Computação II – Orientação a Objetos

Fabio Mascarenhas - 2014.1

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

Calculadora Android

- Como exemplo de uma aplicação Android um pouco mais complexa que *Hello, World*, vamos construir uma calculadora como a do laboratório

- O *modelo* que vamos usar é exatamente o mesmo, já que ele não depende de nenhuma classe que não está presente no sistema Android

- Para a *visão* e o *controlador*, vamos usar os componentes embutidos do sistema Android: atividades, listeners e views

Layout de Grade

- Nossa calculadora tem um display e dezesseis botões organizados em fileiras de quatro
- Uma maneira natural de organizar esses componentes é como uma grade de quatro colunas e cinco linhas, onde o display ocupa toda a primeira linha
- O sistema Android tem um layout perfeito para isso: `GridLayout`
- Com um `GridLayout` podemos dispor componentes em qualquer ponto da grade, e fazer eles ocuparem múltiplas linhas e/ou colunas
- Também podemos controlar onde o componente fica dentro da célula que ocupa

Display

- Para o display, vamos usar uma `TextView`
- Não vamos usar um recurso para o texto do `TextView`, já que seu conteúdo vai mudar dinamicamente
- Para conectar a `TextView` com o modelo, precisamos de uma instância de `ObservadorDisplay`
- Usamos findViewById para obter a instância de TextView do display

Botões e Cliques

- Um clique (toque) no botão pode ser observado por um OnClickListener
- Usamos findViewById para cada botão, e registramos um listener para ele
- Podemos usar um listener diferente para cada botão
- Uma simplificação é reaproveitar o mesmo listener para os dígitos: o listener recebe o componente que foi clicado, podemos pegar o texto do botão e converter para um número para ter o dígito
- Uma vez que conectamos cada botão a um listener, a calculadora está funcionando, mas ainda não está completa

Estado da Aplicação

- Em um computador, aplicações ficam abertas até o usuário fechá-las explicitamente
- No sistema Android, se uma aplicação não está mais em primeiro plano, o sistema pode decidir fechá-la para economizar memória
- Se o usuário volta para a aplicação, é como se ela estivesse sendo iniciada de novo
- Na nossa calculadora, isso implica que perdemos todo o estado atual dela

Salvando o Estado

- Quando o sistema fecha uma aplicação, ele dá uma chance para ela guardar toda a informação necessária para recriar o estado onde estava
- Isso é feito pelo método `onSaveInstanceState` de `Activity`
- Esse método recebe um objeto `Bundle`, onde a informação necessária deve ser armazenada
- Esse objeto é passado depois para o método `onCreate`, quando a aplicação é reiniciada
- Podemos guardar nele informação para recriar o modelo da calculadora

Serialização

- Podemos recriar o estado da calculadora só a partir de dados primitivos, com alguma “mágica”
  - `Class.forName(“Nome”).getConstructor(...).newInstance(...)`
- Uma alternativa é *serializar* todo o modelo, empacotando todos os seus objetos em uma representação que permite depois reconstruí-los
- Para marcar um objeto como serializável, precisamos implementar a interface `Serializable`
- Essa interface não tem métodos; o trabalho todo é feito pelo sistema Java

Externalizable

- Para os objetos de uma classe serem serializáveis, todos os seus campos devem ser também
- Não podemos tornar todo o modelo serializável, pois não podemos serializar o `ObservadorDisplay`
- `Externalizable` é uma subinterface se `Serializable` onde podemos controlar o que é serializado
- `Externalizable` tem dois métodos que permitem serializar e desserializar as partes do objeto que queremos
- No caso do modelo, é tudo exceto o observador