Computação II – Orientação a Objetos

Fabio Mascarenhas - 2014.1

http://www.dcc.ufrj.br/~fabiom/java

Inflando layouts

- Se quisermos um layout mais complexo nas linhas de uma `ListView`, podemos criar objetos layout e adicionar objetos view a ele

- Ou podemos declarar um layout em XML e *inflar* esse layout

- Para isso usamos o *serviço* `LAYOUT_INFLATER_SERVICE`, uma instância de LayoutInflater:

```
LayoutInflater inf =
    (LayoutInflater)lv.getContext().getSystemService(
        Context.LAYOUT_INFLATER_SERVICE);
view = inf.inflate(R.layout.item_lv, null);
```

Menu de Opções

- É comum que as atividades de uma aplicação Android tenham um *menu de opções*

- Na nossa aplicação de busca, por exemplo, vamos por alguns termos de busca comuns

- Começamos editando o XML do menu, acrescentando blocos item:

```
<item
    android:id="@+id/busca_java"
    android:orderInCategory="100"
    android:showAsAction="never"
    android:title="Java"/>
```

Ações e exibição

- Se quisermos que uma opção apareça como uma ação na barra de título então mudamos o atributo `showAsAction` para *"ifRoom"* ao invés de *"never"*

- Para exibir o menu, redefinimos o método onCreateOptionsMenu da atividade, e então *inflamos* o menu:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.buscas, menu);
    return true;
}
```

Respondendo a cliques

- Quando o usuário toca em um item do menu, ou em uma ação, o sistema chama o método `onOptionsItemSelected`, basta redefini-lo:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.busca_java:
            busca("java");
            return true;
        case R.id.busca_ufrj:
            busca("ufrj");
            return true;
        default:
            return super.onOptionsItemSelected(item);
        }
    }
}
```

Menus dinâmicos

- E se quisermos mostrar no menu as cinco últimas frases buscadas?

- É fácil guardar uma lista de termos de busca, mas como fazemos o menu mudar junto? Implementando o método `onPrepareOptionsMenu` ao invés de `onCreateOptionsMenu`:

```
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    menu.clear();
    menu.add(0, R.id.busca_java, 1, "Java");
    menu.add(0, R.id.busca_ufrj, 2, "UFRJ");
    return true;
}
```

- Podemos usar quaisquer números no segundo parâmetro

Editor de Figuras

- Vamos usar nosso modelo de editor de figuras para fazer uma versão Android dele

- Ao invés de usarmos botões, vamos usar toda a área da aplicação como área de desenho, e usar ações e itens de menu para controlar o modo atual do editor

- Vamos implementar uma view para ser o Canvas do editor, e conectá-lo com componentes Android implementando as outras interfaces do modelo

- Nosso canvas também vai capturar os eventos de clique e arrasto, e passá-los para um controlador que vai traduzi-lo nos métodos do modelo

Canvas

- Para poder desenhar em uma `View`, redefinimos seu método `onDraw`
- Esse método recebe uma instância de `Canvas`
- Para desenhar também de uma instância de `Paint`, que dá a cor e o estilo de desenho (no nosso caso, queremos que as figuras sejam preenchidas)

```
pen = new Paint();
pen.setARGB(255,0,0,0);
pen.setStyle(Style.FILL);
```

```
@Override
public void onDraw(Canvas c) {
  super.onDraw(c);
  c.drawRGB(255,255,255);
  c.drawRect(10, 10, 100, 100, pen);
}
```

onTouchEvent

- Para capturar toques na nossa área de desenho, redefinimos o método `onTouchEvent`, que recebe uma instância de `MotionEvent`
- Estamos interessados em três partes do evento: a ação, a coordenada `x` e a coordenada y
- A ação diz se um toque começou (o usuário encostou o dedo na tela), se um toque terminou (o usuário removeu o dedo), ou se ele se moveu (o usuário deslizou o dedo sobre a tela)
- Vamos traduzir isso em eventos aperto, solta e arrasto no nosso controlador

Checked menus

- Para seleção do modo de edição, vamos usar *checked menus* no menu da barra de aplicação

- Esses menus têm uma checkbox ao lado; a checkbox do modo atual aparecerá ticada

```
<item android:id="@+id/modo_mover"
      android:showAsAction="never"
      android:checkable="true"
      android:checked="true"
      android:title="Mover"/>
```

- O estado de cada item virá de objetos que implementam `Toggle`, e ficam conectados ao modelo