Computação II – Orientação a Objetos

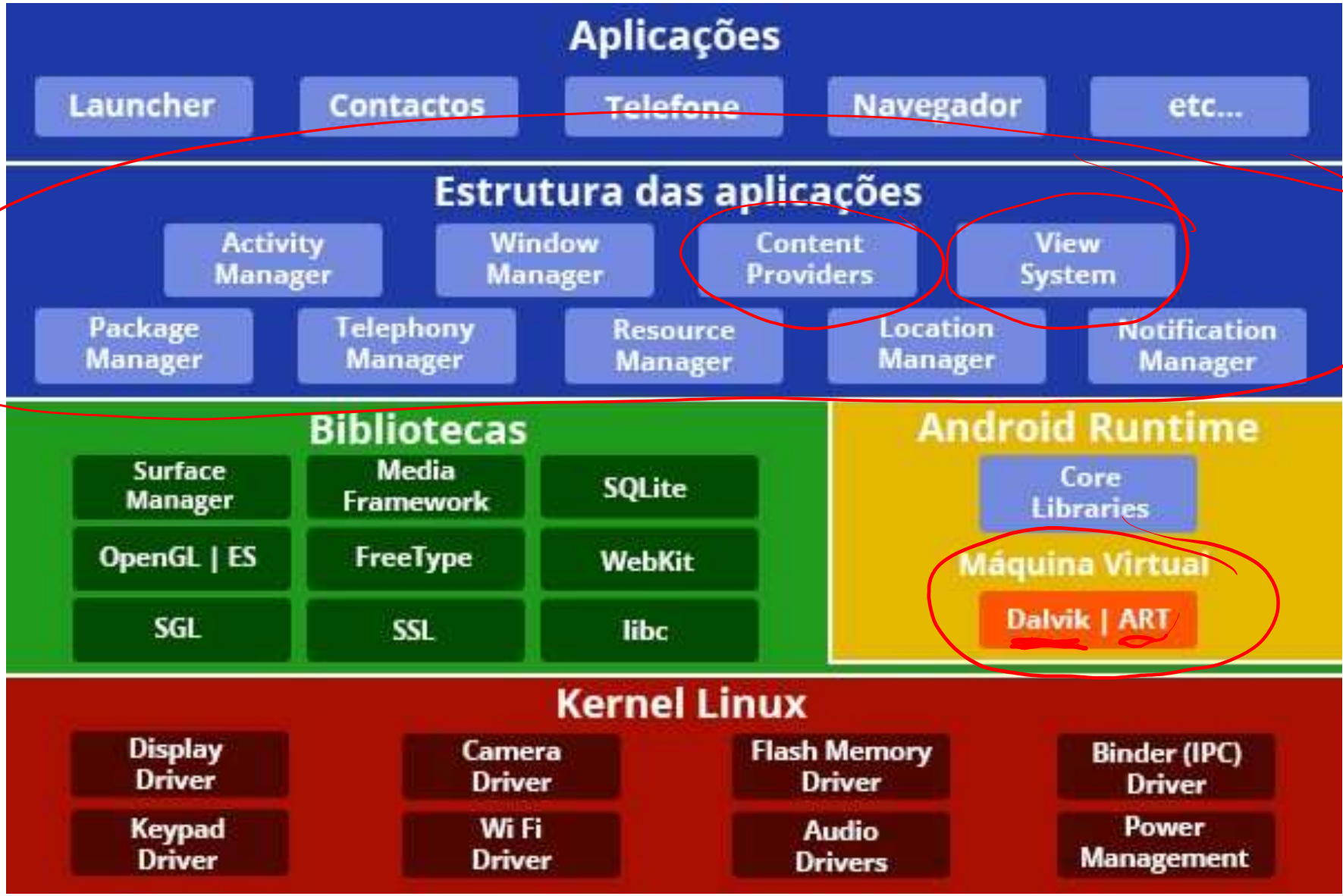
Fabio Mascarenhas - 2016.2

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

Android

- Android é um sistema operacional para dispositivos móveis
  - *Kernel* Linux, drivers e bibliotecas do sistema, *frameworks* de aplicação (Android Software Development Kit, Android Native Development Kit) e aplicações embutidas
- Aqui estaremos interessados no Android SDK, um framework para desenvolvimento de aplicações para Android na linguagem Java
- O sistema Android possui farta documentação

Arquitetura do SO Android



Java no Android

- O sistema Android não usa a implementação oficial de Java, mas a sua própria, um pouco atrás da versão 1.8 (a não ser na próxima versão do Android, a versão N)

- A maior parte das classes nos pacotes `java.*` e `javax.*` estão presentes

- Classes específicas do sistema Android estão nos pacotes `android.*`, e classes úteis para aplicações Android estão nos pacotes `org.*`

- Outras bibliotecas Java geralmente funcionam sem modificações, mas às vezes dependem de partes que não estão no Android, mesmo que possuam equivalentes

Android Studio

- O Android Studio é um ambiente de desenvolvimento para Android que junta todo o necessário para desenvolver em Android:
  - O Android SDK
  - O *emulador* Android e uma *imagem* para o emulador
  - Um ambiente integrado de desenvolvimento
- Baixe o instalador Windows nesse link (cerca de 1Gb!)

Hello, Android

- Vamos criar e rodar uma aplicação
- Uma aplicação Android já vem com muita estrutura, pois ela está inserida em um framework bastante complexo
- Ela já está separada em um *controlador* (a classe principal) e uma *visão* (descrita em arquivos XML), e mesmo algumas partes que formam um *modelo* bem simples (no arquivo `strings.xml`)
- Para rodar essa aplicação, primeiro precisamos criar um *dispositivo virtual* no emulador (ou conectar um dispositivo real via USB)

Componentes de uma App Android

- Atividades (*Activities*)
- Serviços (*Services*)
- Comunicadores (*Broadcastreceivers*)
- Provedores de Conteúdo (*Content Providers*)
- Recursos (*Resources*)
- O sistema Android instancia os componentes sob demanda, e cada um deles tem seu propósito e API

Recursos

- Recursos são metadados usados pela aplicação
  - Strings, layouts, imagens, menus, animações...
- Poderiam ser todos instanciados por código Java, mas usar arquivos de recursos deixa as aplicações mais flexíveis e configuráveis
- Normalmente recursos são descritos em arquivos XML
- Exemplo: internacionalização

XML

- Um formato para descrever dados
- Um arquivo XML é um conjunto de blocos, onde cada bloco é delimitado por uma *tag de abertura* e uma *tag de fechamento*
- Uma *tag de abertura* é composta por <, o *nome da tag*, os *atributos*, e >, enquanto uma *tag de fechamento* é </, o nome da tag, e >
- Cada *atributo* é composto do *nome* do atributo, o operador =, e o *valor* do atributo

Strings

- Strings são tipicamente usadas para configurar o idioma da aplicação

- Descritas em um arquivo strings.xml, com blocos do tipo:

```
<string name="hello_world">Olá, Android!</string>
```

- Podem ter *tags* `<b>`, `<i>` e `<u>` para formatação

- Referenciadas por outros recursos com `@string/hello_world` ou em código Java por `R.string.hello_world`

Layouts

- Layouts são descrições dos componentes visuais que formam as telas da aplicação

- Eles estão arquivos XML em `res/layout`, e contêm um bloco XML com uma das tags de layout possíveis; dentro desse bloco colocamos outros blocos com tags dos componentes que queremos

- No HelloAndroid, o layout está nos arquivos `activity_hello.xml` e `content_hello.xml`, e podemos referenciar esses layouts com `@layout/content_hello` em um recurso, ou `R.layout.content_hello` em código Java

`LinearLayout` e `CoordinatorLayout`

- `LinearLayout` é o tipo de layout mais simples
- Distribui seus componentes em sequência (por isso o nome), ou na vertical ou na horizontal (atributo `android:orientation`)
- Pode conter outros LinearLayouts!
- `CoordinatorLayout` é o tipo *default* de layout padrão, e permite que partes dele afetem outras partes (veja esse post)
- Layouts são componentes (*views*) como quaisquer outros

AndroidManifest

- O *manifesto da aplicação* descreve os atributos gerais da aplicação:
  - Nome, versão e ícone
  - Atividade principal da aplicação, e seus outros componentes
  - Versão mínima do Android que ela precisa
  - Permissões (câmera, internet, contatos, etc.)

Atividades

- Uma atividade é uma subclasse da classe abstrata `Activity`, e representa uma ação do usuário, e sua interface gráfica

- O método principal que uma atividade deve implementar é o onCreate, onde ela:

  - Restaura o seu *estado*

  - Diz qual a *visão* que ela vai exibir

  - Inicializa os componentes da interface, e liga seus eventos a ações

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 Tabela

- Nossa calculadora tem um display e dezesseis botões organizados em fileiras de quatro
- Uma maneira natural de organizar esses componentes é como uma tabela de quatro colunas e cinco linhas, onde o display ocupa toda a primeira linha
- O sistema Android tem um layout perfeito para isso: `TableLayout`
- Cada linha da tabela é um componente `TableRow`
- É mais fácil editar diretamente o XML do que usar o designer

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