

Linguagem Técnica de Programação Mobile

AULA 1 – Revisão de POO e Visão Geral sobre o Google Android

Prof. João Paulo Pimentel
joao.pimentel@projecao.br



Roteiro da Aula



- Conceitos de Programação Orientada a Objetos
 - Classes e Objetos
 - Herança e Polimorfismo
 - Encapsulamento
 - Modificadores de acesso
 - Abstração
 - Visão Geral sobre o Google Android
 - Estrutura
 - Arquitetura
 - Aplicações
 - Android Runtime
 - Linux Kernel
-

Programação orientada a objetos

- Princípios da programação orientada a objetos
 - A programação orientada a objetos (POO) consiste em uma metodologia de desenvolvimento de software para gerenciamento de problemas cada vez mais complexos que não poderiam ser solucionados com antigas técnicas com foco meramente na codificação do software (ECKEL, 2006).
 - Os princípios básicos da POO são descritos na figura seguinte no próximo slide:
-

Programação orientada a objetos

Classe

- Representação de um conjunto de objetos com características afins. Definição do comportamento dos objetos (métodos) e seus atributos (atributos).

Objeto

- Uma instância de uma classe.
- Armazenamento de estados através de seus atributos e reação a mensagens enviadas por outros objetos.

Herança

- Mecanismo pela qual uma classe (sub-classe) pode estender outra classe (super-classe), estendendo seus comportamentos e atributos.

Polimorfismo

- Princípio pelo qual as instâncias de duas classes ou mais classes derivadas de uma mesma super-classe podem invocar métodos com a mesma assinatura, mas com comportamentos distintos.

Encapsulamento

- Proibição do acesso direto ao estado de um objeto, disponibilizando apenas métodos que alterem esses estados na interface pública.

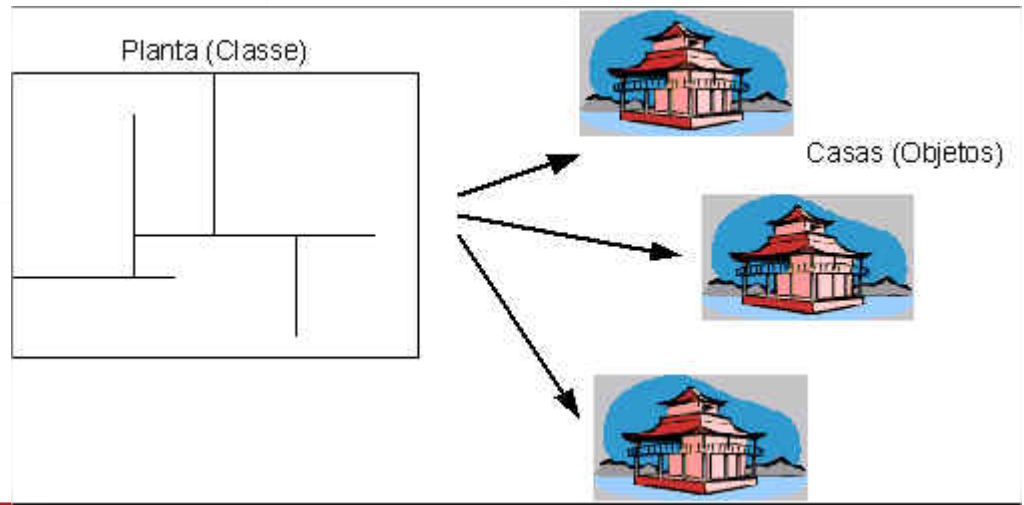
Classes e Objetos



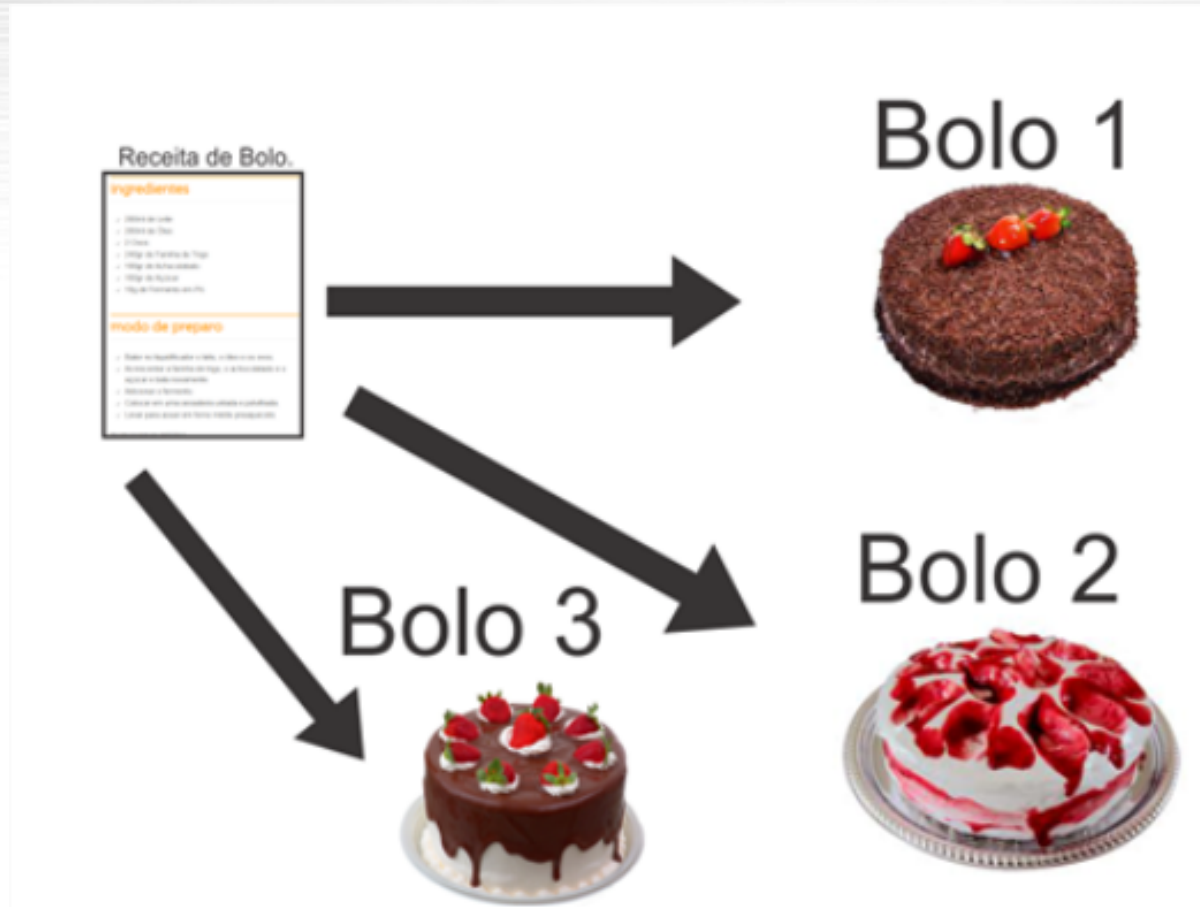
= Objeto



= Classe




Classes e Objetos



Praticando... 17 minutos...

- Como seria uma classe de uma conta corrente?
 - No Eclipse, Netbeans, Bloco de Notas, Notepad++ ou mesmo no caderno escreva uma classe **ContaCorrente** cujos atributos são **saldo** e **chequeEspecial**.
 - Depois, escreva uma classe chamada **Loja**. Toda loja tem uma **conta corrente** (utilizar herança).
 - **Main (JavaApplicationBanco)**.
 - Como seria uma implementação com encapsulamento?
-

Desenvolvendo no Netbeans IDE 8.2



Novo Aplicação Java

Etapas

- Escolher Projeto
- Nome e Localização**

Nome e Localização

Nome do Projeto:

JavaApplicationBanco

Localização do Projeto:

C:\Users\JoãoPaulo\Documents\NetBeansProjects

Procurar...

Pasta do Projeto:

Paulo\Documents\NetBeansProjects\JavaApplicationBanco

☐ Usar Pasta Dedicada para Armazenar Bibliotecas

Pasta Bibliotecas:

Procurar...

Usuários e projetos diferentes podem compartilhar as mesmas bibliotecas de compilação (consulte a Ajuda para obter detalhes).

☒ Criar Classe Principal

javaapplicationbanco.JavaApplicationBanco

< Voltar

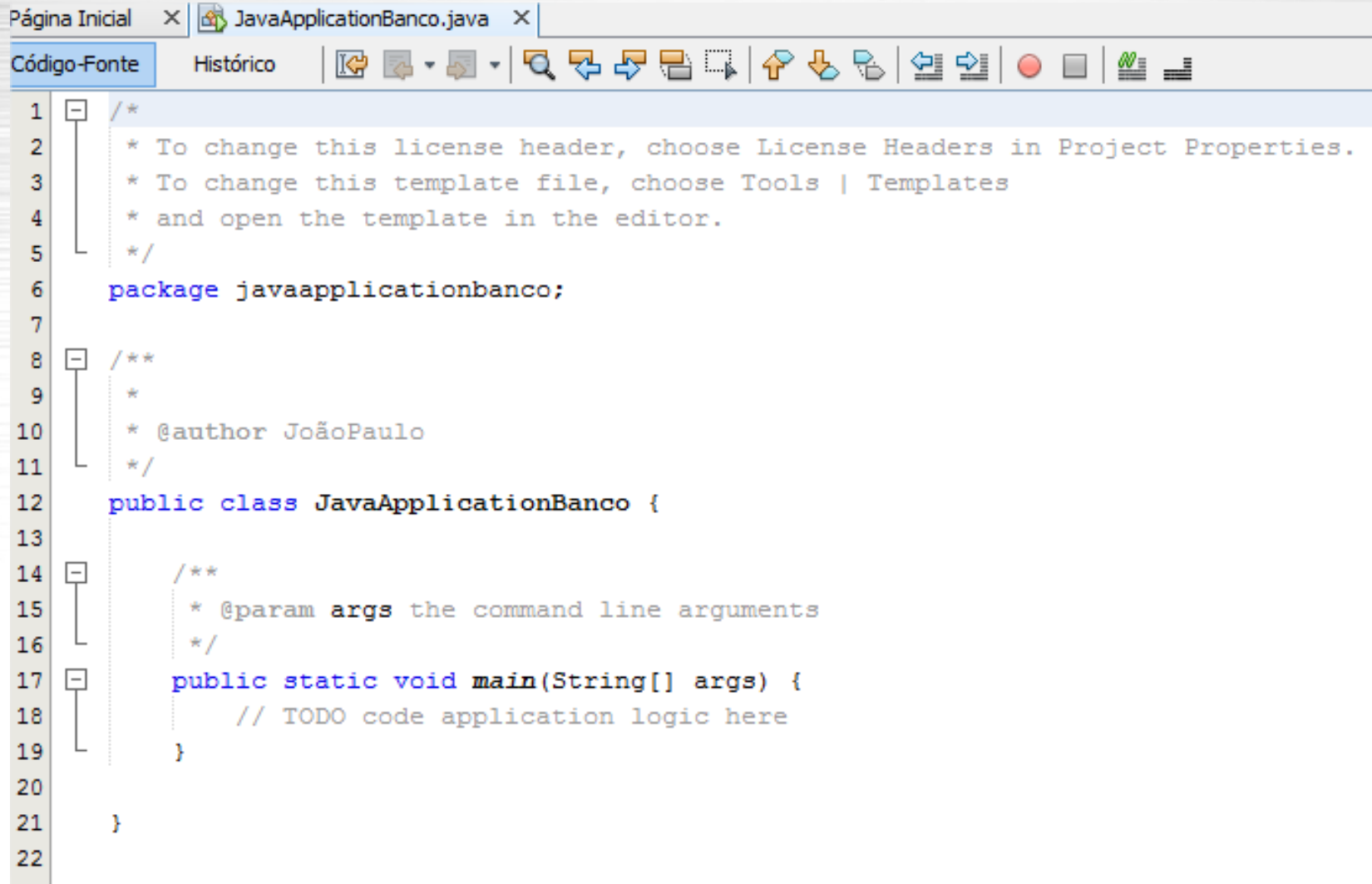
Próximo >

Finalizar

Cancelar

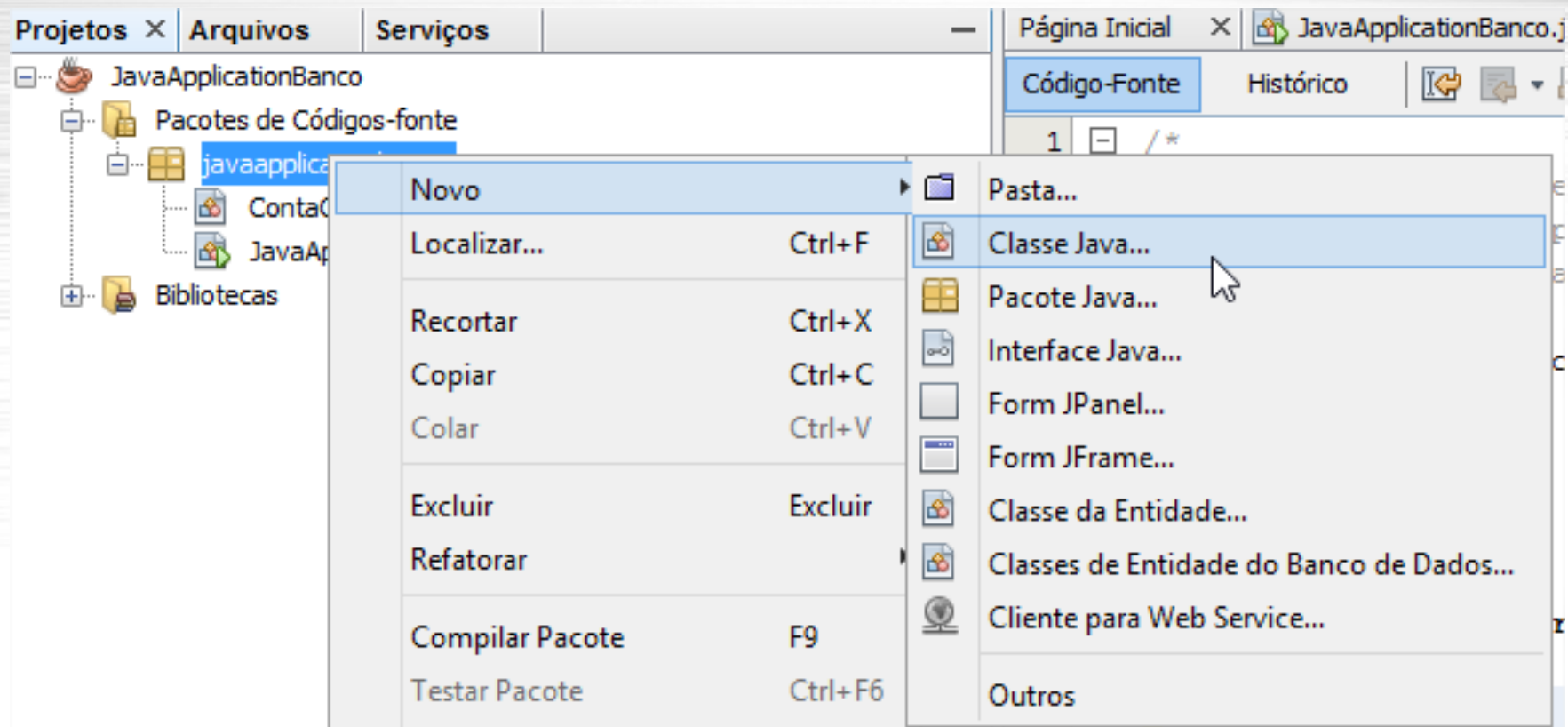
Ajuda

Desenvolvendo no Netbeans IDE 8.2

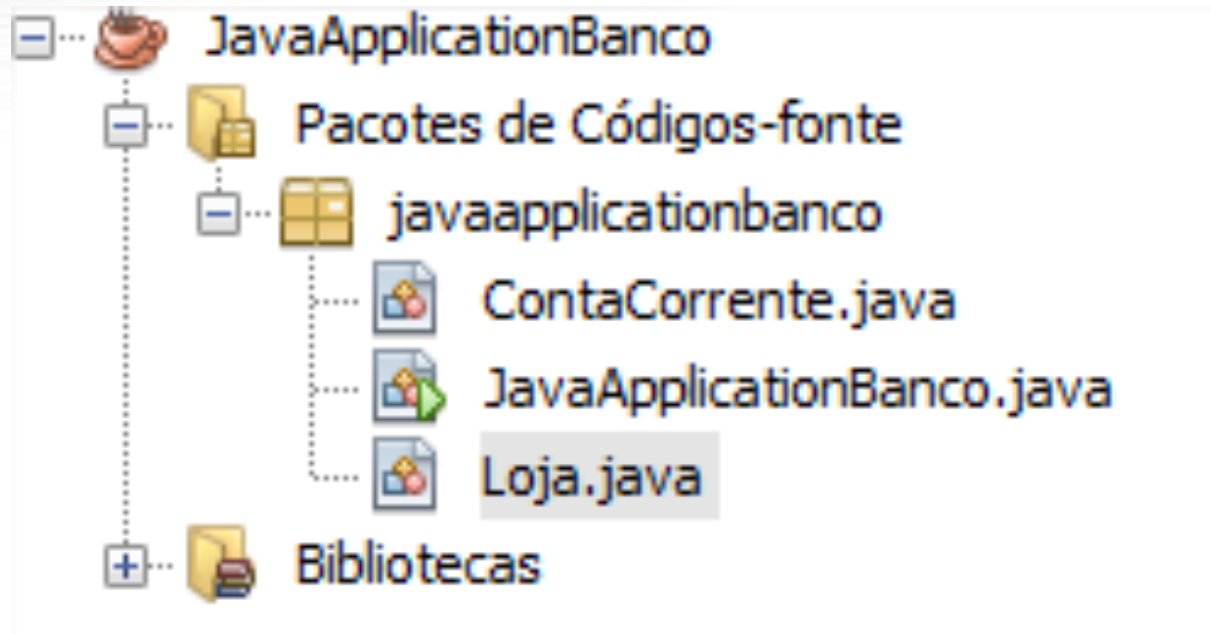


```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package javaapplicationbanco;
7
8  /**
9   *
10   * @author JoãoPaulo
11   */
12  public class JavaApplicationBanco {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19      }
20
21  }
22
```

Desenvolvendo no Netbeans IDE 8.2



Desenvolvendo no Netbeans IDE 8.2



Desenvolvendo no Netbeans IDE 8.2

Vejamos um “Mau Exemplo”. Por quê?????

```
class ContaCorrente{                                     //Mau exemplo
    public int saldo;
    public int chequeEspecial;

    public ContaCorrente(int saldo, int chequeEspecial){
        this.saldo = saldo;
        this.chequeEspecial = chequeEspecial;
    }
}

class Loja{
    public ContaCorrente minhaConta;

    public Loja(ContaCorrente minhaConta){
        this.minhaConta = minhaConta;
    }
}

class Main{
    public static void main(String [] args){
        ContaCorrente c1 = new ContaCorrente(500, 100);
        // ...
    }
}
```

Falando sobre o exemplo anterior...

- O que poderia acontecer com o exemplo anterior?
 - As lojas poderiam não ter tomado o cuidado suficiente ao manipular a variável saldo de seus clientes...
 - Toda responsabilidade fica sobre os programadores das Lojas que vão utilizar esta classe. A classe está **desprotegida**!
 - Eventualmente alguma outra classe (programada por um desenvolvedor descuidado) poderia debitar de uma conta mais do que ela realmente tem em saldo...
-

Encapsulamento

- O que é encapsulamento?

saldo

cheque especial



Encapsulamento

- Mas como conseguimos encapsulamento?
 - Mantenha suas variáveis de instância protegida (modificador de acesso **private**)
 - Crie métodos de acesso públicos (getters).
 - Mas quem são estes Modificadores de Acessos?
 - public
 - private
 - protected
 - default ou package-private (sem modificador explícito)
-

Encapsulamento

- **Modificadores de acesso :**
 - **public**
 - acesso global;
 - **private**
 - acesso restrito aos membros da classe;
 - **protected**
 - acesso permitido à todas as classes do mesmo pacote;
 - acesso permitido a todas as sub-classes (herança);
 - **default ou package-private (sem modificador explícito)**
 - acesso permitido à todas as classes do mesmo pacote.
-

Corrigindo o Mau Exemplo anterior

- Corrija o que você eventualmente fez errado na implementação anterior (se fez errado!!).
 - Praticando (mais 13 minutos).
 - Escreva a classe **ContaCorrente** cujos atributos **saldo** e **chequeEspecial**.
 - Lembre-se de encapsular estes atributos e permitir manipulação através dos métodos **sacar** e **depositar**, e acessos através dos getters.
 - Depois, escreva uma classe chamada **Loja**. Toda loja tem uma **conta corrente (herança)**.
 - No Main crie um objeto conta para realizar um saque e uma loja para realizar um depósito;
 - Mostrar os saldos da conta e da loja.
-

Com Encapsulamento

```
public class ContaCorrenteComEncapsulamento {
    private int saldo;
    private int chequeEspecial;

    public ContaCorrenteComEncapsulamento(int saldo, int chequeEspecial) {
        this.saldo = saldo;
        this.chequeEspecial = chequeEspecial;
    }

    public int getSaldo() {
        return saldo;
    }

    public int getChequeEspecial() {
        return chequeEspecial;
    }

    public void sacar(int valor) throws Exception {
        if (valor <= 0)
            throw new Exception ("Nao podemos sacar um valor negativo ou zero! Valor:" + valor);
        else if (valor > (saldo + chequeEspecial))
            throw new Exception ("Opecacao negada! O cliente nao tem saldo suficiente! Saldo atual: " +
                (saldo + chequeEspecial));
        else
            saldo -= valor;
    }
}
```



Continuando o exercício...



Como seria o método **depositar()**; ?

Vamos desenvolver???????

Visão Geral sobre o Google Android

O que é o Android????

Android é uma plataforma desenvolvida pela Google voltada para dispositivos móveis, totalmente aberta é livre (Open Source), que foi divulgada em 5 de novembro de 2007.





Visão Geral sobre o Google Android



Inicialmente o sistema Android foi desenvolvido pelo Google e atualmente essa plataforma é mantida pela OHA (Open Handset Alliance (<http://www.openhandsetalliance.com>), um grupo constituído por aproximadamente 84 empresas as quais se uniram para inovar e acelerar o desenvolvimento de aplicações e serviços, com o objetivo de trazer aos consumidores uma experiência mais rica em termos de recursos, menos dispendiosa em termos financeiros para o mercado móvel.

Visão Geral sobre o Google Android

Um dos primeiros SmartPhones que ofereceu suporte a esse sistema operacional foi o G1 da empresa T-Mobile. Confira na imagem seguinte:



G1 - T-Mobile



Visão Geral sobre o Google Android



Popularmente hoje o sistema Android encontra-se não somente nos SmartPhones como também em Tablets, TVs (Android TV), Automóveis (Android Auto) e também em relógios de pulso (o famoso Android Wear).

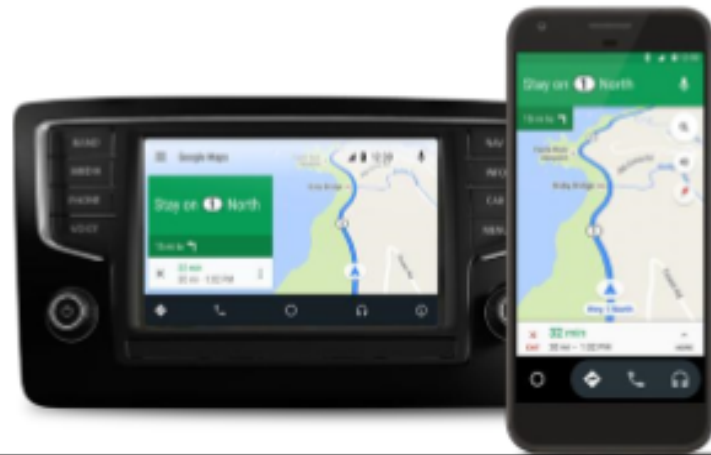
Seguem nos próximos slides alguns dos dispositivos que encontramos hoje no mercado com o sistema operacional Android:

Visão Geral sobre o Google Android

Dispositivos Android	
Tipos	Exemplos
SmartPhones	
Tablets	

Visão Geral sobre o Google Android

Automóveis (Android Auto)



Visão Geral sobre o Google Android

Dispositivos Android	
Tipos	Exemplos
TVs (Android TV)	
Relógio de Pulso (Android Wear)	



Estrutura Geral da plataforma Google Android



O Android SDK é uma ferramenta de desenvolvimento que disponibiliza um conjunto de APIs necessárias para desenvolver aplicações para a plataforma Android, utilizando a linguagem Java.

Vamos conhecer os recursos encontrados nessa plataforma:

- **Application framework:** Permite a reutilização e substituição de componentes ;
 - **Dalvik virtual machine:** É uma Máquina Virtual Java (JVM) voltada para dispositivos móveis ;
 - **Browser Integrado** baseado no webkit engine ;
 - **Gráficos Otimizados** O Android é constituído por bibliotecas 2D e 3D baseada na especificação OpenGL ES 1.0 ;
-

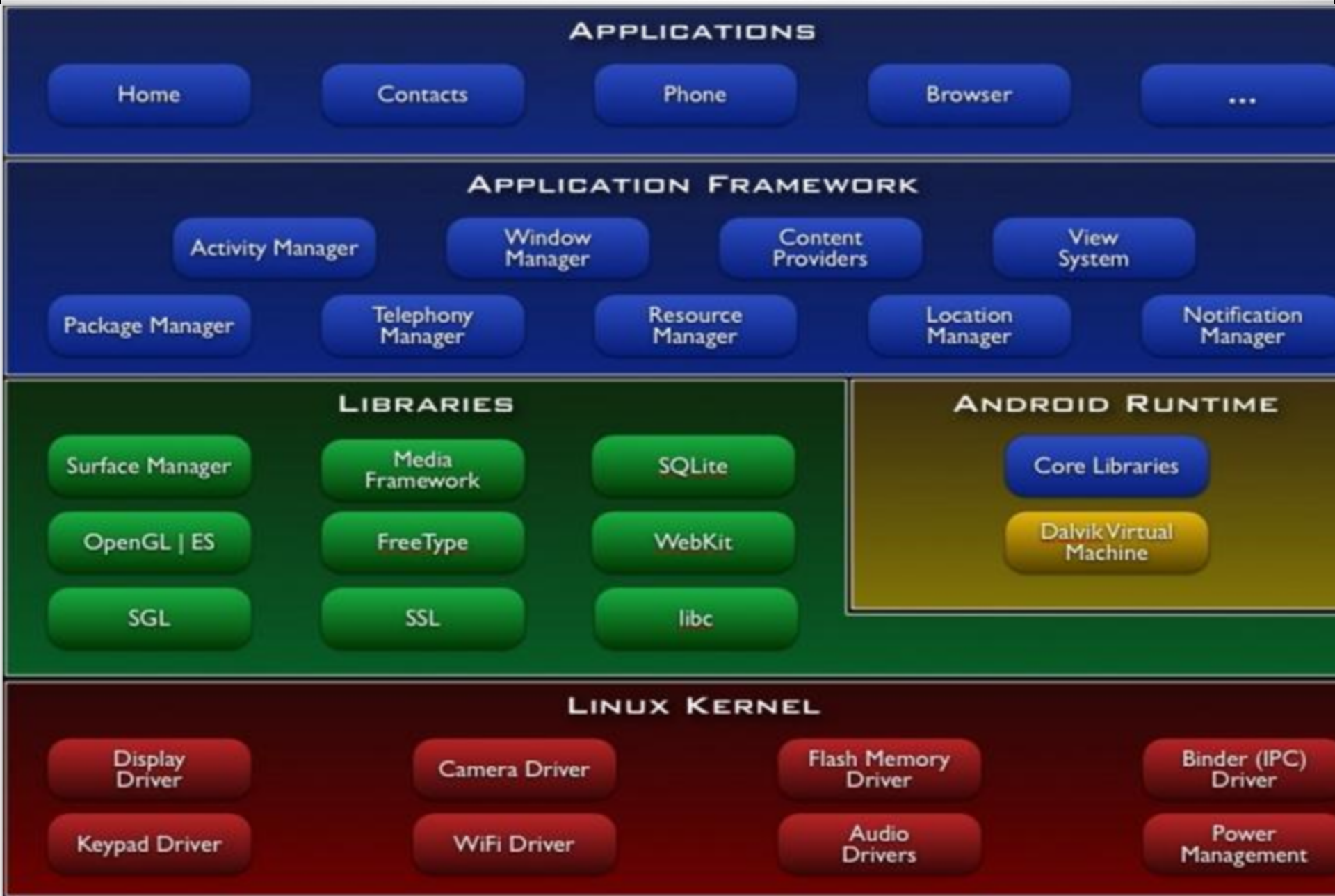


Estrutura Geral da plataforma Google Android



- **SQLite:** Sistema Gerenciador de Banco de Dados (SGBD) já embutido no Android para guardar dados ;
 - **Suporte multimídia:** A plataforma já oferece para áudio, vídeo e formatos de imagem (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF) ;
 - **Telefonia GSM** (dependente de hardware) ;
 - **Bluetooth, EDGE, 3G, e WiFi** (dependente de hardware) ;
 - **Câmera, GPS, compasso, e acelerômetro** (dependente de hardware) ;
 - **Rico ambiente de desenvolvimento** , incluindo um emulador de dispositivo, ferramentas de depuração, memória e performance.
-

A arquitetura Android





Aplicações fundamentais



O Android nos fornece um conjunto de aplicações fundamentais, são elas:

- um cliente de e-mail;
 - programa de SMS;
 - agenda;
 - mapas;
 - navegador;
 - contatos entre outros.
-



Aplicações fundamentais



Todos os aplicativos presentes no Android foram desenvolvidos na linguagem de programação Java.

O Android nos fornece um conjunto de bibliotecas C/C++ utilizadas por vários componentes do sistema.

Vejamos algumas das bibliotecas no próximo slide:

Aplicações fundamentais

- **System C library:** Consiste em uma implementação derivada da biblioteca C padrão baseado no sistema (libc) do BSD sintonizada para dispositivos rodando Linux.
 - **Media Libraries:** Baseado no PacketVideo's OpenCORE; são as bibliotecas que suportam os mais diversos formatos de áudio e vídeo, incluindo também imagens.
 - **Surface Manager:** Responsável pelo acesso ao subsistema de exibição bem como as múltiplas camadas de aplicações 2D e 3D;
-

Aplicações fundamentais

- **LibWebCore**: Consiste em um web browser engine utilizado tanto no Android Browser quanto para exibições web.
 - **SGL**: o engine de gráficos 2D - 3D libraries: Uma implementação baseada no OpenGL ES 1.0 APIs; As bibliotecas utilizam aceleração 3D via hardware (quando disponível) ou o software de renderização 3D altamente otimizado incluído no Android.
 - **FreeType** – Biblioteca responsável pela renderização de fontes bitmap e vector;
-



Aplicações fundamentais



- **SQLite**: conforme já mencionado, consiste no sistema gerenciador de banco de dados (SGBD) relacional disponível para todas as aplicações.



Android Runtime

O Android é constituído por um conjunto de bibliotecas que fornece a maioria das funcionalidades disponíveis nas principais bibliotecas da linguagem Java.

Toda aplicação Android roda em seu próprio processo, com sua própria instância da máquina virtual Dalvik. O Dalvik foi escrito de forma a executar várias VMs eficientemente.

Ele executa arquivos .dex, que é otimizado para consumo mínimo de memória.

Android Runtime

A VM é baseada em registros e roda classes compiladas pela linguagem Java que foram transformadas em arquivos .dex, através da ferramenta “dx” incluída no SDK.

O Dalvik VM foi baseado no kernel do Linux para funcionalidades subjacentes como o encadeamento e a gestão de baixo nível de memória.



Linux Kernel



O Android foi projetado em cima da versão 2.6 do kernel do Linux para os serviços centrais do sistema, tais como segurança, gestão de memória, gestão de processos, etc.

O kernel também atua como uma camada de abstração entre o hardware e o resto do software.



Por hoje é só !!!

Até a próxima aula...
