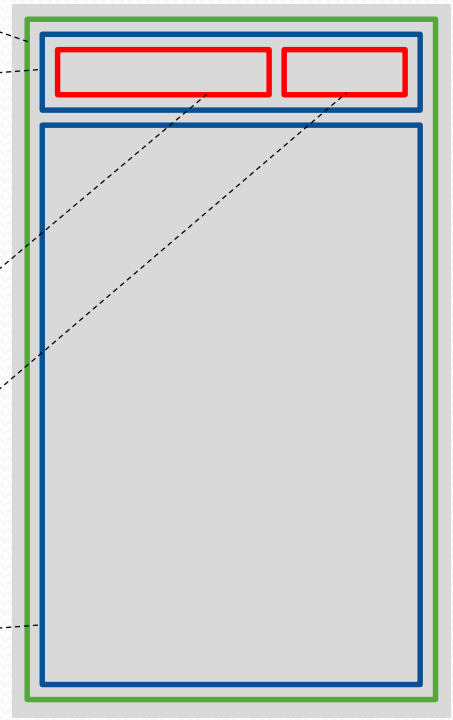
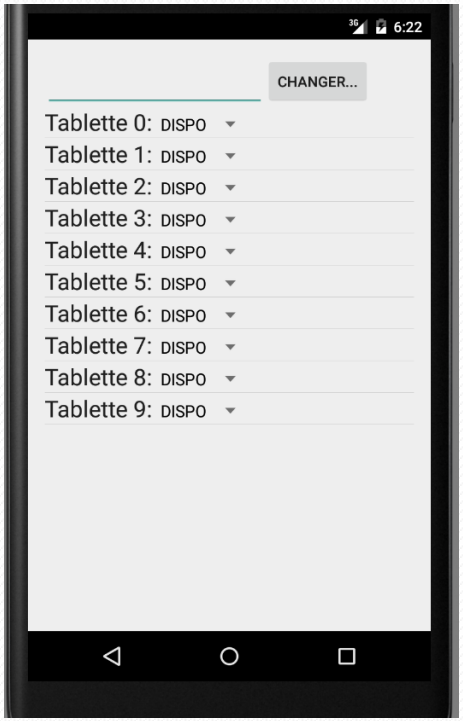
TP2. Layout

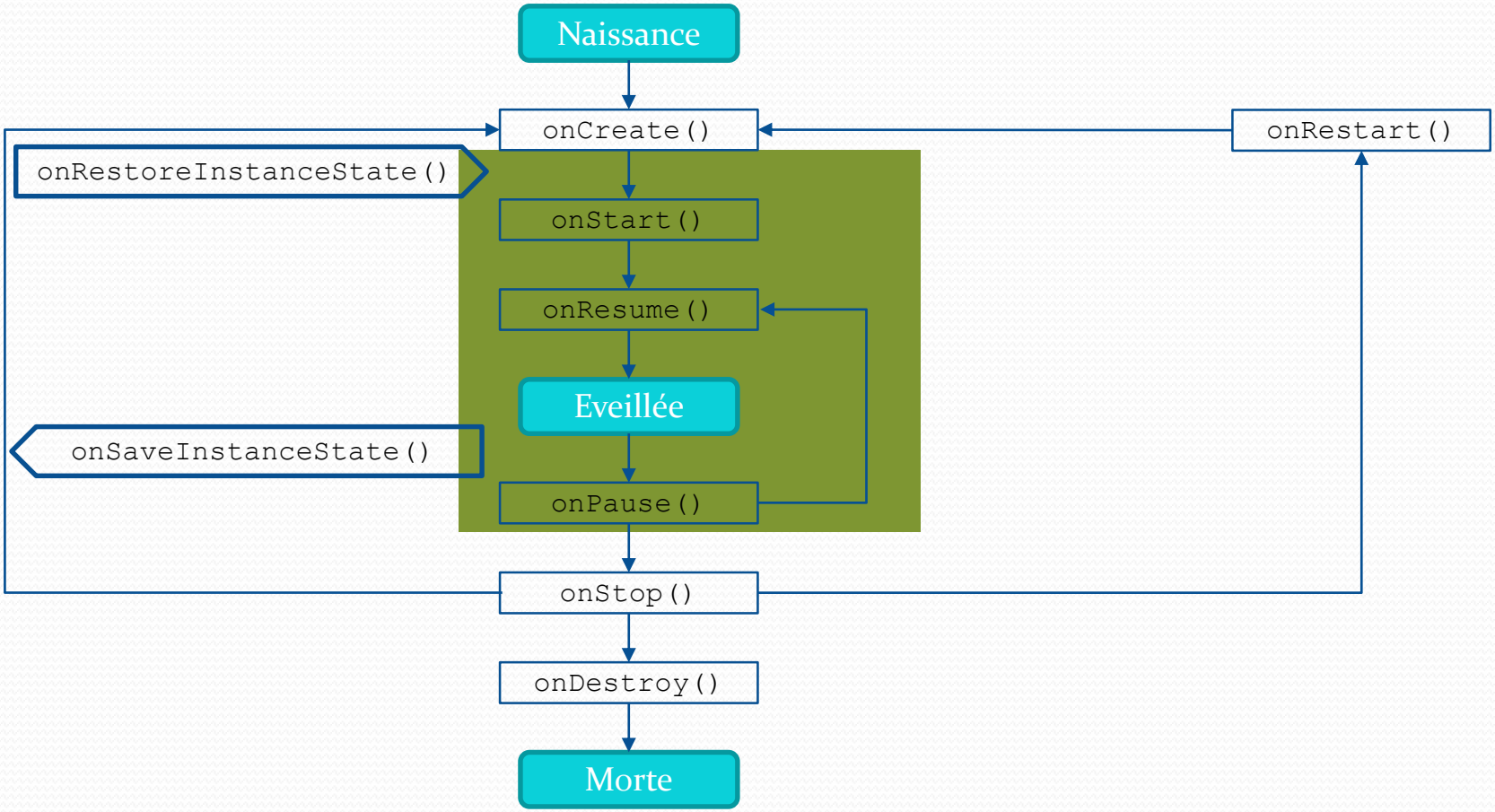
```
<LinearLayout ...
   android:orientation="vertical"
                              ...
   android:id="@+id/FirstView¤">
   <LinearLayout
         android:orientation="horizontal"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content">
         <EditText
                              ...
            android:inputType="date"
            android:id="@+id/dateItemt"/>
      <Button
            android:text="@string/Changer"
            android:id="@+id/button"
                              ...
            android:onClick="doOnPrefs" />
      </LinearLayout>

   <ListView
                              ...
      android:id="@+id/liste" />
</LinearLayout>
```





TP2. Cycle de vie



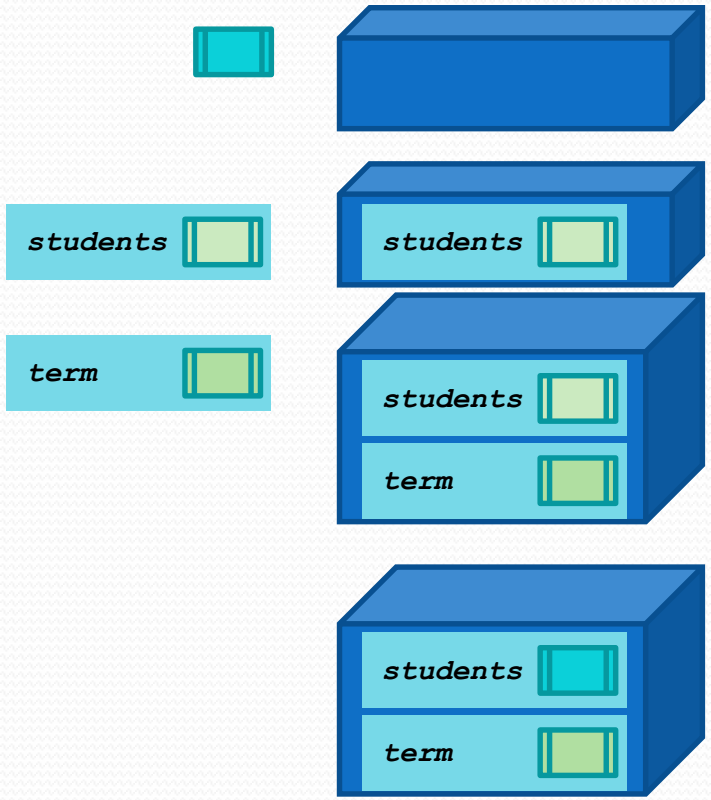
TP2. Sauvegarde, restauration

onSaveInstanceState(Bundle outState)

```
int[] students ;
```

```
outState.putIntArray("students", students) ;
```

```
outState.putStringArray("term", tablettes) ;
```



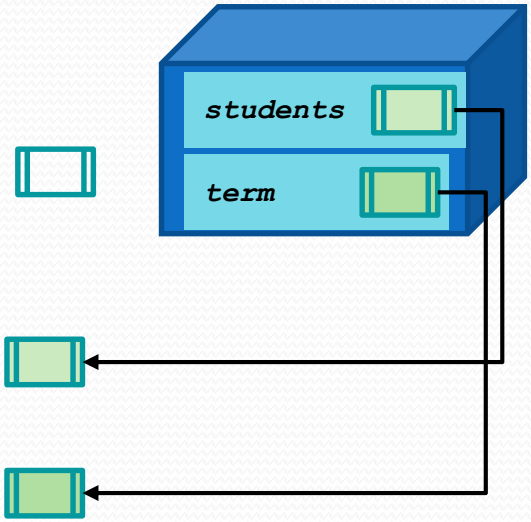
TP2. Sauvegarde, restauration

onRestoreInstanceState(Bundle inState)

```
int[] students ;
String[] tablettes ;
```

```
students = inState.getIntArray("students") ;
```

```
tablettes = inState.getStringArray("term") ;
```



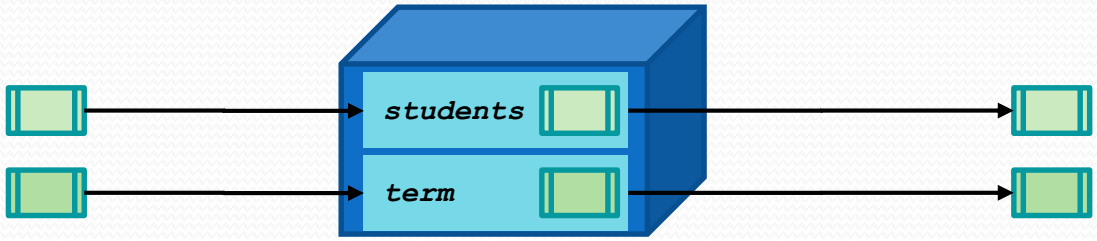
TP2. Vue d'ensemble

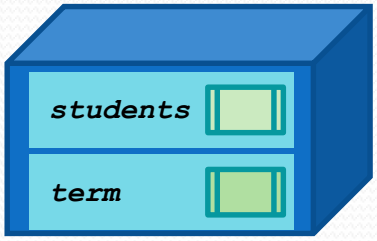
onSaveInstanceState(inState)

onRestoreInstanceState(inState)

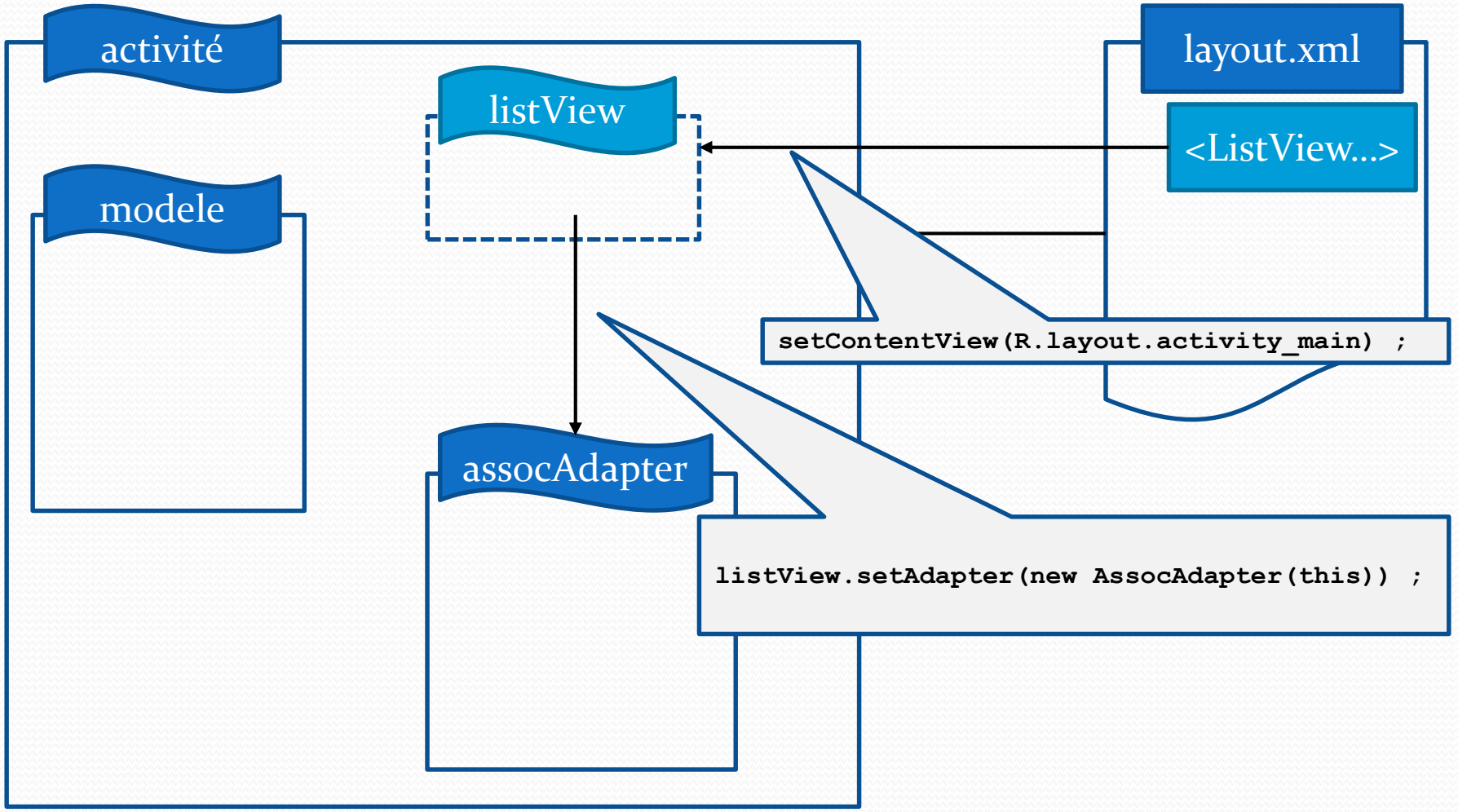
Modele.save(inState) ;

Modele = new Modele.save(inState) ;

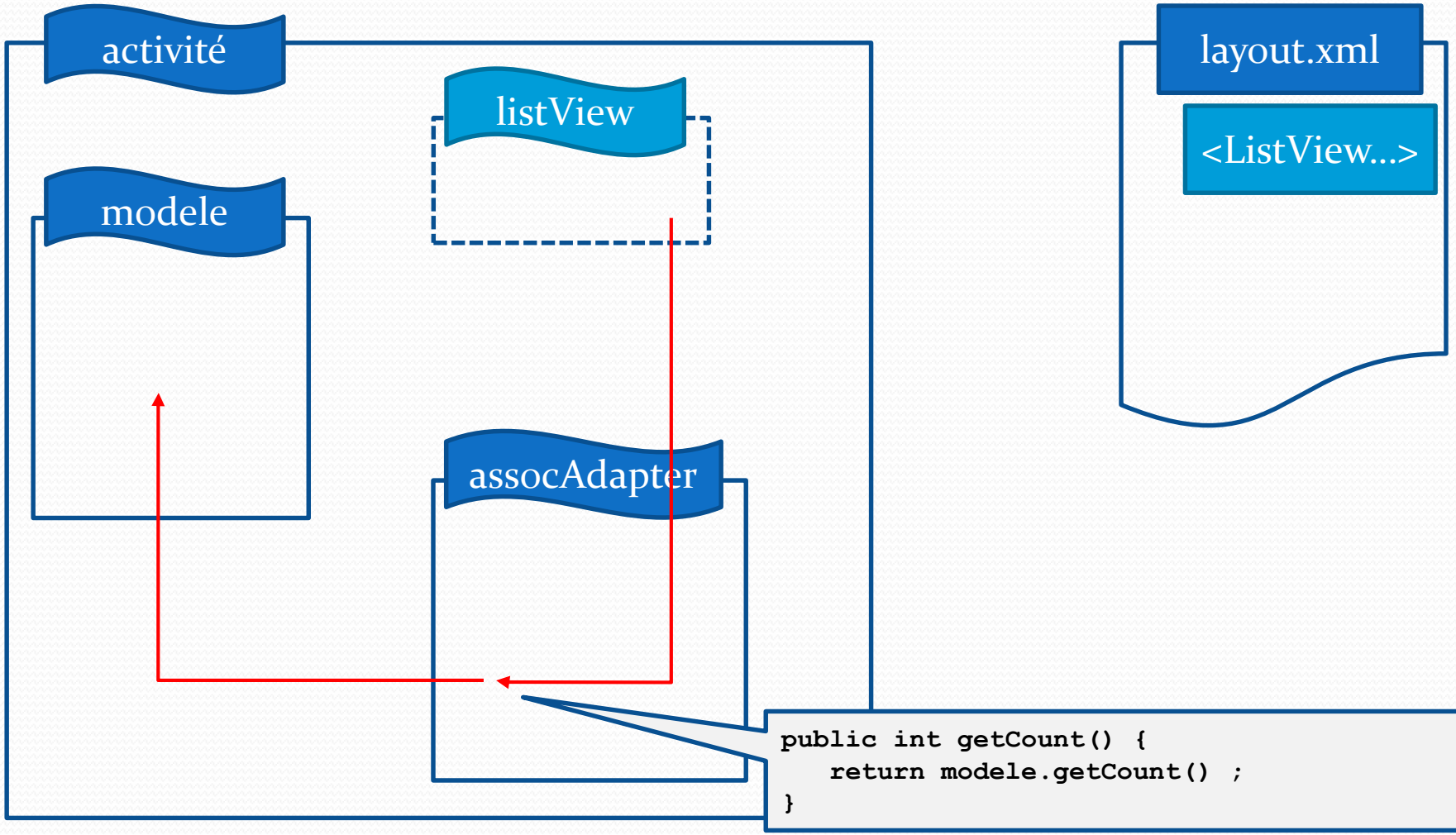




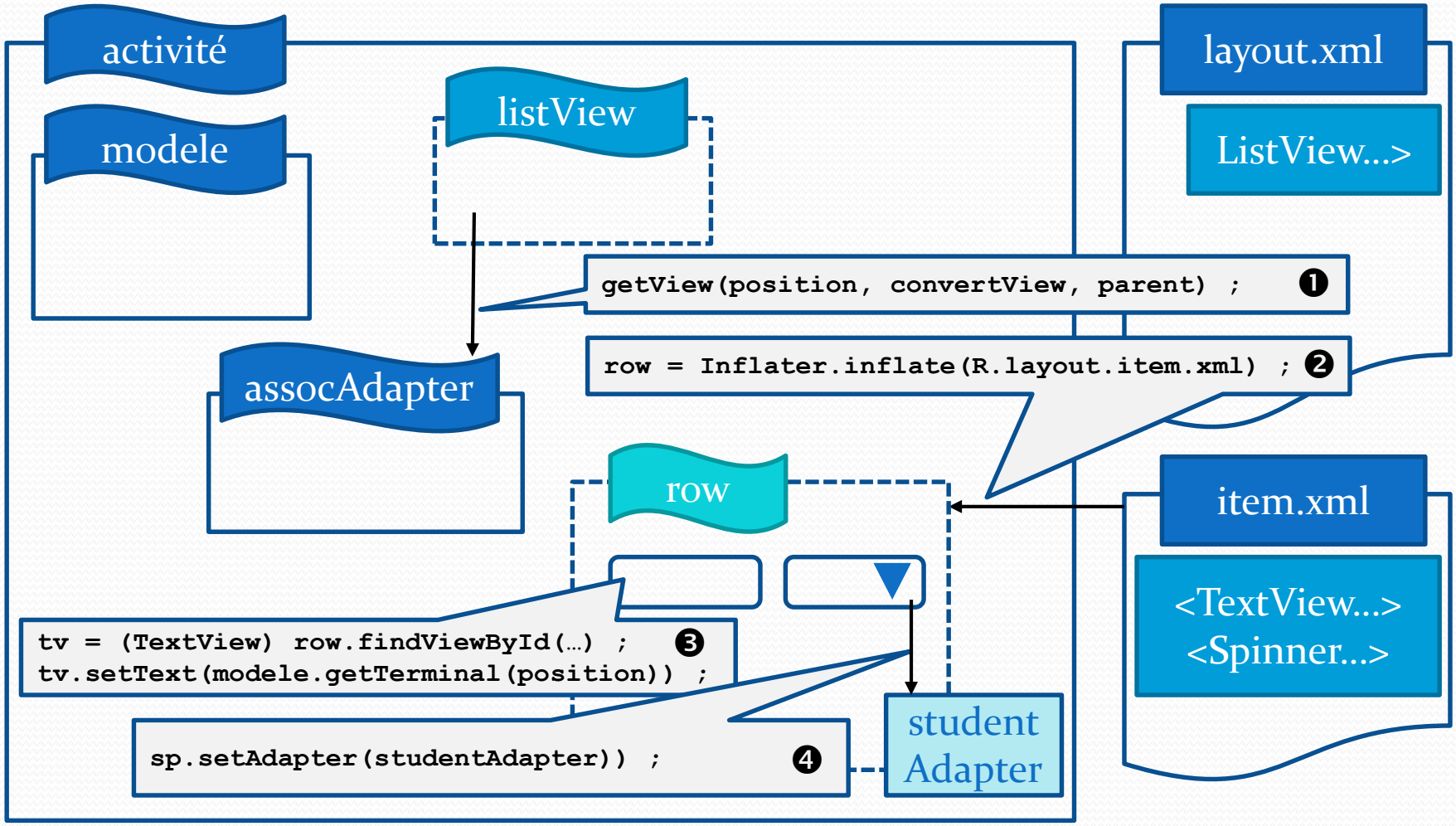
TP2. Structure de l'activité.



TP2. Activité (flot des données)



TP2. Activité (mise en forme)



Classe Modele.

```
public Modele() {
        tablettes = getResources().getStringArray(R.array.tablets) ;

        /* Ci-dessous le code permettant d'initialiser le tableau tablettes à partir du nombre de tablettes et du préfixe supposés
        existant dans les références. */
         */
//          SharedPreferences mySP = getSharedPreferences(MY_PREFS, Activity.MODE_PRIVATE) ;
//          int nbTablettes = mySP.getInt(NB_TABS, NB_TABS_DEF) ;
//          String radTab = mySP.getString(RAD_TABS, RAD_TABS_DEF) ;
//          tablettes = new String[nbTablettes] ;
//          for (int i = 1 ; i <= nbTablettes ; i++) {
//              tablettes[i] = radTab + i;
//          }
        // RAZ tableau students.
        students = new int[tablettes.length] ;
        for (int i = 0 ; i < students.length ; i++) {
           students[i] = 0 ;
        }
    }
    /* Reconstruction à partir d'un bundle */
    public Modele(Bundle inState) {
       tablettes = getResources().getStringArray(R.array.tablets) ;
       students = inState.getIntArray("students") ;
    }
    /* Sauvegarde vers un Bundle */
    public void save(Bundle outState) {
       outState.putIntArray("students", students) ;
    }
    /* Fonction d'accès aux données */

    /**
     * GetCount: retourne le nombre de tablettes.
     */
    public int getCount() {
       return tablettes.length ;
    }
```

Classe Modele.

```
/**
     * getStudent:  Retourne le numéroi de l'étudiant associé à une tablette.
     *  (o si pas d'étudiant -> tablette disponible)
     * @param position Le numéro de la tablette visée (de o à ... )
     * @return le numéro de l'étudiant associé.
     */
    public int getStudent(int position) {
       return students[position] ;
    }

    /**
     * getTerminal: Calcule le nom d'une tablette.
     * @param position: le numéro de la tablette (o à ...)
     * @return Le nom courant de la tablette.
     */
    public String getTerminal(int position) {
       if (position < o) return null ;
       if (position >= tablettes.length) return null ;
       return tablettes[position] ;
    }

    /**
     * setStudent: Affecte un étudiant (spcifié par son numéro) à une tablette (spécifiée par son index).
     * Permet également de rendre disponioble une tablette (si student = o).
     * Retourne faux en cas d'érreur:
     * - Si l'étudiant a déjà une autre tablette affectée.
     * - Si Si le numéro d'étudiant n'est pas valide (entre o et nbEtudiants).
     * @param position
     * @param student
     * @return Vrai si ok, faux sinon.
     */
    public boolean setStudent(int position, int student) {
       if (student != o) {
          for (int i = o ; i < students.length ; i++) {
             if ((i != position) &&(students[i] == student)) return false ;
          }
       }
       students[position] = student ;
       return true ;
    }
 }
```

Classe StudentListener

```
/**
    * Classe qui répercute sur le modèle le changement de séléection d'un spinner
    * par le biais d'un traitemlent de l'événement onItemSelected.
    * Si la mise à jour ne peut se faire, il y a affichage d'une fenêtre d'avertissement.
    * Le context Android ainsi que la position traitée sont fournis par le constructeur.
    */
   class StudentListener implements AdapterView.OnItemSelectedListener {
       protected int position ;
       protected Context context ;
       protected Spinner spStudent ;

       /**
        * Constructeur.
        * @param context   Le contexte Android.
        * @param position  La position (c.a.d. le terminal) associée.
        * @param spStudent Le spinner contrôlé.
        */
       public StudentListener(Context context, int position, Spinner spStudent) {
           this.position = position ;
           this.context = context ;
           this.spStudent = spStudent ;
       }
```

Classe StudentListener

```
        /**
         * Déselection du spinner.
         * @param parent
         */
        public void onNothingSelected(AdapterView<?> parent) {
            Toast.makeText(context,"unselect",Toast.LENGTH_SHORT).show();
        }
        /**
         * Selection d'un élément.
         * @param parent
         * @param view
         * @param position
         * @param id
         */
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id)
        {
            /* Le modele est un attribut de la classe englobante. MainActivty */
            if (!(modele.setStudent(this.position, position))) {
                Toast.makeText(context,
context.getResources().getText(R.string.err_01),Toast.LENGTH_SHORT).show();
                spStudent.setSelection(modele.getStudent(this.position));
            }
        }
    }
```

Classe AssocAdapter

```
class AssocAdapter extends BaseAdapter implements ListAdapter {
        protected Context context ;
        public AssocAdapter(Context context) {
            super() ;
            this.context = context ;
        }
        @Override
        public boolean areAllItemsEnabled() { return true ; }
        @Override
        public boolean isEnabled(int position) { return true ;  }
        @Override
        public int getCount() { return modele.getCount() ; }
        @Override
        public Integer getItem(int position) { return modele.getStudent(position)
; }
        @Override
        public long getItemId(int position) { return position ; }
        @Override
        public int getItemViewType(int position) { return 0 ; }
        @Override
        public int getViewTypeCount() { return 1 ; }
        @Override
        public boolean hasStableIds() { return true ; }
        @Override
        public boolean isEmpty() { return (getCount() == 0) ; }
```

Classe AssocAdapter

```
@Override
   public View getView(int position, View convertView, ViewGroup parent) {
           LayoutInflater inflater = getLayoutInflater() ;
           View row = inflater.inflate(R.layout.layout_association, parent, false) ;
           TextView tvTablette = (TextView) row.findViewById(R.id.tvTablette) ;
           tvTablette.setText(modele.getTerminal(position)) ;
           Spinner spStudent = (Spinner) row.findViewById(R.id.spStudent) ;
           spStudent.setAdapter(studentAdapter);
           spStudent.setOnItemSelectedListener(
                   new StudentListener(context, position, spStudent));
           spStudent.setSelection(modele.getStudent(position));
           return row ;
       }
   }
```