TP ANDROID

LES BASES

Préambule

Matériel utilisé : tablette acer iconia A1 810 resolution 768x1024 (mdpi)
Logiciels : Bundle Eclipse (pour le designer choisir 7'' WSVGA par exemple) et Genymotion (modifier la résolution : 768x1024 – 160 mdpi

Premier projet

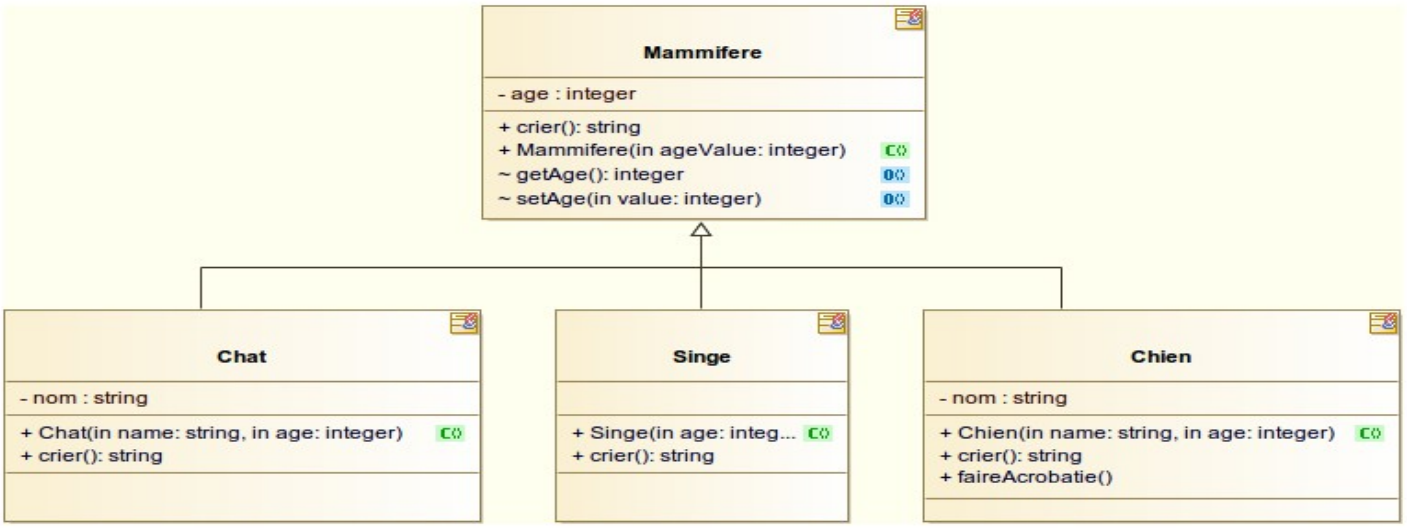
- A l'aide de l'assistant créez un projet dans votre répertoire de travail. Rajoutez un bouton et un Textview avec l'outil Graphical layout. Testez avec GenyMotion. Faire une rotation de l'écran.
- Rajouter une balise android:id dans le relativeLayout proposé par défaut et donnez un identifiant.Dans votre code Java modifier le texte de votre TextView. Testez.Créez un bouton de manière dynamique et rajoutez le à la vue (code Java). Testez avec GenyMotion.
- Créer un dossier layout-land dans le répertoire res. Copiez le layout précédent dans le dossier. Modifier le layout en rajoutant des éléments graphiques de votre choix. Testez avec GenyMotion. Faire une rotation de l'écran.

Méthode call-back d'une activité

- Créer un nouveau projet. Rajoutez une check-box. Chercher la liste des méthodes de callback sur le web : site android developper. Sur le principe du transparent de cours, rajoutez les méthodes à votre activité. Chaque méthode de call-back incluera une trace de type Log.i(....).
  Lancez l'activité avec l'**émulateur inclus en Eclipse**. Ouvrez la perspective DDMS. Examiner votre LogCat.
- Ouvrez un terminal de commande. Connectez vous en telnet à l'émulateur :
  telnet localhost 5554
  Simulez un appel entrant .Tapez help pour avoir la liste des commandes : exemple de simulation d'appel :
  gsm call 060505050505
  Observez le cycle d'activité
- Cliquez sur la touche de l'émulateur back. Observez le cycle d'activité. Cliquez sur la touche de l'émulateur home. Observez le cycle d'activité.

Concepts objets

Soit la hiérarchie de classes présentées en cours:



- Créez un projet Java. Créez un premier fichier décrivant la classe Mammifère (clic droit souris > New Class). Recopiez le code du transparent de cours.
  Créez un deuxième fichier (même procédure mais validez la case static main). Créer un objet de type mammifère. Faites le crier. Afficher son age.
- Essayer de modifier son age sans passer par les getters/setter (accès direct au champ age depuis l'extérieur).Changer l'âge. Afficher l'âge pour voir si le champ a été modifié.

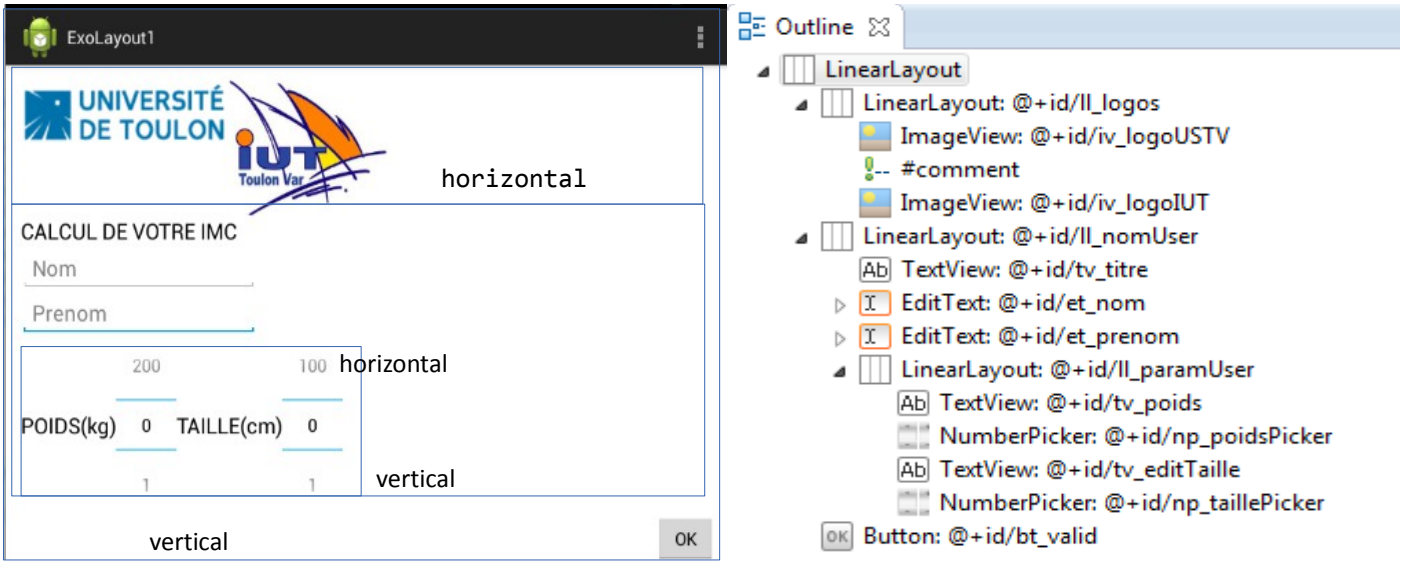
- Rajouter la classe Chien. Créer un chien. Afficher son âge. Faire une acrobatie. Peut-on afficher son nom. Apporter une modification dans la classe pour cela (public interdit!!!).
- Rajouter des System.out.println dans les constructeurs de Mammifere et de Chien. Relancez le programme et regarder l'ordre d'appel des constructeurs.
- Créer une référence monChien de type Chien (pas de new). Créer une référence monMammifere de type Mammifere (pas de new).
  Créer un objet de type Chien référencé par monMammifere (une ligne d'instruction). Essayer de faire des acrobaties ! Essayer de changer l'âge. Que s'est-il passé ? Quel type d'opération a t'on fait ?
  Ecrire la ligne suivante : `monChien=monMammifere;` Que vous indique le compilateur ? Comment s'appelle cette opération ? L'objet réel a t'il changé entre les 2 opérations ?

- Rajouter la classe Chat. Créer un objet Chien référencé par monChien. Créer une référence monChat et une référence monMammifere.
  Tester `monMammifere=monChien; . Méthodes accessibles ?`
  `Tester monChat=monChien; . Méthodes accessibles ?`
  Créer un Chat référencé par mon Chat. Tester `monMammifere=monChat. Méthodes accessibles ?`

- Reprendre l'exemple du cours : créer un tableau de mammifère. Le peupler d'animaux. Appliquer un traitement commun à tous ces animaux : les faire crier par exemple !

Les linear layouts

L'objectif est la réalisation de l'interface graphique suivant permettant la saisie des données pour le calcul de l'indice de masse corporelle.



- Ouvrez le projet Exolayout1.Suivez la structure imposée ainsi que la désignation des éléments placés. Les textes affichés sont codés 'en dur' dans le xml. Dans votre code java vous rajouterez du code pour la configuration des NumberPicker ressemblant à :

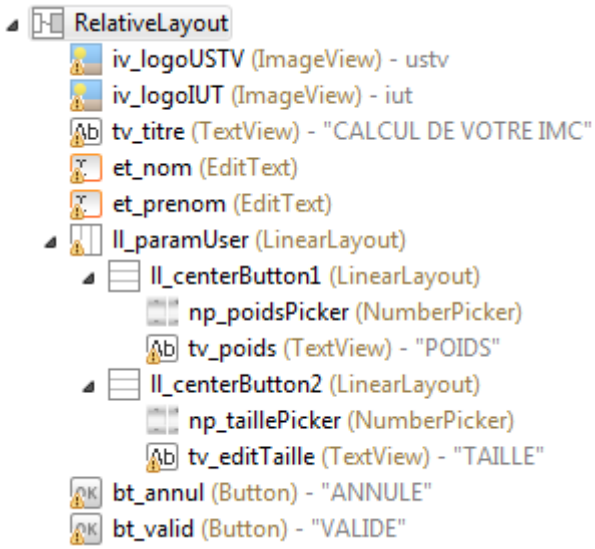
```
taillePicker.setMaxValue(100);
taillePicker.setMinValue(0);
taillePicker.setWrapSelectorWheel(true);
```

- Essayer d'aligner de logo IUT sur la droite en jouant sur layout_gravity. Que se passe t'-il ?

Relativelayout

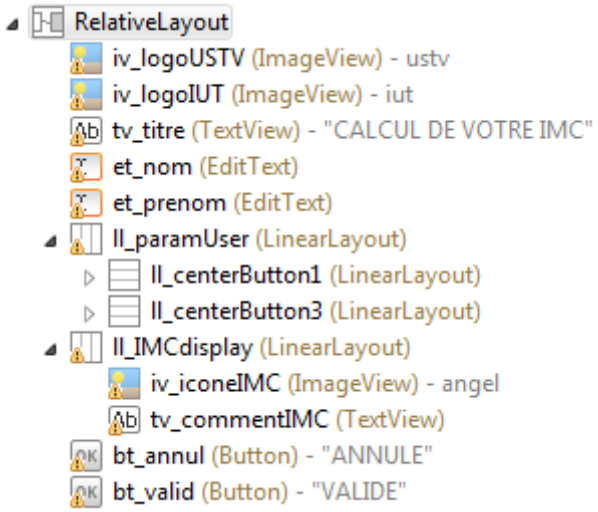
Nous allons améliorer la disposition précédente en utilisant maintenant le conteneur relativelayout à la racine du layout. Faire une copie du projet précédent et renommer. N'abusez pas du designer et préférez les

copier/coller dans le xml.
N'oubliez la configuration des NumberPicker dans le code Java. Le centrage des NumberPicker ,est traité dans votre poly (astuce des LinearLayout imbriqué).
Pour gérer l'espacement entre les éléments utilisez la balise : `android:layout_marginTop="xxdp"`



Gérer les clics

Nous allons exploiter les données et gérer le bouton VALIDE. Ouvrez le projet ClicHandler existant qui est à compléter.
L'interface visuelle est légèrement modifiée comme suit.



- Rajouter dans le layout un linearlayout horizontal ll_IMCdisplay incluant un ImageView + un textView. Ce linearlayout sera invisible au départ (`android:visibility="invisible" )`
- Modifier les attributs des 2 bouttons afin de les aligner en bas de page à droite.
- Dans votre code Java déclarer en tant que champs privés vos NumberPicker ainsi que le textView et l'imageView

précédemment rajouté. Implémenter la gestion du clic selon la méthode 1 de votre poly. Tester le clic changeant la visité du linearlayout ll_IMCdisplay : `ll.setVisibility(View.VISIBLE);`

- Mettre en place une methode privé `private double calculIMC(int poids, int taille)` qui calcule et renvoie l'IMC. Cette méthode est appelée depuis le gestionnaire de clic.
- Compléter le handler associé au bouton de validation
  Conseils et explications:
  Récupération de la valeur d' un NumberPicker `this.poidsPicker.getValue();`
  Contrôle de la visibilité d'un du élément : `ll.setVisibility(View.VISIBLE);`
  Manipulation et formatage des chaines de caractères

```
DecimalFormat f = new DecimalFormat();
f.setMaximumFractionDigits(1);
String imcChaine=f.format(imc);
String comment= "IMC: "+imcChaine;
```

  Mise à jour l'icône en fonction de la valeur de l'IMC
  Exemple :

```
if (imc<18.5){ //maigreur
                this.iconeIMC.setImageResource(R.drawable.sad);
                comment=comment+" attention : etat de maigreur";
          }
          else {

….....................
```

- Conclusion sur cette exercice : Cette interface visuelle ,quoique que fonctionnelle, est un exemple de mauvaise pratique dans un contexte objet.
  En effet, pour ne citer que cela, nous constatons que la couche métier (calculer l'IMC) n'est pas modélisée sous formes d'objets (classes et encapsulations). Par conséquent elle est fortement couplée à l'IHM , difficile à maintenir et à étendre (plusieurs modes de calcul de l'IMC, cas des enfants, gestions de plusieurs utilisateurs etc...) et peu sécurisée.

Ressources

- Copier et coller le précédent projet.Renommer. Remplacer tous les codages en « dur » du layout par des ressources (marges, textes etc...)
  Mettez un peu couleur dans la précédente interface

Navigation

L'objectif est de réaliser une navigation simple entre 2 activités avec passage de données.

Vous travaillerez sur le squelette projet Navigation1 fourni.

TP ANDROID

LES BASES

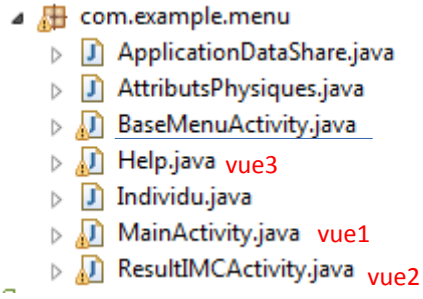
- Dans un premier temps compléter le projet de telle manière à lancer la page d'affichage sur le clic (pas de transmission de données entre les deux activités)
- Gérer la transmission des données et la mise à jour de l'affichage

Menu

L'objectif est de réaliser une navigation simple par le biais d'un menu.

Le scénario est le suivant :
- saisie des données puis clic sur valide
- calcul et visualisation de l'imc en démarrant l'activité par le biais du menu
- visualisation d'une aide en démarrant l'activité par le biais du menu

Un squelette a compléter est fourni (exo Menu)



Stratégie de mise en place du menu : Plutôt que se surcharger pour chaque activité onOptionsItemSelected et onCreateOptionsMenu une classe BaseMenuActivity , qui définira un menu commun à nos trois activités, sera crée. Pour ce faire les trois activités de notre application hériterons de BaseMenuActivity.

Stratégie de passage des valeurs : Compte tenu de la stratégie adoptée pour le menu nous utiliserons l'équivalent d'une variable globale vue par toutes les activités. Pour cela nous exploiterons le fait que les objets définis au niveau de l'application sont accessibles depuis les activités. Il faut pour cela hériter de la classe Application , personnaliser notre application puis enregistrer l'application dans le manifeste.

```
public class ApplicationDataShare extends Application{

    private static ApplicationDataShare applicationInstance;
    private Individu personne;
                                        Ma référence globale
    @Override
    public void onCreate() {
        super.onCreate();

<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
     android:name="com.example.menu.ApplicationDataShare"
    android:theme="@style/AppTheme" >
    <activity
        android:name="com.example.menu.MainActivity"
```

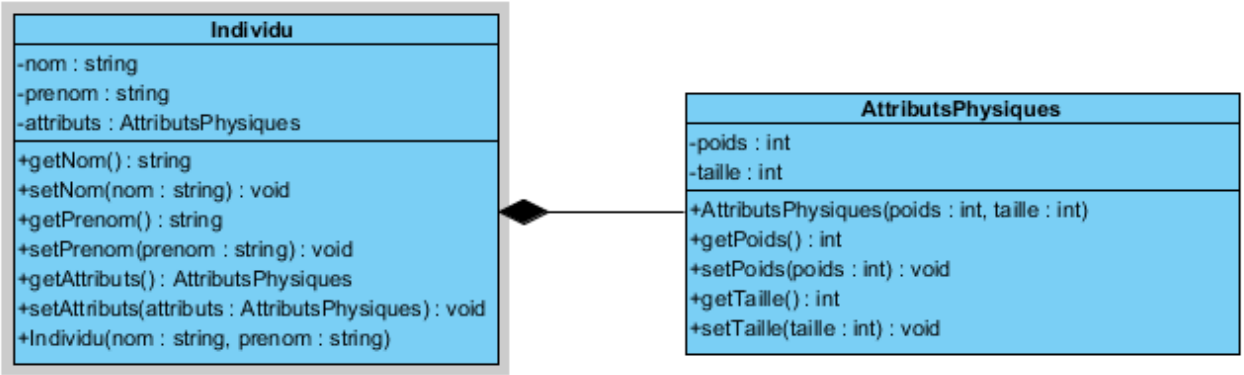
J'enregistre mon application customisée !!

TP ANDROID

LES BASES

Le travail à réaliser sur le squelette à compléter est le suivant :

- Compléter la classe BaseMenuActivity et la ressource menu afin de proposer le schéma de navigation attendu.Les icônes utilisées sont : *@android:drawable/ic_menu_edit, @android:drawable/ic_menu_gallery et @android:drawable/ic_menu_info_details*
- Compléter les fichiers et Individu correspondant aux classes définis ci-dessous



- Compléter le code manquant dans MainActivity et ResultIMCActivity