Travaux Pratiques

Tous les TPs ci-dessous seront à rendre et seront notés. Chaque TP sera archivé séparément, et contiendra l'ensemble des fichiers du projet. Ces archives seront à envoyer ensemble à mf+m4206c@dagrut.info après la dernière séance de ce module.

Les points seront attribués :

- En fonction de l'avancement global dans les TPs
- Si les consignes sont respectées

Plusieurs points seront enlevés si :

- L'indentation n'est pas correcte et uniforme
- Les conventions de nommage fournies dans le cours ne sont pas respectées
- Des copiers-collers sont constatés, principalement entre étudiants

Les améliorations possibles pour chaque application ne seront pas notées.

Note : lorsque vous archiverez un projet Android Studio, pensez aussi à faire `Build > Clean project` avant de l'archiver, afin de supprimer un maximum de fichiers temporaires et ainsi d'avoir une archive plus petite.

TP 1 : Plus ou Moins

Créez une application comportant trois éléments visuellement :

- Une zone de texte (`TextView`)
- Un champ de text (`EditText`)
- Un bouton

Lorsque l'application démarre, un nombre aléatoire entre 1 et 100 est généré. Le but de l'utilisateur est de trouver ce nombre avec le moins d'essais possibles. Le champ de texte sert à entrer ce nombre, et le bouton à valider sa réponse.

Le `TextView` affichera alors si le nombre à trouver est plus grand ou plus petit que le dernier nombre entré.

Afin de générer un nombre aléatoire entre 1 et 10 par exemple, vous pourrez utiliser ce code :

```
Random rand = new Random();
int n = rand.nextInt(10) + 1;
```

Améliorations possibles :

- Ajoutez un menu pour afficher la solution
- Ajoutez un menu pour réinitialiser la valeur à trouver
- Ajoutez un mode deux joueurs, afin d'avoir un joueur qui choisit le nombre à trouver et un second qui le cherchera.

TP 2 : Pendu

Créez une application permettant de jouer au jeu du pendu. L'interface comprendra au moins trois éléments :

- Une zone de texte comportant les caractères devinés, ou un underscore (_) pour ceux qui ne l'ont pas encore été trouvés.
- Une image représentant un personnage à pendre/pendu (`ImageView`).
- Une zone de texte permettant de proposer un caractère.
- Un bouton permettant de valider ce caractère.

Vous ferez en sorte de ne pas permettre à l'utilisateur qu'il puisse entrer plusieurs caractères à la fois.

Au démarrage de l'application, vous sélectionnerez un mot parmi une liste fixe, par exemple :

```
Lys
Voie
Rue
Sac
Savate
Panneau
Chat
Lune
Canard
Turquoise
Trouble
```

Améliorations possibles :

- Ajoutez un menu pour afficher le mot à trouver.
- Ajoutez un menu pour réinitialiser le mot à trouver.
- Ajoutez un mode deux joueurs, afin d'avoir un joueur qui choisit le mot à trouver et un second qui le cherchera.

TP 3 : Géolocalisation

Pour cette application, vous créerez une application permettant de suivre les coordonnées GPS d'un téléphone, en les affichant à l'écran.

Vous implémenterez l'interface `LocationListener` sur votre `Activity`, récupèrerez une instance de LocationManager via :

```
location_manager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
```

puis demanderez des mises à jour sur vos coordonnées GPS via :

```
location_manager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
3000, 10, this);
```

Vous afficherez dans un ou plusieurs `TextView` à l'écran les valeurs de la latitude, longitude, vitesse et altitude de votre téléphone récupérées de l'instance Location dans votre méthode `onLocationChanged`.

Améliorations possibles :

- Affichez les points récupérés sur une carte (Si vous utilisez l'API google maps, vous devrez créer un compte Google et demander un token pour avoir accès à cette API).

TP 4 : Webview

Pour cette application, vous créerez une interface comportant trois éléments :

- Une `WebView`
- Un champ de text (`EditText`)
- Un bouton

Lorsque le bouton est pressé, vous lancerez le téléchargement de l'URL fournie dans le champ de texte, et vous afficherez la page HTML résultant de ce téléchargement dans votre `WebView`. Le téléchargement de la page devra être géré par votre code et non par la `WebView`! En d'autres termes, vous devrez recoder cette partie de la webview pour permettre cela. Vous pourrez utiliser vos codes faits en TD pour vous aider.

Une fois cela fait, vous intercepterez les chargements de liens HTTP (provoqués par des clics sur des liens par exemple) dans cette webview pour utiliser votre code au lieu de laisser la webview charger la page. Faites par ailleurs attention à ne pas lancer deux fois un même téléchargement !

Vous pourrez vous aider :

- Du code produit dans le TD 3 pour le client HTTP
- Du fichier NetHandler.java fourni avec ce sujet de TP.
- De l'exemple NetworkConnect (Écran principal d'Android Studio > Import an Android code sample > NetworkConnect).

TP 5 : TO-DO List

Pour cette application, vous créerez une interface deux activités :

La première comportera une ListView (gérée au niveau du code via un ArrayAdapter).

La seconde sera un formulaire permettant l'ajout d'une tâche. Cette tâche pourra comporter 3 éléments :

- Une priorité (faible, moyenne, haute)
- Un titre
- Une description détaillée

Vous utiliserez les moyens de communication entre activités standard, comme les Intent.

Améliorations possibles :

- Ajoutez une date de création et une date de modification (gérées en interne).
- Ajoutez un menu permettant de trier les todos par priorité ou par date.
- Au lieu de stocker les données en mémoires, utilisez une base de donnée SQLite.

TP 6 : Chat

Pour cette application vous créerez un client de chat, permettant de discuter avec les autres personnes connectées.

L'application disposera au moins d'un moyen d'afficher les messages (Un `TextView` sera plus simple pour commencer, mais vous devrez utiliser un `ListView` lié à un `ArrayAdapter`). Chaque message envoyé sera de la forme `"Pseudo : Message"` suivi d'un retour à la ligne.

L'application contiendra aussi un bouton dont le texte sera le pseudonyme de l'utilisateur, et pour lequel un clic sur celui-ci demandera un pseudonyme à l'utilisateur. Vous pourrez vous inspirer de ce code pour cela :

```
        AlertDialog.Builder alert = new AlertDialog.Builder(this);

        alert.setTitle("Title");
        alert.setMessage("Message");

        // Set an EditText view to get user input
        final EditText input = new EditText(this);
        alert.setView(input);

        alert.setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int
whichButton) {
                String value = input.getText().toString();
                // Utilisez la valeur ici
            }
        });

        alert.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int
whichButton) {
                // Canceled.
            }
        });

        alert.show();
```

Les messages seront envoyés via la pression de la touche "retour à la ligne" du clavier Android (qui

peut être récupérée via la méthode `setOnKeyListener` d'un `TextEdit`.

Afin de vous aider dans cette tâche, un serveur HTTP codé en java (non-android!) est à votre disposition dans l'archive jointe à ce TP (Sous le nom `ChatServer`). Une classe permettant d'échanger les messages avec le serveur est aussi présente (Sous le nom `NetHandler`), à utiliser dans votre application Android.

Il vous est fortement conseillé de lire et de comprendre le code de la classe `NetHandler`.

Améliorations possibles :

- Ajoutez un horodatage aux messages (Local à chaque téléphone, aucune date ne sera transmise en réseau!)
- Ajoutez une couleur choisie aléatoirement pour chaque pseudonyme (La couleur devra rester la même pour un même pseudonyme).
- Créez un protocole de communication permettant de lister les utilisateurs connectés. Dans ce cas, chaque information transmise par un client pourrait être de la forme :
  - `MSG pseudo message` : Ou `pseudo` est le pseudonyme de l'utilisateur et `message` son message.
  - `LST` : Commande permettant de lister les utilisateurs (chaque utilisateur présent répondra alors avec le message suivant).
  - `IAM pseudo` : message permettant d'indiquer que vous êtes connectés en tant que `pseudo`.
- Vous noterez que ce protocole est peu optimisé (Si un utilisateur se déconnecte, cela se fait toujours de manière invisible pour les autres), mais convient parfaitement à notre exemple.
- En utilisant ce protocole, ajoutez un mode Whiteboard (tableau blanc) basique permettant d’éditer une image (une view dans notre cas) à plusieurs. Il vous faudra stocker l’état du tableau pour le transmettre aux nouveaux arrivants.