09

Para saber mais - Outras implementações de callback

O callback `ItemTouchHelper.Callback` é uma implementação primária para criar um callback para o `ItemTouchHelper`, mas existem outras classes que possibilitam a implementação do callback também. É o caso da `ItemTouchHelper.SimpleCallback` (https://developer.android.com/reference/android/support/v7/widget/helper/ItemTouchHelper.SimpleCallback.html), que é uma subclasse do `ItemTouchHelper.Callback` e tem o objetivo de evitar a implementação do método `getMovementFlags()`.

Em uma amostra de código, poderíamos ter o mesmo resultado da seguinte maneira com o `SimpleCallback`:

```
new ItemTouchHelper.SimpleCallback(
    ItemTouchHelper.RIGHT | ItemTouchHelper.LEFT,
    ItemTouchHelper.DOWN | ItemTouchHelper.UP
            | ItemTouchHelper.RIGHT | ItemTouchHelper.LEFT) {
@Override
public boolean onMove(RecyclerView recyclerView, RecyclerView.ViewHolder viewHolder, RecyclerView.Vi
    int posicaoInicial = viewHolder.getAdapterPosition();
    int posicaoFinal = target.getAdapterPosition();
    trocaNotas(posicaoInicial, posicaoFinal);
    return true;
}

private void trocaNotas(int posicaoInicial, int posicaoFinal) {
    new NotaDAO().troca(posicaoInicial, posicaoFinal);
    adapter.troca(posicaoInicial, posicaoFinal);
}

@Override
public void onSwiped(RecyclerView.ViewHolder viewHolder, int direction) {
    int posicaoDaNotaDeslizada = viewHolder.getAdapterPosition();
    removeNota(posicaoDaNotaDeslizada);
}

private void removeNota(int posicao) {
    new NotaDAO().remove(posicao);
    adapter.remove(posicao);
}
};
```

Sim, é uma implementação no estilo de classe anônima assim como fazemos com as interfaces.

Note que a diferença é que enviamos as flags para arrastar e deslizar os elementos sucessivamente direto no construtor.