ANDROID - TP1

Source et pdf de ce TP :
https://bitbucket.org/olevitt/technologies-mobiles

Ce TP a deux objectifs :
- Se familiariser avec le SDK android et le plugin ADT pour eclipse
- Construire une petite application de simulateur de crédit

1 Mise en place

1.1 Verifier l'installation

- Lancer Eclipse
- Vérifier dans **Window/Preferences** la présence du menu android
- Vérifier que le chemin du SDK est bien rempli

1. Quels sont les niveaux d'API disponibles ?

1.2 Hello world

- Créer un nouveau projet android via **File/New/Other/Android/Android Project**
- Executer l'application (ctrl + F11)

2. Pourquoi l'application ne se lance t'elle pas ?

- L'android virtual device manager s'ouvre et permet de créer un émulateur
- L'AVD manager est aussi disponible via **Window / Android virtual device manager**

1.3 Créer un appareil virtuel

- Créer un nouvel appareil en API 17 (4.2)
- Démarrer l'appareil
- Manipuler un peu l'émulateur
- Lancer l'application sur l'émulateur (ctrl + F11)

3. Quelles sont les applications pré-installées sur l'émulateur ?
4. Comment voir la log de l'émulateur ?

2 Construire notre simulateur de crédit

2.1 Un peu d'UI

Modifier le layout main pour y ajouter quelques composants :
- Un champ de texte (EditText) : montant
- Un champ de texte (EditText) : durée
- Ajouter un bouton (Button) : calculer
- Obliger l'utilisateur à ne rentrer que des chiffres dans les champs en utilisant la propriété InputType

5. Que faut t'il prévoir pour pouvoir utiliser ces views en java ?

2.2 Rendre l'interface vivante

- Ecouter les clicks sur le bouton et déclencher un toast lors d'un click sur le bouton
- Rappel : 2 techniques ont été vues en cours pour écouter les actions
- Dans le toast, afficher le contenu du champ de texte (utiliser la méthode getText() de EditText)

2.3 Implémenter la logique métier

- Lors d'un click sur le bouton, récupérer les valeurs de montant et durée
- Aide : Integer.parseInt(String) permet de convertir une chaîne en entier
- Implémenter la logique métier en utilisant le code ci-dessous
- Afficher ce résultat dans un toast
- Afficher ce résultat dans une view (TextView)

```
1 private double calculMensualites(
2     double montant, double taux,double nbmensualites) {
3         return (k * (t/12)) / (1-Math.pow((1+t/12), -n));
4 }
```

6. Comment cacher la view contenant les mensualités tant qu'elles n'ont pas été calculées ?
7. Que proposez vous si l'utilisateur n'a pas rempli les 2 champs ?

3 Aller plus loin

3.1 Un peu de style

- Changer le nom de l'application
- Modifier l'icône pour le lanceur d'applications
- Modifier le layout pour organiser les différents éléments de façon "user-friendly"

3.2 Traduire l'application

- S'assurer que toutes les strings ont été définis en XML
- Créer un dossier values-fr et y copier le fichier strings.xml
- Traduire le contenu de strings.xml
- Tester sur émulateur en changeant la langue dans le menu

Les parties ci-dessous contiennent des élements non (encore) vus en cours.

3.3 Partager le résultat

- Ajouter un bouton : partager
- Mettre une icône de partage à ce bouton (setBackgroundResource)
- Lors d'un click sur le bouton, lancer un intent implicite pour partager le résultat

8. Comment filtrer les activities proposées pour le partage ?
9. Que se passe t'il si aucune activity n'est capable de répondre à l'intent ?

3.4 Créer un historique des calculs

10. Quelle solution de stockage de l'historique des calculs vous parait la mieux adaptée ? (Rappel : on a le choix entre préferences, BDD SQLite et fichier plat)

- Mettre en place la solution choisie
- A chaque calcul, rajouter un élément à l'historique

3.5 Afficher l'historique

- L'historique sera affiché dans un écran séparé
- Créer une nouvelle Activity qui contiendra l'historique (penser à la déclarer dans le Manifest)

11. Quel ViewGroup vous paraît le plus adapté pour afficher l'historique (nombre indéfini d'éléments) ?
12. Comment lancer cette activité ?