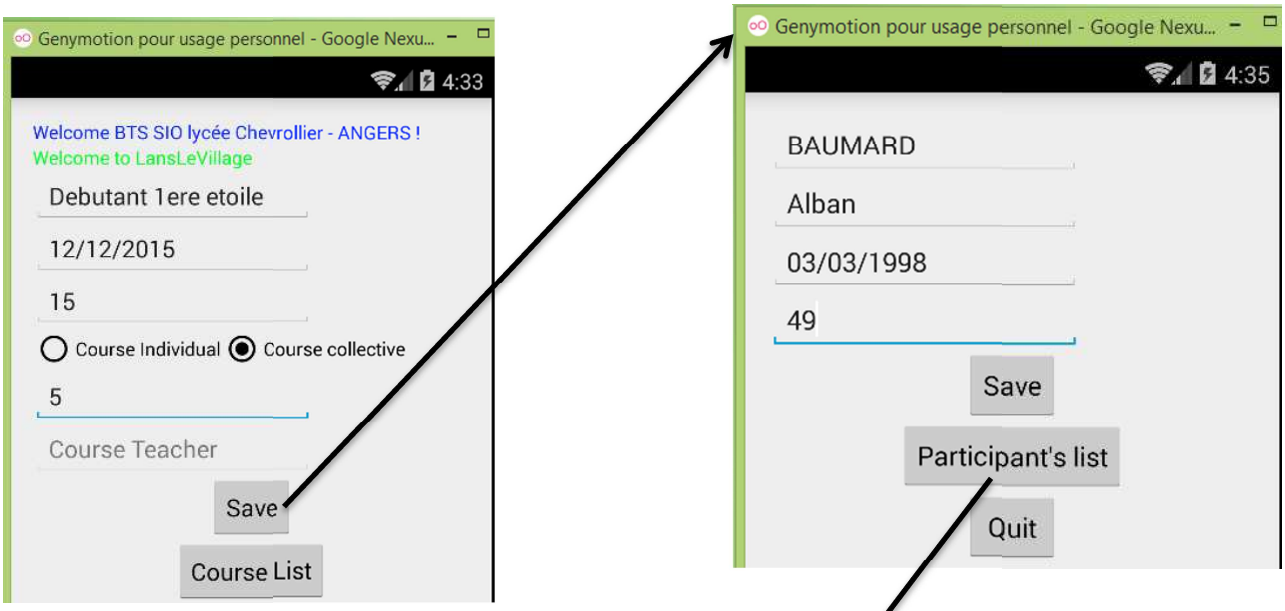
# TPB5 : AFFICHAGE EN LISTE

Objectif du TP :

- Affichage autrement que dans un Toast.
- Utiliser le principe de la ListView…

## AFFICHAGE DES PARTICIPANTS AVEC UNE LISTVIEW




**affichage de la liste triée par ordre des dates de Naissance**

**Explication :**

On va donc utiliser une classe d'android la ListView, qui comme son nom l'indique est une liste de vue comprenant des **items ;** pour afficher une liste **d'item** dans celle-ci, il lui faut un adaptateur de données.
Ici chaque élément de la liste est une représentation complexe d'une instance de la classe participant.

Nous devons créer :

**A.** une vue " **vue_un_participant** "qui va nous permettre d'afficher **un** élément de la liste**.**

**A'** une vue " **affiche_Liste_Participant** "qui va nous permettre d'afficher **toute la listeView**

**B.** une classe de type **adapter** qui va permettre de **remplir la vue d'un élément de la liste** et de gérer les événements de l'utilisateur (clic,...)

**C.** l'Activity qui permettra **d'afficher la listview** et à partir d'une collection, d'appeler l'adapter pour remplir chaque élément de la liste.

**C** **ACTIVITY**
AfficheListeParticipant.JAVA

```
oncreate {
lvListe = (ListView)
      findViewById(R.id.lvListeP);

Participantadapter adapter = new
Participantadapter(this, malisteP);

lvListe.setAdapter(adapter);

}
```

**ici** `maListeP` est une ArrayList qui récupère tous les participants à afficher dans la listeView (ces participants sont présents dans les cours stockés dans la BDDO)

**A'** **LAYOUT**
**affiche_liste_participant**.XML

layout contenant juste une **ListView** nommée ***lvListeP***

**A** **LAYOUT**
**vue-un-participant.**XML

nom prenom dateNais dept

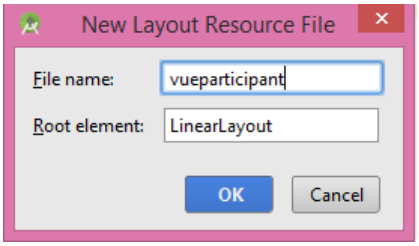
**B** classe **ParticipantAdapter**

```
public ParticipantAdapter(Context context,
List<Participant> vListeP) {
    }

public View getView(int position, View
convertView, ViewGroup parent) {
```

❶ *formate la vue avec le ViewHolder*
❷ *charge la vue d'un élément*
❸ *récuperation de la liste de tous les éléments à afficher*

```
    }

    private class ViewHolder {
      TextView vnom, vprenom, vdatenaiss, vnumDep;}
```

# A. CRÉATION DE LA VUE D'UN ÉLÉMENT DE LA LISTE




Créer un Layout **vue_un_participant** ( res / layout clic droit new layout resource File (xml)) qui permet d'afficher **un seul** participant tel que décrit au-dessous :

Nom Prénom dateNaissance NumDépartement

⇨ **choisir 4 textView et les positionner de manière *horizontale*.**

*Rappel*

| | |
|---|---|
| fill_parent | The view should be as big as its parent (minus padding).<br>This constant is deprecated starting from API Level 8 and is replaced by match_parent. |
| match_parent | The view should be as big as its parent (minus padding). Introduced in API Level 8. |
| wrap_content | The view should be only big enough to enclose its content (plus padding). |

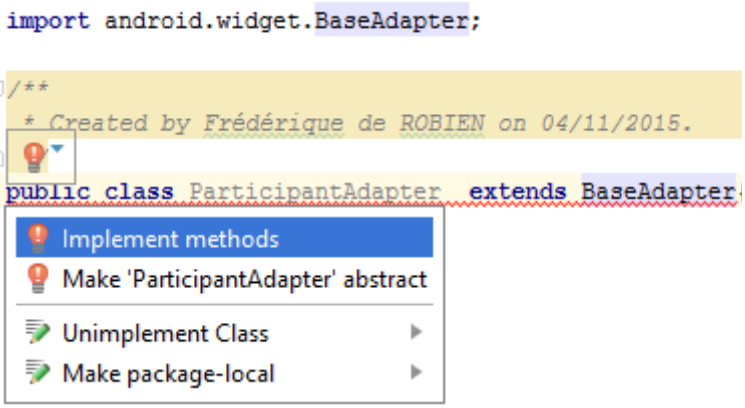
Nous venons donc de créer notre gabarit d'affichage (Vue spécifique d'un élément de la liste) d'un objet participant pour notre liste.

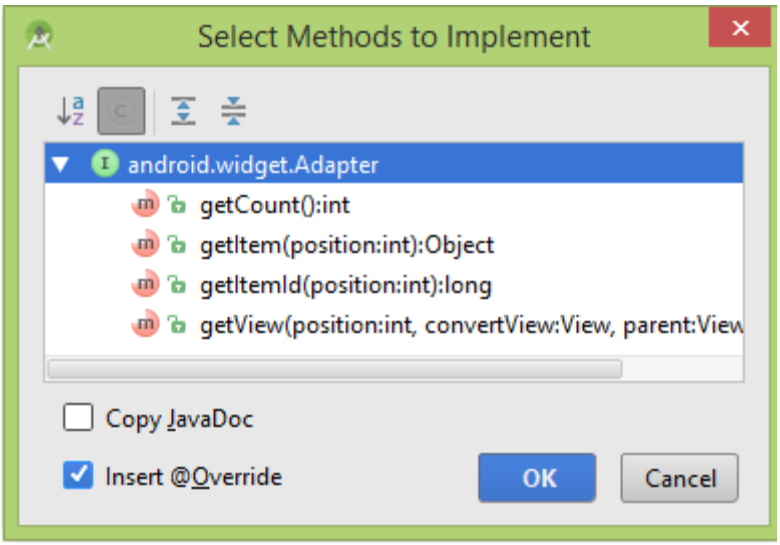
# A'. CRÉATION DE LA VUE DE LA LISTVIEW (AFFICHELISTPARTICIPANT)

Créer un nouveau layout qui contient juste une ListView nommée ***lvListeP***

# B .CRÉATION DE NOTRE CLASS DE TYPE ADAPTER

Maintenant on va se pencher sur notre adaptateur personnalisé.
Créez une classe nommée ParticipantAdapter, héritant de la classe BaseAdapter.




Le générateur de classes a rajouté directement les méthodes à remplir pour le bon fonctionnement de l'adaptateur car BaseAdapter possède ces **méthodes abstraites qu'il faut donc redéfinir** !

On retrouvera ces méthodes :

**getCount**() qui retournera le nombre d'éléments dans notre liste.

**getItem**() qui retournera notre objet participant à la position indiquée.

**getItemId**() qui retournera l'id du participant.

**getView()** qui retournera la vue de l'item pour l'affichage.

Rajoutons un type **List<Participant>,** dans notre classe et créons un constructeur par défaut prenant une liste et un contexte en paramètres.

Rajoutons aussi à notre classe un **LayoutInflater**, qui aura pour mission de **charger** notre fichier **vue_un_participant.xml** pour l'item, c'est là que l'on utilisera le contexte.

```
//Déclarer la liste nommée ListeP (de type List et non ArrayList typée avec la classe Participant)
```

✍_____

```
private LayoutInflater inflater;

public ParticipantAdapter(Context context, List<Participant> vListeP) {

        inflater = LayoutInflater.from(context);
        // A compléter

        ✍_____________________________
        }
```

Remplissons nos différentes méthodes :

- **getCount** qui retournera la taille de la liste :
- **getItem** qui retournera l'item:
- **getItemId** qui retournera la position de l'item.

```
@Override
public long getItemId(int position) {
    return position;
}
```

- **getView** qui récupèrera la Vue

Avant de modifier la méthode getView nous devons créer une classe (une classe dans une classe! c'est une classe INTERNE) qui sera nommée **ViewHolder**.

Elle nous servira à mémoriser les éléments de la liste en mémoire pour qu'à chaque rafraichissement l'écran ne scintille pas (c'est un genre de buffer/cache comme en graphisme).

```
private class ViewHolder {
    TextView vnom, vprenom, vdatenaiss, vnumDep;
}
```

💣Le nom des Textview n'a pas de rapport avec votre layout **vue_un_participant.xml**

**getView** utilise le **ViewHolder**, vérifie que la **view** présente n'est pas **null** sinon la crée et ensuite charge l'XML (**vue_un_participant.xml**) en mémoire pour l'attribuer à notre objet.

Et enfin taggue notre objet pour pouvoir le récupérer à la prochaine mise à jour graphique.

Et pour finir, attribue les données et retourne la vue.

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {

	ViewHolder holder;
	if(convertView == null) {
		holder = new ViewHolder();
		convertView = inflater.inflate(R.layout.vue_un_participant,null);
		holder.vnom = (TextView)convertView.findViewById(R.id.tvNom);
		etc...

		convertView.setTag(holder);
	} else {
		holder = (ViewHolder) convertView.getTag();
	}
	holder.vnom.setText(listeP.get(position).getNom());
	etc...
	return convertView;
}
```

 À adapter selon votre Layout **vue_un_participant** et votre classe **Participant** getter/setter

*aide pour le numéro de département qui est en byte :*

```
holder.vnumDep.setText(((Byte)listeP.get(position).getNumDepartement()).toString());
```

*aide pour la gestion de la date de naissance :*

```
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
et faites : String dateString = sdf.format(listeP.get(position).getDateNais());
```

## C. CRÉER L'ACTIVITY AFFICHELISTEPARTICIPANT

Créer une nouvelle activity nommé *AfficheListeParticipant*

Dans le Linearlayout ajouter une listview d'id *lvListe*P permettant d'afficher notre liste de participant

Vous pouvez améliorer votre listview avec des séparateurs
exemple :

```
android:divider="#000000"
android:dividerHeight="1dp"
```

Dans votre activity déclarer

```
private ListView lvListe;
```

puis ajouter à la fin de la méthode onCreate(Bundle savedInstanceState) afin d'afficher notre liste

```
lvListe = (ListView)findViewById(R.id.lvListeP);
Participantadapter adapter = new Participantadapter(this, malisteP);
lvListe.setAdapter(adapter);
```

**La question est ici comment récupérer malisteP** (liste de tous les participants à afficher dans la listView)**?**

Il suffit de déclarer malisteP comme une list de Participant, une instance de la classe Modele et d'appeler la méthode listeParticipants (À CRÉER en s'inspirant de listeCours) pour initialiser malisteP.

## TEST DE LA CRÉATION DE VOTRE LISTE.

(A partir de votre ActivityParticipant faites appel à la nouvelle activity Affichelisteparticipant sur le click du bouton Liste des Participants afin de tester votre affichage de la liste de vos participants.)

```
Intent i = new Intent(this, MaDeuxiemeActivite.class);
startActivity(i);
```

💣 Le this est souvent remplacé par getApplicationContext()

## PROBLÈME : LA LISTE CONTIENT TROP DE PARTICIPANTS !

Tel que nous avons initialisé la liste, nous avons tous les participants de TOUS les cours OR nous voulons que les participants du cours que nous venons de créer !

*À vous de trouver la solution ? (Nous l'avons déjà fait précédemment !)*

***aide* : 2 choses à faire :**

**1ère:** *Pour afficher que les participants d'un cours précis, on passe à l'activity le cours récuperé en BDO*

**2ème: dans la classe Modèle, écrire une nouvelle méthode listeParticipants**

```
public ArrayList<Participant> listeParticipants(Cours c) {

    /* A partir du cours, récupérer les participants à ajouter à la liste des participants à afficher */
    /*obligation de tester si c'est un cours collectif ou individuel*/
```

## SUITE DU TP : CRÉER UNE LISTE QUI AFFICHE LES COURS !

Reproduisez la même logique pour afficher les cours dans une ListeView !

👁 + 🧠 + 💻 = *Bon développement ….*

Observation + Réflexion + Réalisation