
	ISET SFAX	DEPARTEMENT TECHNOLOGIES DE L'INFORMATIQUE	
Atelier 6			
Matière : ATELIER DEVELOPPEMENT MOBILE NATIF			DSI3
Enseignants : S. Hadhri & M. Hadji & H. Souissi			

Exercice 1

L'objectif de cet exercice est de programmer une application Android nommée « Saisons » qui permet de parcourir les images des 4 saisons en utilisant le toucher : On voudra pouvoir passer d'une saison à la suivante en passant le doigt de droite à gauche et inversement.

Son interface (formée par une ImageView et un TextView) est la suivante :



Travail demandé :

1. Ajouter le code nécessaire afin de pouvoir passer d'une saison à la suivante en passant le doigt de droite à gauche et inversement.
 - Appliquer l'écouteur `setOnTouchListener` à l'`ImageView` : La fonction `onTouch (View v, MotionEvent event)` sera appelée à chaque toucher.
 - Utiliser la méthode `MotionEvent.getAction()` pour récupérer l'action à l'écran, elle peut avoir comme valeurs :
 - `MotionEvent.ACTION_DOWN` : L'utilisateur vient d'appuyer sur l'écran. C'est la première valeur récupérée suite à une action sur l'écran
 - `MotionEvent.ACTION_UP` : Envoyé lorsque l'utilisateur cesse d'appuyer sur l'écran
 - Utiliser également la méthode `event.getX()` qui renvoie la position de l'abscisse.

Remarque :

Changer le type de retour de `false` à `true` afin de gérer tous les types d'événements : `ACTION_DOWN`, `ACTION_MOVE`, `ACTION_UP`, ...

Squelette de code :

```
ivSaison.setOnTouchListener(new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        switch (event.getAction())
        {
```

```

// when user first touches the screen we get x coordinate
case MotionEvent.ACTION_DOWN:
{
    x1 =.....
    break;
}
case MotionEvent.ACTION_UP:
{
    x2 =.....

    //if left to right sweep event on screen
    if (x1 < x2)
    {
        .....
    }

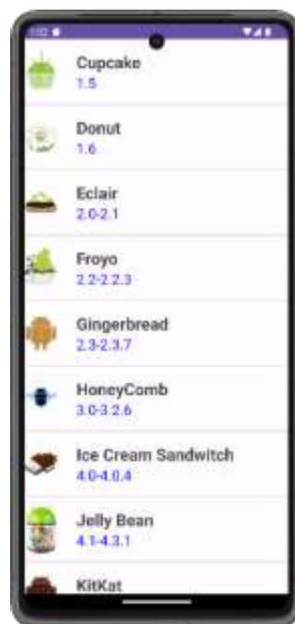
    // if right to left sweep event on screen
    else
    {
        .....
    }
    .....
    break;
}
}
return true;
}
});

```

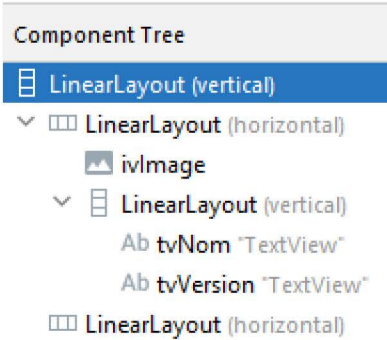
2. Ajouter un menu formé par 4 items représentant les saisons : Au clic d'un menu, on atterrit directement à la saison correspondante.
3. Ajouter « Quitter » au menu précédant qui avant de fermer l'application affiche une alertDialog qui demande de confirmer le choix.

Exercice 2

« **VersionAndroid** » est une application Android permettant d'afficher la liste des versions d'Android ainsi que leurs logos en utilisation une **RecyclerView**.



1. Insérer dans le layout « **activity_main.xml** » une RecyclerView afin d'afficher la liste des versions d'Android
2. Ajouter un layout « **item_row.xml** » pour préparer l'interface d'un élément (ligne) selon cette structure :



3. Créer la classe « **Item.java** » qui représente le modèle de l'application et qui possède les attributs suivants :
 - *nom* et *version* de type chaîne de caractère
 - *image* de type entier
4. Générer le *constructeur* ainsi que les *getters* de cette classe
5. Ajouter la classe « **MyViewHolder.java** » héritant de la classe « **RecyclerView.ViewHolder** ».
 - a. Déclarer dans cette classe les attributs suivants :
 - *tvNom* et *tvVersion* de type `TextView`
 - *ivImage* de type `ImageView`
 - b. Initialiser dans le constructeur de la classe « **MyViewHolder.java** » ses attributs avec leurs correspondants du layout « **item_row.xml** »

```

public MyViewHolder(View itemView) {
    super(itemView);
    tvNom = itemView.findViewById(R.id.tvNom);
    tvVersion = itemView.findViewById(R.id.tvVersion);
    ivImage = itemView.findViewById(R.id.ivImage);
}

```

6. Ajouter la classe « **MyAdapter.java** » héritant de la classe **RecyclerView.Adapter<MyViewHolder>**
 - a. Déclarer dans cette classe l'attribut suivant : *items* de type `List<Item>`
 - b. Implémenter ses méthodes : `onCreateViewHolder`, `onBindViewHolder` et `getItemCount`
 - c. Générer le constructeur de cette classe
 - d. Modifier la méthode `onCreateViewHolder` :

```

@Override
public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View view =
        LayoutInflater.from(parent.getContext()).inflate(R.layout.item_row,
            parent, false);
    return new MyViewHolder(view);
}

```

- e. Ajout le script suivant dans la méthode `onBindViewHolder` pour afficher la liste des Items :

```

final Item item = items.get(position);
holder.tvNom.setText(item.getNom());
holder.tvVersion.setText(item.getVersion());
holder.ivImage.setImageResource(item.getImage());

```

- f. Modifier la valeur retourner de la méthode `getItemCount` (0) par `items.size()`
7. Télécharger les images (urlz.fr/nSFd) et les mettre dans le dossier **drawable**
 8. Appliquer dans la classe « **MainActivity.java** » les changements suivants :
 - a. Déclarer une variable de type **RecyclerView**

- b. Initialiser cette variable dans la méthode **OnCreate()**
- c. Définir la fonction **remplir()** et l'appeler dans la méthode **OnCreate()** ; sachant qu'elle permet de remplir la RecyclerView par la liste des objets de la classe **Item** :

```
private void remplir() {  
    List<Item> items = new ArrayList<Item>();  
    items.add(new Item("Cupcake", "1.5", R.drawable.ic_launcher_background));  
    items.add(new Item("Donut", "1.6", R.drawable.ic_launcher_foreground));  
    ...  
    recyclerView.setLayoutManager(new LinearLayoutManager(this));  
    recyclerView.setAdapter(new MyAdapter(items));  
}
```

- 9. On souhaite ajouter un écouteur de type clic à la RecyclerView :
 - a. Changer dans **MyAdapter** l'objet items en static
 - b. Ajouter dans **MyViewHolder** un écouteur de type **OnClick** sur l'élément itemView pour afficher dans un Toast le nom de la version Android.