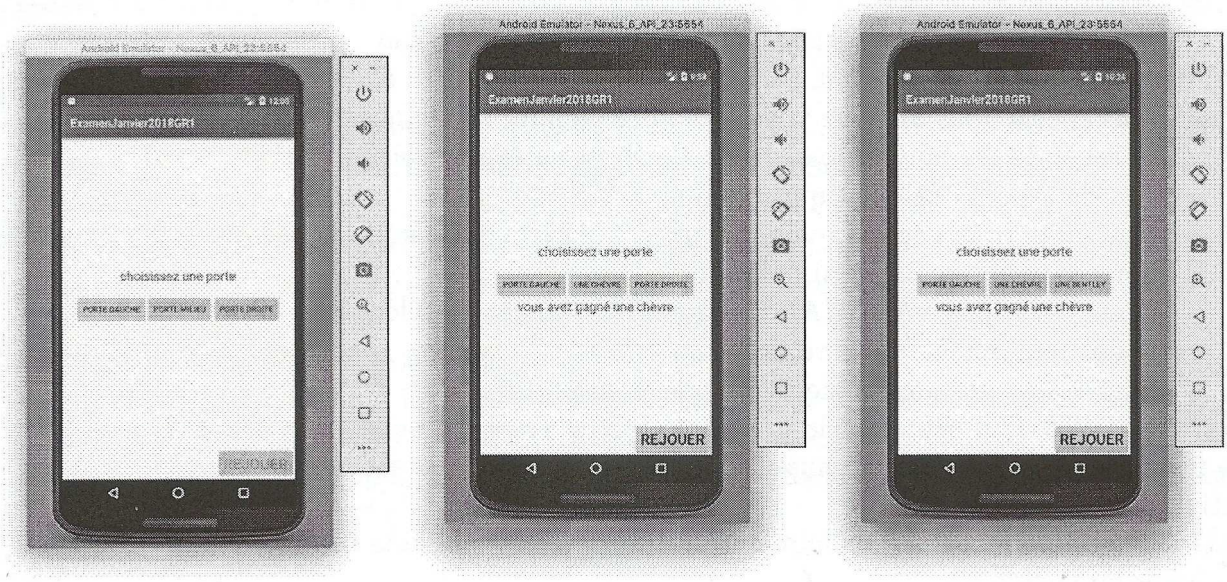
Question 1: Implémentez ce comportement.
Indication Il est commode et recommandé de mettre les trois boutons dans un tableau `Button tabPortes[3]`.
L'activité choisit la porte pour y mettre la Bentlay en tirant un nombre aléatoire parmi 0,1,2 (l'indice de `tabPortes`. Pour trouver un nombre aléatoire on initialise un objet de la classe `java.util.Random`

```
Random random = new Random();
```

et on obtient l'indice de la porte grâce au

```
int indice_porte = random.nextInt(3);
```

Chaque appel à `random.nextInt(3)` fournira un nouveau nombre aléatoire.



Question 2: Après avoir ouvert la première porte l'utilisateur peut cliquer sur d'autres portes pour les ouvrir et découvrir les prix cachés. Mais la récompense gagnée, affichée au-dessous de portes, ne doit pas changer, c'est toujours la récompense qui était derrière la première porte ouverte.
Implémenter ce comportement.

Question 3: Sans doute quand l'utilisateur tourne l'appareil alors le jeu s'initialise (les portes se referment, l'affichage de la récompense disparaît et on rejoue la place de la Bentley).
On veut que après le changement de l'orientation de l'appareil la configuration soit au moins partiellement restaurée :

— que l'affichage de la récompense au-dessous de la porte ne change pas,
— et que la position de la Bentley ne change pas.

Par contre, pour simplifier, on permet que les portes se referment. Donc le changement de l'orientation doit préserver la position de la Bentlay et le texte affiché au-dessous de portes. Implémenter ce comportement.

Question 4: Le bouton `REJOUER` doit permettre de rejouer le jeu : les portes se referment, l'affichage du prix gagné disparaît, l'activité trouve un nouveau emplacement pour la Bentley, grâce au nouvel appel

```
indice_porte = random.nextInt(3);
```

Exercice 3 [Annuaire (base de données)] Dans cet exercice on construit un simple annuaire téléphonique. Cet annuaire est implémenté avec une base de données très rudimentaire contenant une seule table. On vous fournit deux fichier `*.java` :

- `Annuaire.java` contient la classe `Annuaire` dérivée de `SQLiteOpenHelper` qui construit la base de données,
- `AnnuaireProvider` contient le `ContentProvider` associé.

Les deux fichiers ne nécessitent aucune modification (sauf le changement de la première ligne) mais si cela vous arrange vous êtes libre de faire de changement.
L'unique table de l'annuaire possède les attributs suivants (voir `Annuaire.java`) :

- `_id` de type INTEGER qui sert du clé,
- `nom`, `prenom` et `pseudo` qui donnent respectivement le nom, le prénom et le pseudonyme d'une personne qui figure dans l'annuaire,
- `numero` de type VARCHAR qui donne le numéro de téléphone,
- `note` qui permet de faire une annotation quelconque (par exemple noter que le numéro est un numéro de téléphone de domicile).

Votre tâche est d'implémenter la première activité associé à cette base de données.
Le fichier `activity_ajouter_numero.xml` fourni sur moodle est le fichier layout de cette activité.
Cette activité permet à la fois faire entrer un nouveau item dans l'annuaire et de voir tous les items de l'annuaire.
L'activité est composé de (voir le fichier layout) :

- d'un `Spinner`, initialement vide,
- de cinq `EditText` correspondant aux attributs de la table `annuaire` (sauf `_id` qui est géré automatiquement par la BD),
- et d'un bouton.

Question 1: L'utilisateur remplit les `EditText` avec les données d'une nouvelle ligne de la table et appuie sur le bouton. L'activité ajoute la nouvelle ligne dans la table.
Si le numéro de téléphone manque alors l'insertion est annulée et l'activité affiche un `Toast` avec l'information.
Si tous les trois champs : nom, prénom, pseudo sont vides l'insertion est annulée et l'activité affiche un `Toast` avec l'information (il faut qu'au moins un de ces trois champs soit non vide).
Si la valeur d'un `EditText` est vide alors la valeur à insérer doit être NULL (et non pas le String vide `""`).
Après l'insertion de la nouvelle ligne dans la table tous les `EditTexts` seront vidés.
Implémenter ce comportement en utilisant le `ContentResolver`.
Indication. Ne pas oubliez de déclarer le ContentProvider `AnnuaireProvider` dans `AndroidManifest.`

Question 2: Faites en sorte que le `Spinner` de l'activité affiche tous les items de la table `annuaire`. Plus exactement pour chaque ligne de la table le `Spinner` affichera les valeurs de trois attributs : `nom`, `prenom` et `pseudo`.

Utilisez directement `ContentResolver` pour interroger la base de donnée. (Vous n'êtes pas obligé d'utiliser `CursorLoader` mais si vous le faites ce sera un plus).
Le fichier `spinner_item.xml` est le fichier layout pour l'adapter `SimpleCursorAdapter` à utiliser avec le `Spinner`.
Quand l'utilisateur entre une nouvelle ligne dans la table le contenu de `Spinner` doit être actualisé pour refléter le nouveau état de la table.

Question 3: Parfois nous voulons enregistrer un nouveau numéro de téléphone pour une personne déjà dans l'annuaire. Dans ce cas il serait commode que les données de cette personne soit automatiquement pre-affichées dans les trois `EditText`.
On demande que, quand l'utilisateur clique sur un élément affiché dans le `Spinner` alors le nom, prénom et pseudo de la personne sélectionné soit automatiquement dans les trois premiers `EditTexts`.
Indication. Dans `OnItemSelectedListener` le paramètre `View` de l'unique méthode du listener est rien d'autre que le `View` sur lequel l'utilisateur clique. Ce `Vue` est décrit par le fichier `spinner_layout.xml` donc pour récupérer la valeur d'un de trois `TextVue` d'un élément du Spinner on appliquera `findVueById(R.id.identifiant)` où l'identifiant est l'id du `TextVue` correspondant.