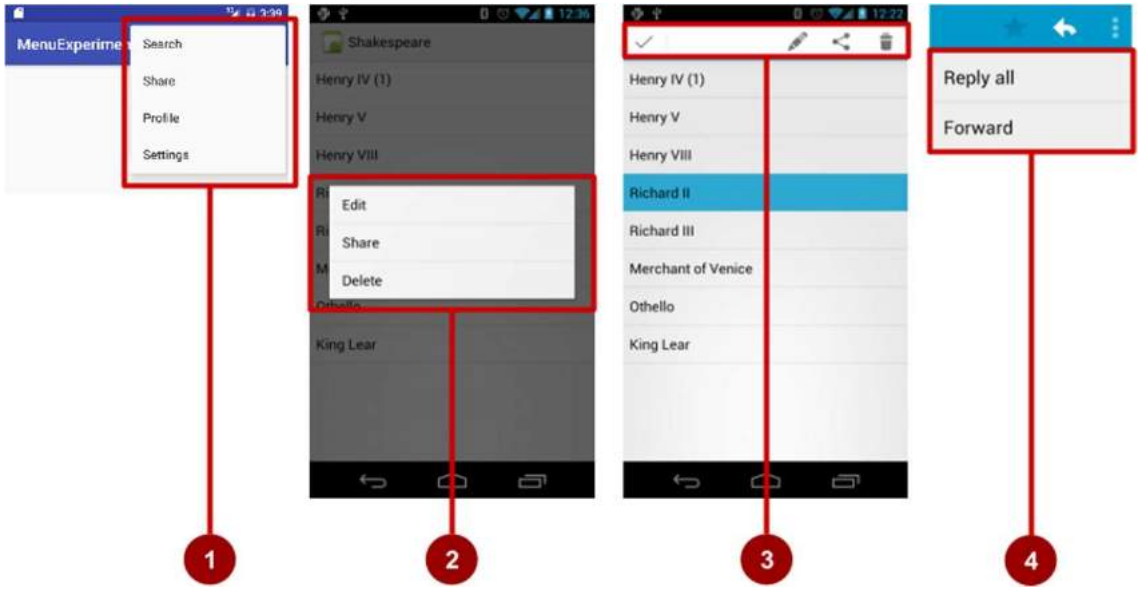
Cours :

# Widgets avancés (Les menus)

Présenté par :
Dr ZEMALI Elamine

# Les menus

- Type de menu :
  1. Menu d'option dans une barre d'application (App bar with options menu)
  2. Context menu (Menu contextuel)
  3. Contextual action bar
  4. Popup menu



# Menu d’option dans une barre d’application

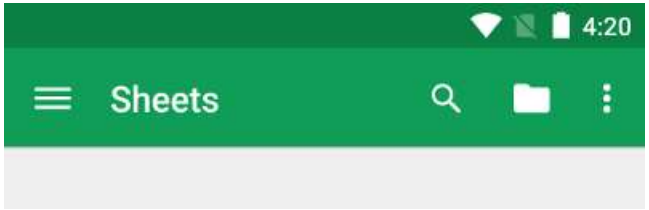


- Un Menu lié à la barre d’application.

- Qu’est ce qu’une barre d’application

# Une barre d’application

- fournit une structure visuelle et des éléments interactifs familiers aux utilisateurs.
- Les fonctions clés de la barre d'application sont les suivantes:
  - Un espace dédié pour donner une identité à votre application et indiquer l'emplacement de l'utilisateur dans l'application.
  - Accès aux actions importantes de manière prévisible, comme la recherche. (Menu)
  - Support de navigation et du changement de vue (avec onglets ou listes déroulantes).



# Comprendre Barre d’application

- Dans sa forme la plus élémentaire, la barre d’application affiche le titre de l'activité d'un côté et un menu de débordement de l'autre.

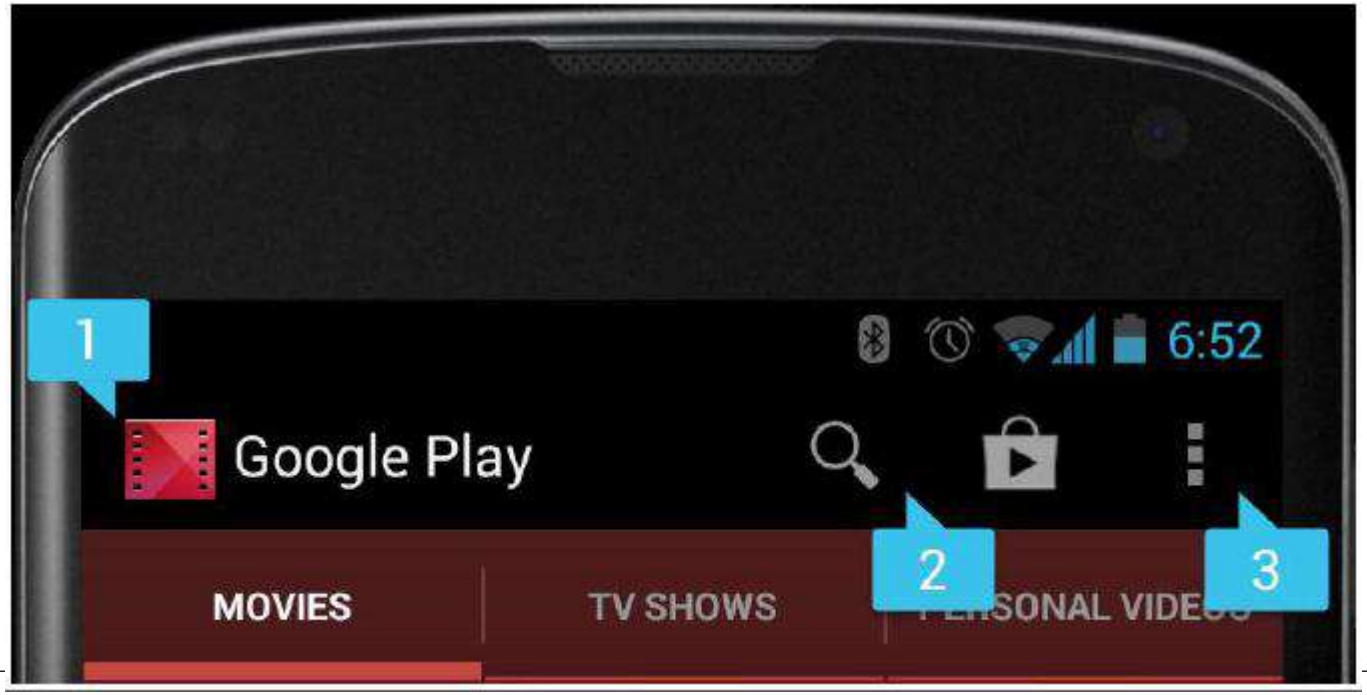


- Il existe deux types de Barre d’applicaton
  - Barre d’action (ActionBar) (par défaut)
  - Barre d’utilités (ToolBar) (la plus compatible)

- À partir d'Android 3.0 (API niveau 11), toutes les activités qui utilisent le thème par défaut ont un ActionBar comme barre d'application.
- les fonctionnalités de la barre d'application ont été progressivement ajoutées à l'ActionBar native sur diverses versions d'Android. Par conséquent, l'ActionBar natif se comporte différemment selon la version du système Android qu'un appareil peut utiliser.
- Les fonctionnalités les plus récentes sont ajoutées à la version de la barre d'outils (ToolBar) de la bibliothèque de support (support library),

# Barre d’action

- La barre d'actions (***ActionBar)*** se situe en haut d'une activité. Elle peut afficher le titre de l'activité, l’icône d’application (1), des actions qui peuvent être lancées (items de menus) (2), des vues additionnelles et un bouton pour avoir d'autres choses encore (3).



# Barre d’action

- Pour modifier les information sur une Barre d’action:



```
//Modifier le titire
getSupportActionBar().setTitle("titre");

// Activer le retour à l’activity home
getSupportActionBar().setDisplayShowHomeEnabled(true);

//changer l’icône de l’app
getSupportActionBar().setIcon(R.mipmap.icon);
```

# Barre d'utilités (ToolBar)



- Pour ajouter ToolBar il faut :
  - Ajouter v7 appcompat support library à votre projet

```
dependencies {
   implementation "com.android.support:support-core-utils:28.0.0"
}
```

  - S'assurer que votre activity hérite de AppCompatActivity.
  - Dans le manifest, sélectionner un thème sans Action bar (NoActionBar), par exemple:

```
<application
   android:theme="@style/Theme.AppCompat.Light.NoActionBar"
 />
```

# Ajouter ToolBar



- Ajouter la balise Toolbar a votre activity layout (XML).

```
<android.support.v7.widget.Toolbar
  android:id="@+id/my_toolbar"
  android:layout_width="match_parent"
  android:layout_height="?attr/actionBarSize"
  android:background="?attr/colorPrimary"
  android:elevation="4dp"
  />
```

- Ajouter le code suivant dans la méthode OnCreate:

```
Toolbar myToolbar = (Toolbar) findViewById(R.id.my_toolbar);
  setSupportActionBar(myToolbar);
```

# Utilisation de ToolBar



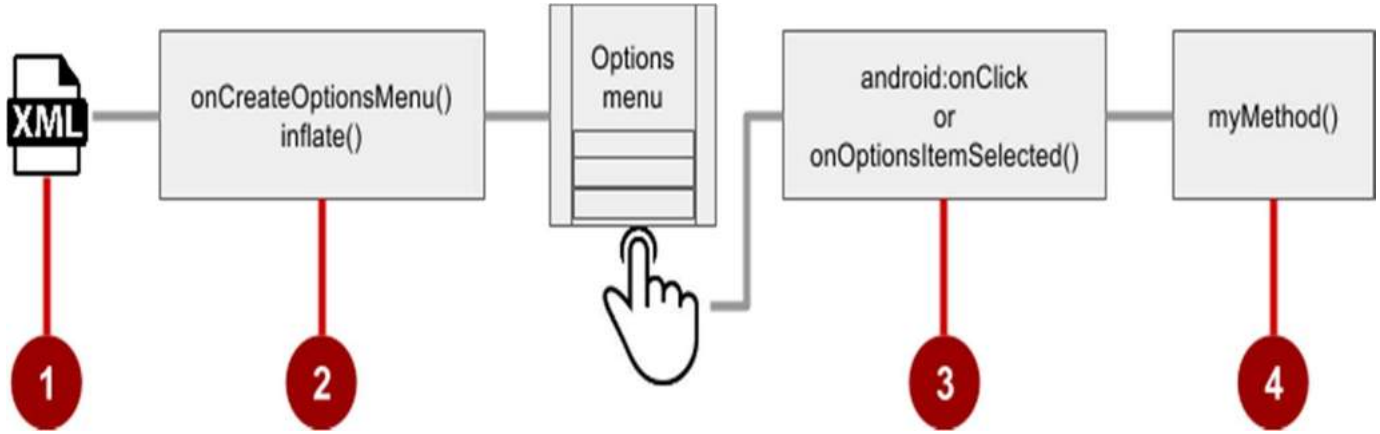
- Une fois ajouter comme barre d'application, on peut utiliser ToolBar en faisant appel à la méthode getsupportActionBar().

# Ajouter  Un menu à la barre d'app



Les étapes :

1. Creer une ressource  de type menu (`menu_main.xml`)
2. `Implémenter onCreateOptionsMenu()` pour coller le menu
3. `Implementer la méthode onClick` ou `onOptionsItemSelected()` pour gerer le clic



# Etape 1

1. Creer une resource de type menu dans un dossier Menu
2. Ajouter une balise "*item*" pour chaque action dans le menu par exemple (**Settings** and **Favorites**):



```
<item  android:id="@+id/option_settings"
       android:title="Settings" />
<item  android:id="@+id/option_favorites"
       android:title="Favorites" />
```

# Etape 1



1. On utilise l'attribue «app:showAsAction » pour spécifier si l'action doit apparaitre comme icone dans la barre (app:showAsAction="ifRoom" ),
2. ou dans le menu sous forme de liste (app:showAsAction="never" ).

# Etape 2

Re-ecrire le contenu de la méthode `onCreateOptionsMenu()` dans l'Activity qui affiche le menu.



```
@Override
public boolean onCreateOptionsMenu(Menu menu) {

  getMenuInflater().inflate(R.menu.menu_main, menu);
  return true;
}
```

- la fonction inflate récupere le fichier xml et le met dans l'objet Menu

# Etape 3

Afin d'implémenter l'action déclanchée lors de selectionner d'un item dans le menu, on doit écrire le code dans la fonction OnOptionsItemSelected, dans l'activity.
Par exemple :



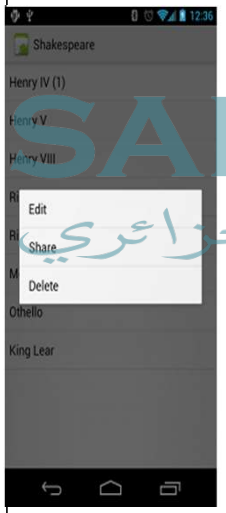
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
    case R.id.action_settings:
      showSettings();
      return true;
    case R.id.action_favorites:
      showFavorites();
      return true;
    default:
        return super.onOptionsItemSelected(item);
    }
}
```

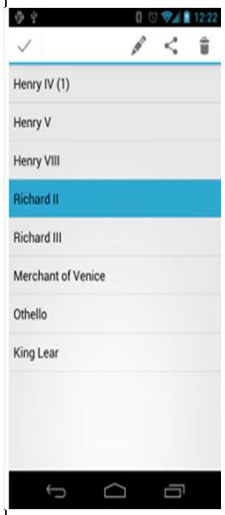
# Les menus contextuels?



- Permet aux utilisateurs d'effectuer une action sur la vue sélectionnée
- Peut être déployé sur n'importe quelle vue
- Le plus souvent utilisé pour les éléments dans RecyclerView, GridView ou autre collection View

# Types de menus contextuels



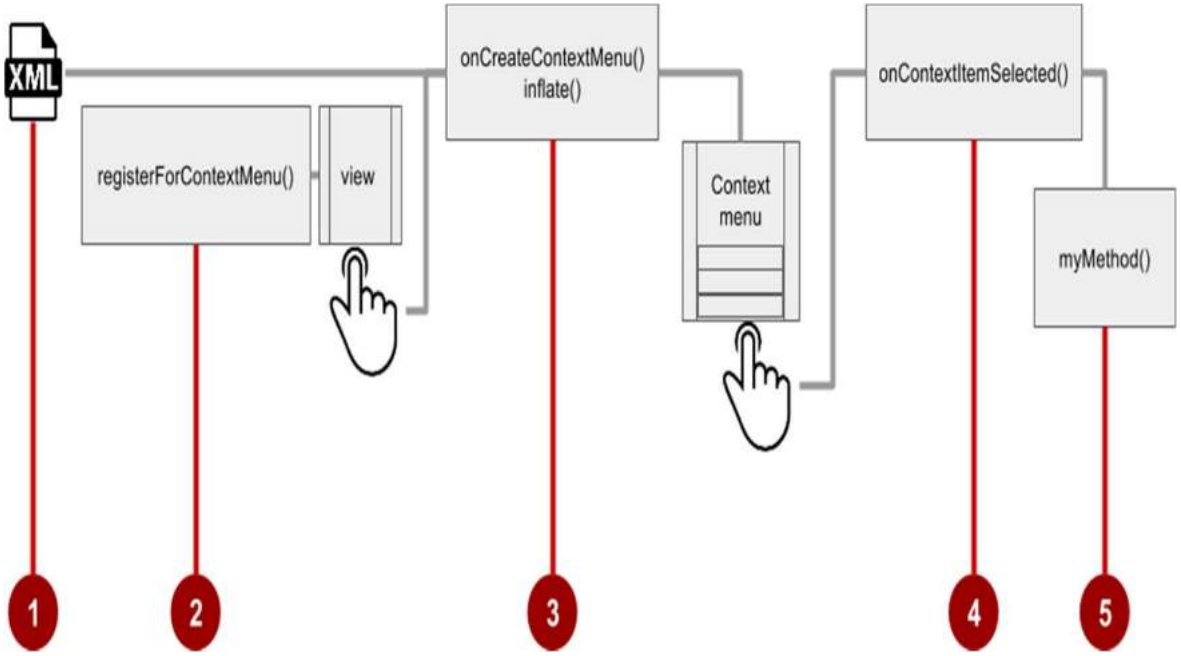



1. Menu contextuel flottant - appuyez longuement sur une vue
    a. L'utilisateur peut modifier la vue
    b. L'utilisateur effectue une action sur une vue à la fois
2. Mode d'action contextuelle: barre d'action temporaire à la place ou en dessous de la barre d'application
    a. Les éléments d'action affectent le ou les éléments View sélectionnés
    b. L'utilisateur peut effectuer une action sur plusieurs éléments de vue à la fois

# Menu contextuel flottant

1. Les étapes :





# Etapes



1. Créer un fichier de ressources de menu XML et attribuer des attributs d'apparence et de position
2. Enregistrer la vue à l'aide de registerForContextMenu ()
3. Implémentez onCreateContextMenu () dans Activity pour  lier ‘inflate’  le menu
4. Implémentez onContextItemSelected () pour gérer les clics sur les éléments de menu
5. Créer une méthode pour effectuer une action pour chaque élément du menu contextuel

# Etape 1

1. Créer un menu XML dans dossier res/menu

Par exemple : (menu_context.xml)



```
<item
   android:id="@+id/context_edit"
   android:title="Edit"
   android:orderInCategory="10"/>

<item
   android:id="@+id/context_share"
   android:title="Share"
   android:orderInCategory="20"/>
```

# Etape 2



Dans la méthode onCreate de l'activity :

2. Adherer la Vue (imageView,textView) a l'objet View.OnCreateContextMenuListener, Cela permet de spécifier la vue qui a va déclencher l'appel au menu

```
TextView article_text = findViewById(R.id.article);
registerForContextMenu(article_text);
```

# Etape 3



3. Spécifier le contextuel menu à afficher (selon la vue et le fichier XML).

- Dans le cas de plusieurs ContextMenu, on ajoute une condition sur le type de View (v).

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                    ContextMenu.ContextMenuInfo menuInfo) {
  super.onCreateContextMenu(menu, v, menuInfo);
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.menu_context, menu);
}
```

# Etape 4



- La méthode est appelée après chaque clique sur une action.
- On récupère l'ID de l'item sélectionné puis on implémente l'action nécessaire.

```
@Override
public boolean onContextItemSelected(MenuItem item) {
   switch (item.getItemId()) {
      case R.id.context_edit:
         editNote();
         return true;
      case R.id.context_share:
         shareNote();
         return true;
      default:
         return super.onContextItemSelected(item);
     }
}
```

# Mode d'action contextuelle



Mode d’action :

1. Mode d'interface utilisateur qui vous permet de remplacer temporairement des parties de l’interface graphique actuelle.
2. Par exemple: sélectionner une section de texte ou appuyer longuement sur un élément peut déclencher le mode action (une barre pour copier, coller, partager ….)
3. Ce mode est une implémentation système d'ActionMode qui affiche une barre d'action contextuelle en haut de l'écran avec des éléments d'action qui affectent le ou les éléments sélectionnés. Lorsque ce mode est activé, les utilisateurs peuvent effectuer une action sur plusieurs éléments à la fois (si votre application le permet).

# Cycle de vie du mode action

- Démarre avec la méthode startActionMode(),
- ActionMode.Callback elle implement les méthodes de cycle de vie qu'on doit modifier:
  - `onCreateActionMode(ActionMode, Menu)`: pour initialiser la création
  - `onPrepareActionMode(ActionMode, Menu)`: apres la création et tant que le ActionMode est invalide ou nulle
  - `onActionItemClicked(ActionMode, MenuItem)` apres chaque clique sur une action
  - `onDestroyActionMode(ActionMode)` la fermeture du mode d'action

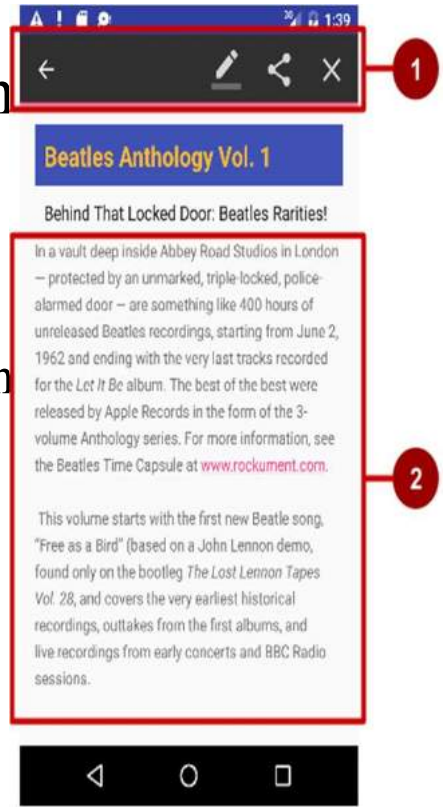
# Barre d’action Contextuelle

Un longue appui sur  les vue fait apparaite la barre d’action contextuelle (contextual action bar)

1- La barre d’action contextuelle avec les action

- **Modifier**, **Partager**, et **Supprimer**
- Valider
- La barre d’action restera visibile jusqu’a la validation

2- La vue sur laquelle, le longue appui a été appliqué



# Etapes

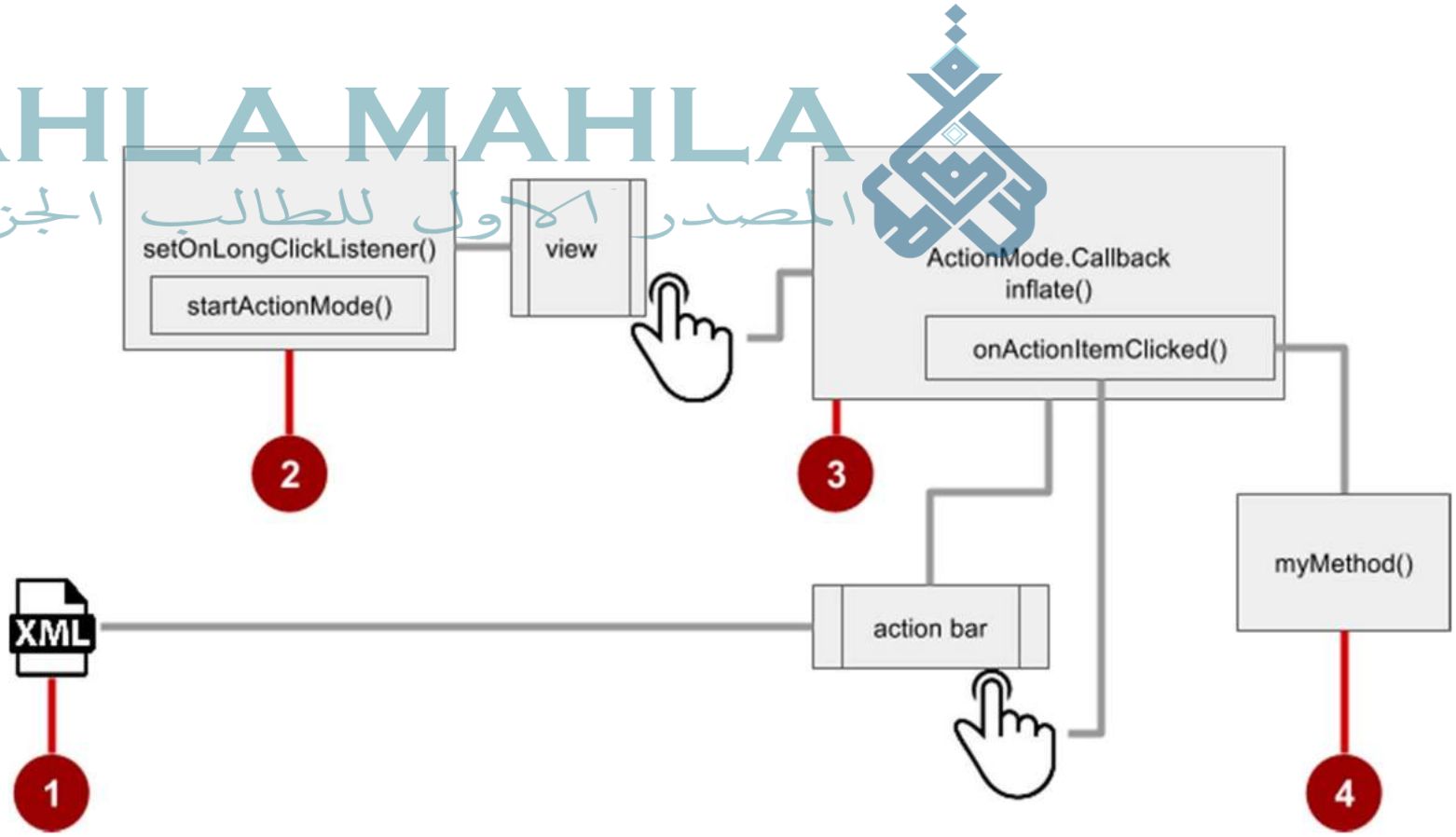


1. Creer un fichier XML de type menu dans resource, et assigner des icons pour chaque iteme
2. `Implementer la méthode setOnLongClickListener()` de la vue ciblée afin de lancer la barre d'action , a l'interiieur on fait appel a la méthode `startActionMode().`

3. Implement er la méthode `ActionMode.Callback` interface pout gerer cycle de vie de ActionMode ensuite ajouter des actions pour tous les items en utilisant `onActionItemClicked()` callback
4. Creer une methode pour effectuer les actions.

# Etapes





# Etape 1

1. Créer un menu XML dans dossier res/menu de préférence avec une icône pour chaque item.

Par exemple : (`menu_context.xml`)

```
<item
    android:id="@+id/ajouter"
    android:icon="@drawable/ic_add"
    app:showAsAction="ifRoom"
    android:title="ajouter" />

<item
    android:id="@+id/partager"
    android:icon="@drawable/ic_share"
    app:showAsAction="ifRoom"
    android:title="Partager" />
```

# Etape 2

- Dans l'activity classe, déclarer l'attribue ActionMode,
- Récuperer la vue qui déclenche l'affichage de menu contextuel.
- Implémenter la méthode SetonLongClickListener
- Vérifier si mActionmode est déjà instanciée
- Lancer Actionmode avec la fonction StartActionMode



```
private ActionMode mActionMode;
```

In onCreate():

```
View view = findViewById(article);
view.setOnLongClickListener(new View.OnLongClickListener() {
   public boolean onLongClick(View view) {
      if (mActionMode != null) return false;
      mActionMode =
            MainActivity.this.startActionMode(mActionModeCallback);
      view.setSelected(true);
      return true;
   }
});
```

# Etape 3



Dans la classe Activity, Implémenter l'objet Callback qui permet de gérer cycle de vie de l'actionMode.

```
public ActionMode.Callback mActionModeCallback =
    new ActionMode.Callback() {
     // Implement action mode callbacks here.
};
```

# Etape 3-1

Lier la ressource XML à l'objet Menu lors de la création



```
@Override
public boolean onCreateActionMode(ActionMode mode, Menu menu) {
   MenuInflater inflater = mode.getMenuInflater();
   inflater.inflate(R.menu.menu_context, menu);
   return true;
}
```

# Etape 3-2

- Cette méthode est Appelée a chaque fois que l'action mode est activé.
- Toujours apres `onCreateActionMode`,



```
@Override
public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
   return false; // Return false if nothing is done.
}
```

# Etape 3-3

- Appelée une fois l'utilisateur choisi une action.
- Les instructions a réaliser seront implemnter dans cette méthode.



```
@Override
public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
   switch (item.getItemId()) {
       case R.id.action_share:
         // Perform action for the Share menu item.
         mode.finish(); // Action picked, so close the action bar.
         return true;
       default:
         return false;
   }
}
```

# Etape 3-4

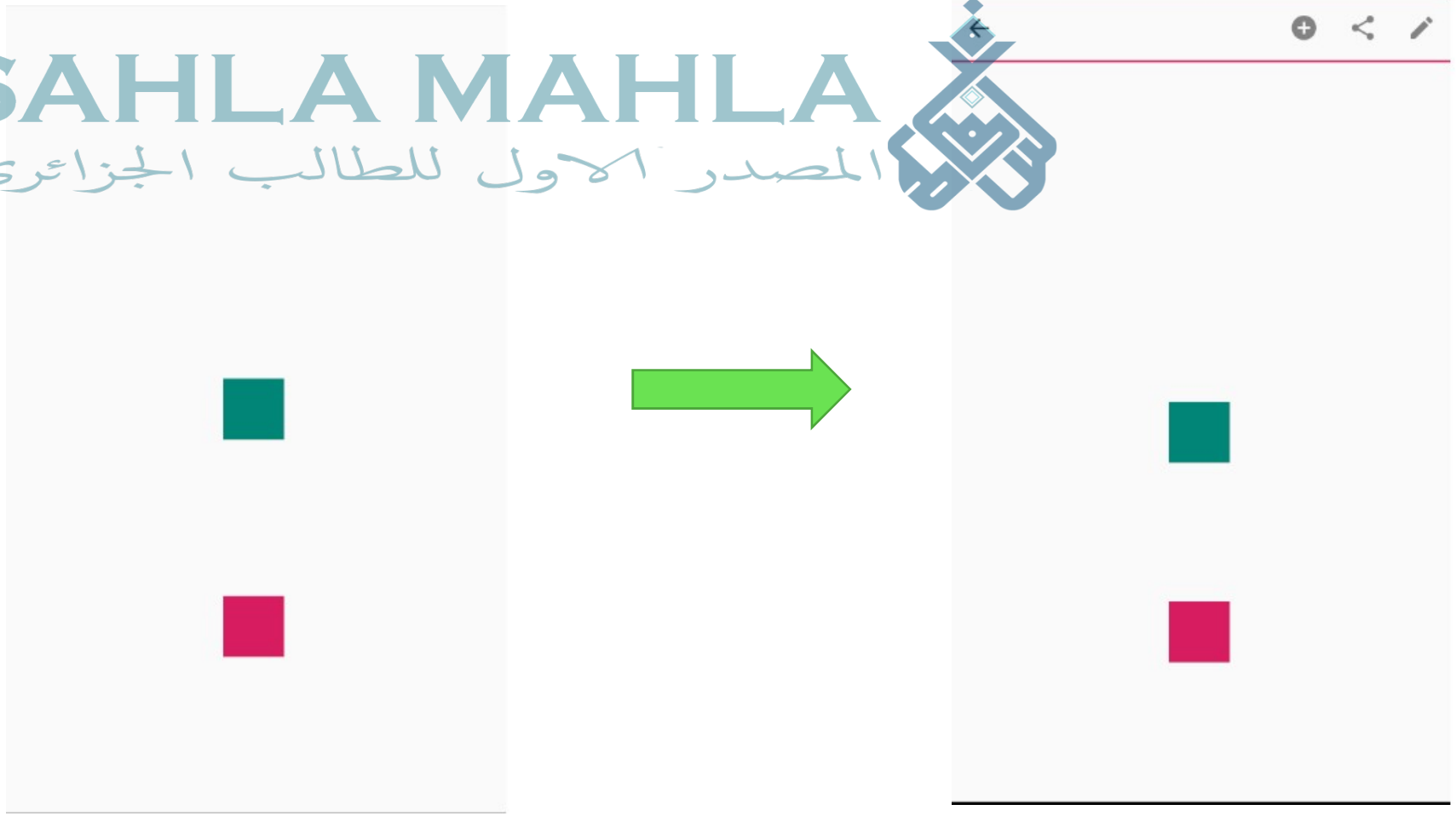


- Applée lorque l'utilisatuer quitte le action mode.
- Il suffit de mettre l'action mode a null.

```
@Override
public void onDestroyActionMode(ActionMode mode) {
   mActionMode = null;
}
```

# Résultat



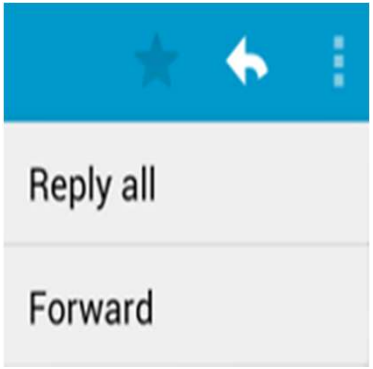


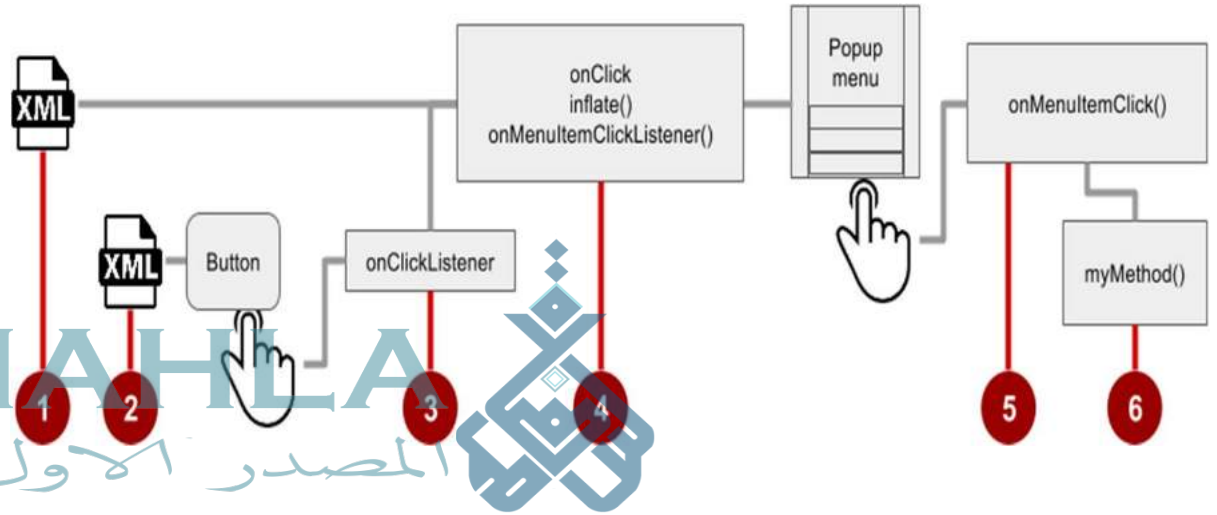
# Popup Menu



- Une Liste Verticale d'itemes liées a une vue
  - Généralement ancré à une icone visible
- Les actions n'affectent pas directement le contenu de

La vue. Par exemple, pour transferer un email. Le menu est lié a l'objet email, et non pas le contenu (le message)








1. Creer fichier XML dans res/menu
2. Ajouter `ImageButton` dans layout Activity pour faire apparaitre le popup menu.
3. Implementer `onClickListener` de composant `ImageButton (dans Activity)`
4. A l'interieur de `onClick()` implementer `onMenuItemClickListener()`
5. Implementer `onMenuItemClick()`
6. Creer des méthodes pour faire executer les actions

# Etape 2



Par exemple :

```
<ImageButton
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/button_popup"
 android:src="@drawable/@drawable/ic_action_popup"/>
```

# Etape 3



```
private ImageButton mButton =
    (ImageButton) findViewById(R.id.button_popup);
```

Dans onCreate():

```
mButton.setOnClickListener(new View.OnClickListener() {
    // define onClick
});
```

# Etape 4

```
@Override
public void onClick(View v) {
    PopupMenu popup = new PopupMenu(MainActivity.this, mButton);
    popup.getMenuInflater().inflate(
        R.menu.menu_popup, popup.getMenu());
    popup.setOnMenuItemClickListener(
        new PopupMenu.OnMenuItemClickListener() {
            // implement click listener.
         });
    popup.show();
}
```

# Etape 5



```
public boolean onMenuItemClick(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.option_forward:
            // Implement code for Forward button.
            return true;
        default:
            return false;
    }
}
```