Programação para Dispositivos Móveis

Fatec Ipiranga
Análise e Desenvolvimento de Sistemas

Aula 04 – Anatomia de uma aplicação Android

Dalton Martins
dmartins@gmail.com

São Paulo, Fevereiro, 2012

Anatomia de uma aplicação Android

Desenvolver uma aplicação para Android envolve termos um bom conhecimento de como funciona sua arquitetura;
É fundamental, pois será a partir dela que conseguiremos organizar e projetar nossas aplicações;
Hoje, veremos os blocos de construção dessa arquitetura, seu papel, função e modo de organização.

Entendendo a terminologia Android

Context: é o centro de comando central de uma aplicação. Todas as funcionalidades específicas de uma aplicação são referenciadas a partir de um contexto;
Activity: Uma aplicação Android consiste numa coleção de tarefas, sendo que cada uma é chamada de Atividade. Cada atividade tem um único propósio;
Intent: É um mecanismo assíncrono para facilitar a troca de mensagens entre atividades. Funciona como uma declaração de intenção de se fazer algo;
Service: Os serviçõs são tarefas que não requerem interação com usuário.

Usando o Contexto da Aplicação

O **Context** é o local central para todas as funcionalidades de alto nível de uma aplicação;
A classe Context pode ser usada para gerenciar detalhes das configurações específicas de uma aplicação, assim como as operações e dados;
Usaremos o Context para acessar configurações e recursos compartilhados entre diversas instâncias de **Activity** (atividades);

Usando o Contexto da Aplicação

Recuperando o Context da aplicação:

Context context = getApplicationContext();

Após recuperar um contexto válido da aplicação, pode ser utilizado para acesso as suas funcionalidades e serviços;

Podemos recuperar recursos de uma aplicação usando o método getResources();

Podemos recuperar as preferências compartilhadas de uma aplicação com a classe SharedPreferences(), que pode ser utilizada para salvar dados simples de uma aplicação, como suas configurações;

O melhor modo de recuperar um recurso é utilizando seu identificador, um número único criado dentro da classe R.java. Ex:

String teste = getResources().getString(R.string.hello);

Usando o Contexto da Aplicação

Outras coisas que podemos fazer a partir de um Context de uma aplicação:

- Lançar uma instância de Activity (será um de nossos usos mais comuns);
- Recuperar dados encapsulados na aplicação;
- Requisitar um serviço do sistema, tal como georeferenciamento;
- Gerenciar arquivos privados da aplicação, diretórios e bancos de dados;
- Inspecionar e garantir permissões da aplicação.

Executando tarefas com as Activities

A classe Activity é o núcleo de um aplicação Android;
Muito de nosso tempo passaremos definindo e implementando uma classe Activity para cada uma das telas de nossa aplicação;
Vejamos um exemplo:

Um jogo possui 5 telas e, logo, 5 Activities para organizar seu funcionamento:

A tela de entrada: ponto de entrada primário da aplicação;

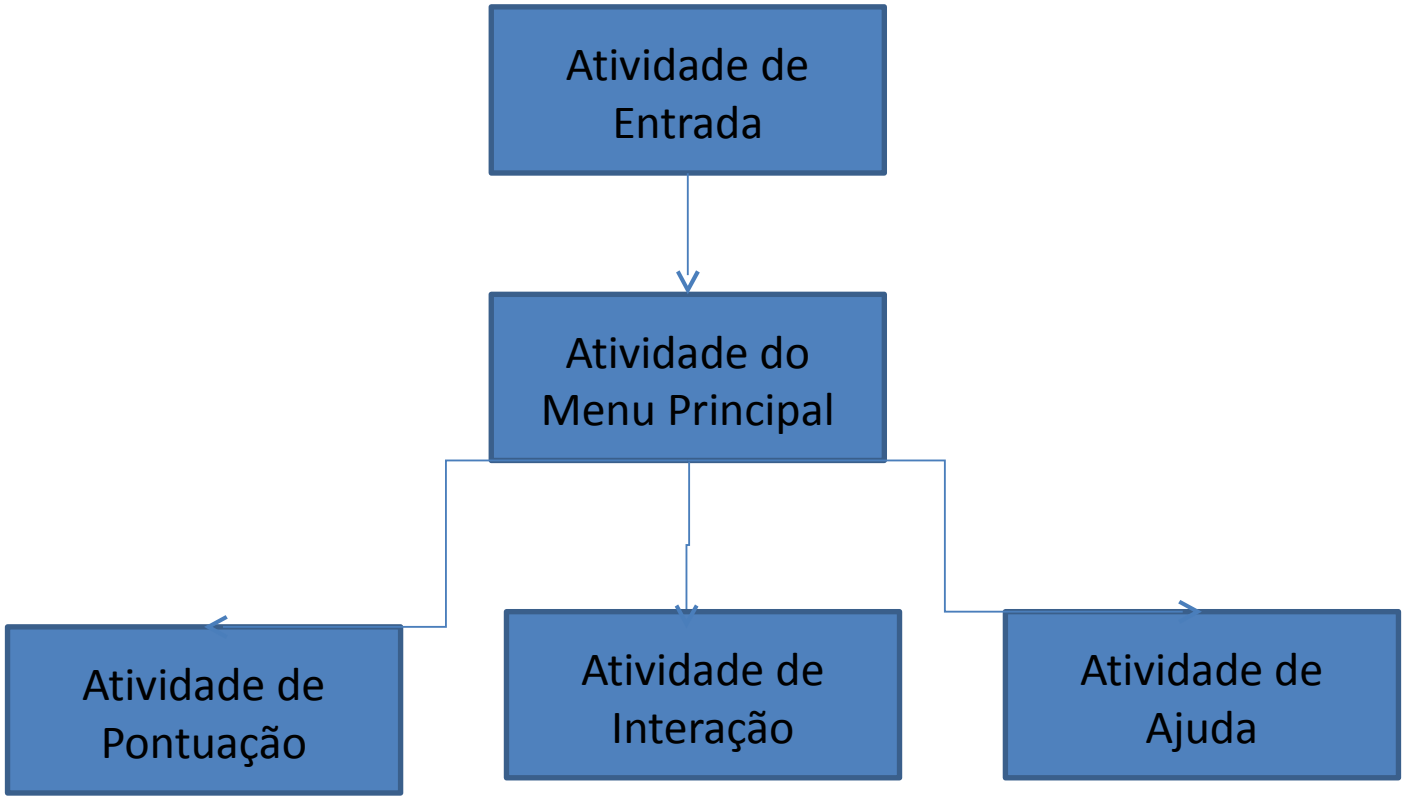
O menu principal: essa atividade funciona como interface para o usuário acessar as principais atividades da aplicação;

A tela do jogo: é onde a interatividade do jogo ocorre;

A tela de pontuação: é onde aparem os pontos após cada rodada;

A tela de ajuda: é onde aparecem informações sobre jogar e sobre a aplicação.

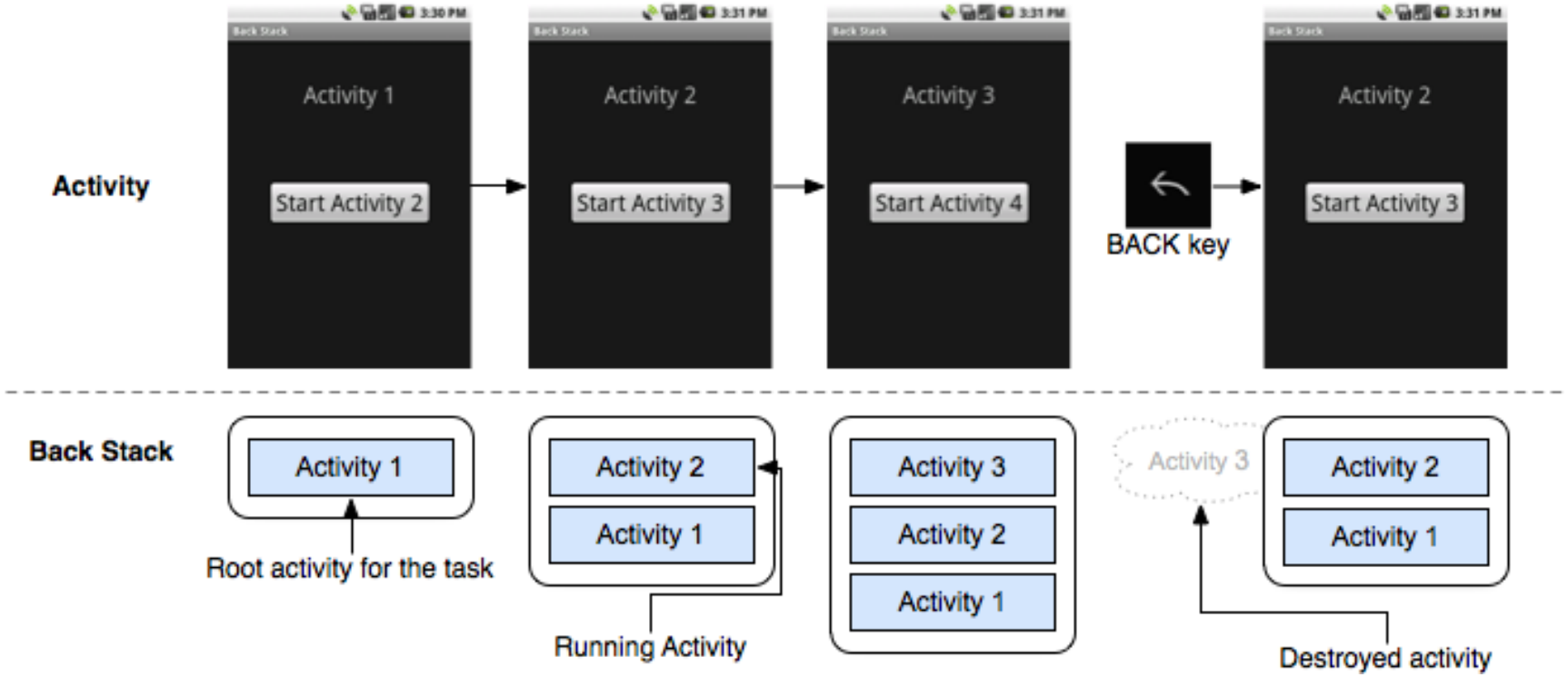
Exemplo de estrutura da Aplicação



O ciclo de vida de uma Activity

Aplicações podem rodar como processos em background, ou mesmo serem interrompidas ou serem colocadas em estado de espera quanto um evento do tipo o telefone tocar ocorrer;
Somente pode haver uma atividade visível para o usuário por vez;
O sistema operacional do Android monta uma pilha de atividades em operação. Quando uma nova atividade inicia, a atividade que estava no topo da pilha entra em espera e a nova atividade vai para o topo da pilha.
Quando a atividade finaliza, ela é removida da pilha e a atividade prévia é ativada.

O ciclo de vida de uma Activity



É de responsabilidade da aplicação gerenciar seu estado, sua memória, recursos dados. Isso significa que nós devemos colocar atividades em espera e torná-las ativas novamente.

Activities callbacks: gerenciando o estado e os recursos de uma aplicação

No ciclo de vida de um Activity, temos diferentes estados que um programador deve conhecer para poder controlá-los:

OnCreate (Bundle savedInstanceState);

onStart();

onRestart();

onResume();

onPause();

onStop();

onDestroy().

Entendendo cada um dos métodos

- **onCreate** → tem um único parâmetro, o Bundle, que será nulo se estivermos criando uma nova atividade. Se a atividade foi morta pelo sistema e agora reiniciada, o Bundle contém o seu estado prévio. É adequado configurarmos o layout da tela e conexão com dados neste método. Será aqui que faremos muitas vezes a chamada a setContentView();

Entendendo cada um dos métodos

- onResume → Quando uma atividade atinge novamente o topo da pilha e sai do background para se tornar a atividade ativa, este método é chamado. Ainda que a atividade não esteja visível, ele é o lugar apropriado para recuperar os recursos que a atividade precisa para rodar. É o lugar adequado para iniciar vídeos, músicas e animações.

Entendendo cada um dos métodos

- **onPause()** → Quando uma nova atividade é realocada para ser a atividade principal, a atividade em execução é colocada em estado de espera. Aqui é onde devemos parar qualquer vídeo, música ou animação que foi iniciada pela aplicação.
- É também aqui a última chance que temos de liberar qualquer recurso inútil a aplicação enquanto ela estiver em estado de espera.
- O que é fundamental é tudo o que seja feito aqui seja feito de modo rápido para não travar próximas atividades que desejam entrar no topo da pilha.

Entendendo cada um dos métodos

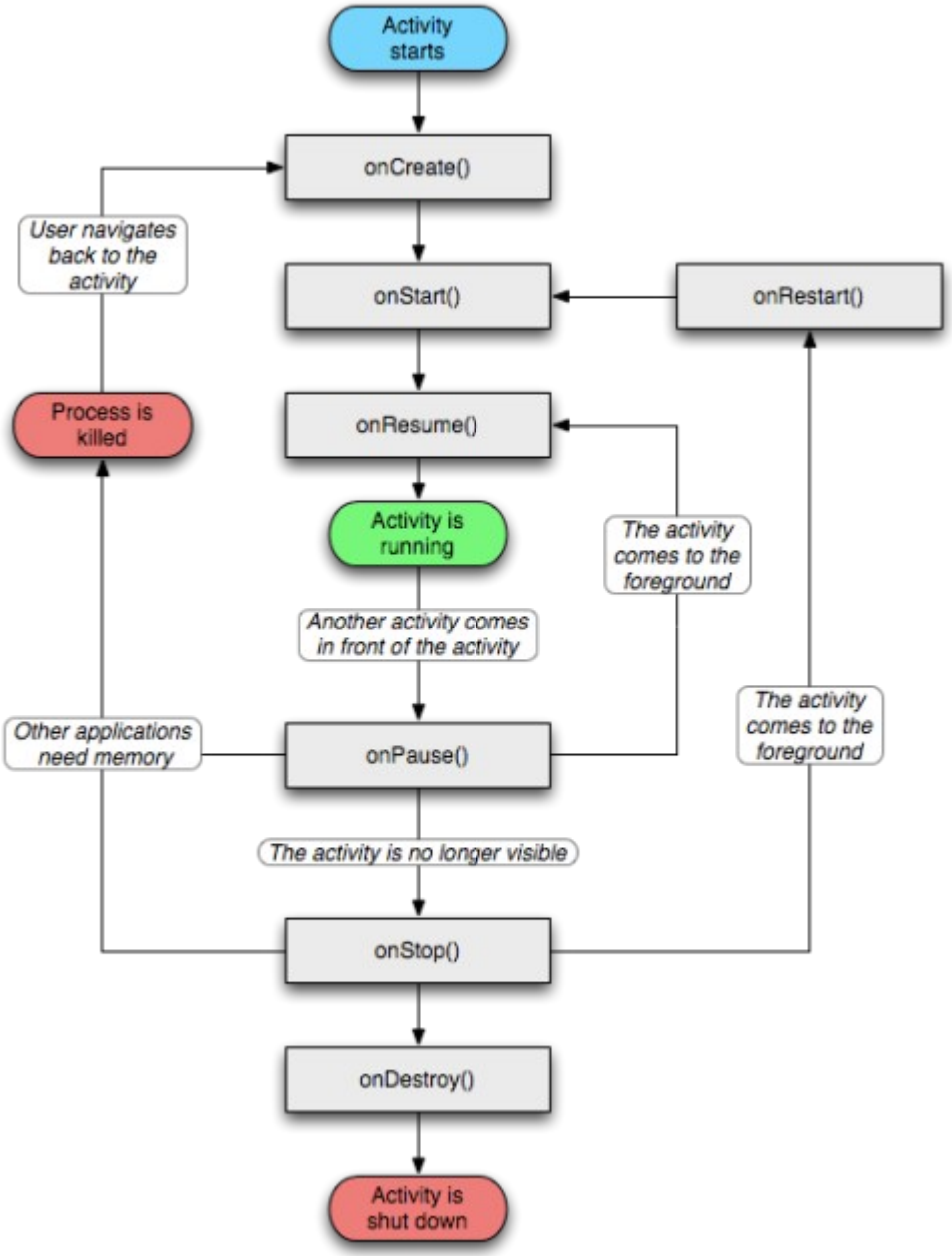
- Sob condições extremas de memória e bateria, o sistema Android pode decidir matar alguma atividade que esteja na pilha aguardando para ser colocada em execução. Ou seja, qualquer atividade pode desaparecer!
- Se uma atividade é morta depois de onPause(), os métodos, os métodos **onStop() e onDestroy()** não podem ser chamados. Por isso, quanto menos recursos uma atividade consumir em estado de espera, menor suas chances de ser morta pelo sistema.
- A atividade não desaparece da pilha. O estado da atividade é salvo em seu Bundle, se ele for implementado o método onSaveInstanceState().

Entendendo cada um dos métodos

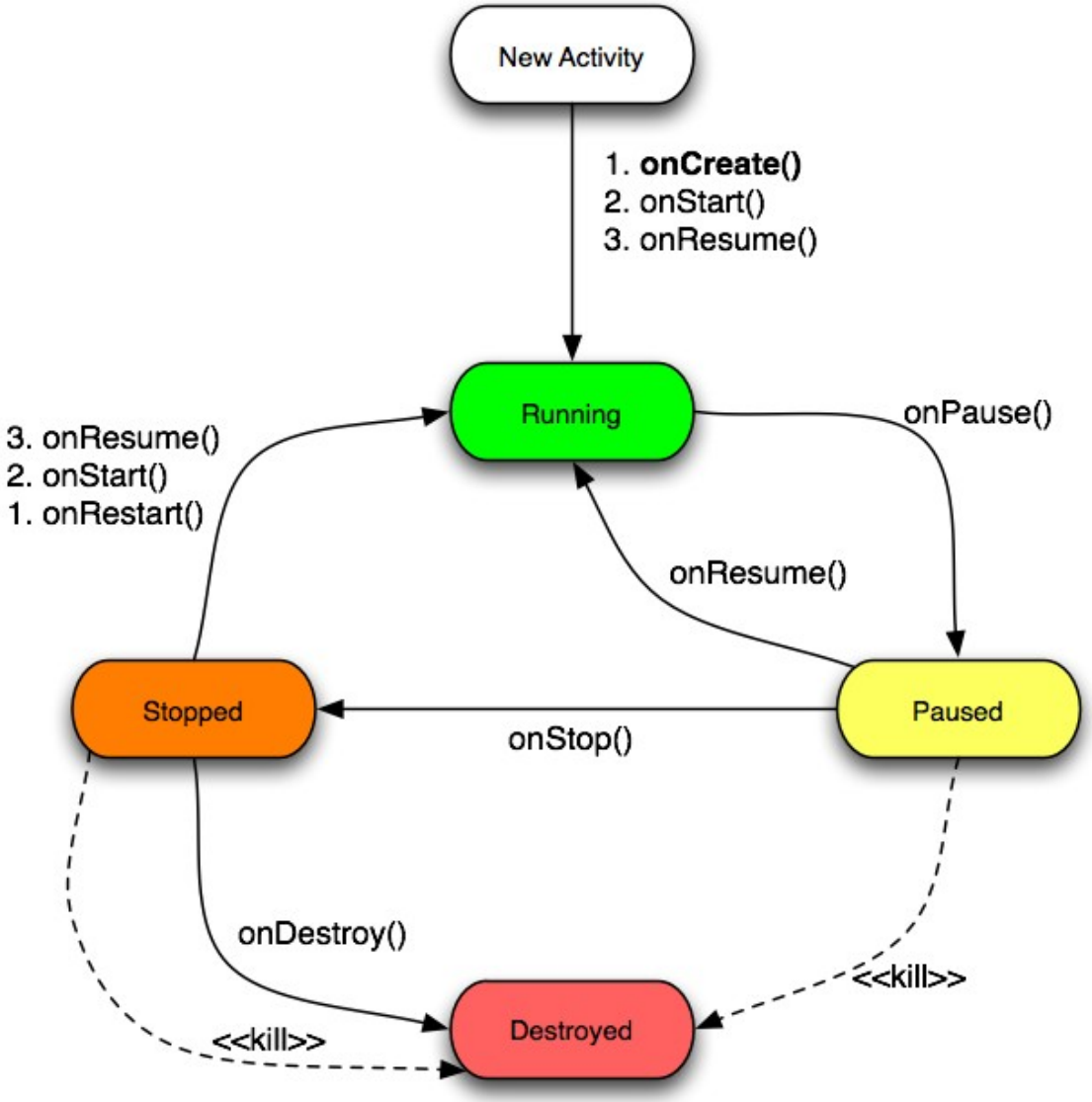
- Se uma atividade é vulnerável a ser morta pelo Android, a Atividade pode salvar suas informações de estado no seu objeto Bundle.
- Isso pode ser feito usando **onSaveInstanceState();**
- Porém, nem sempre é garantido. Logo, vale salvar qualquer dado fundamental no método onPause();
- Quando a atividade é reiniciada, o Bundle é passado para o método onCreate, permitindo a atividade voltar ao seu estado anterior.
- Podemos também recuperar o Bundle no método onStart() usando o método onRestoreInstaceState().

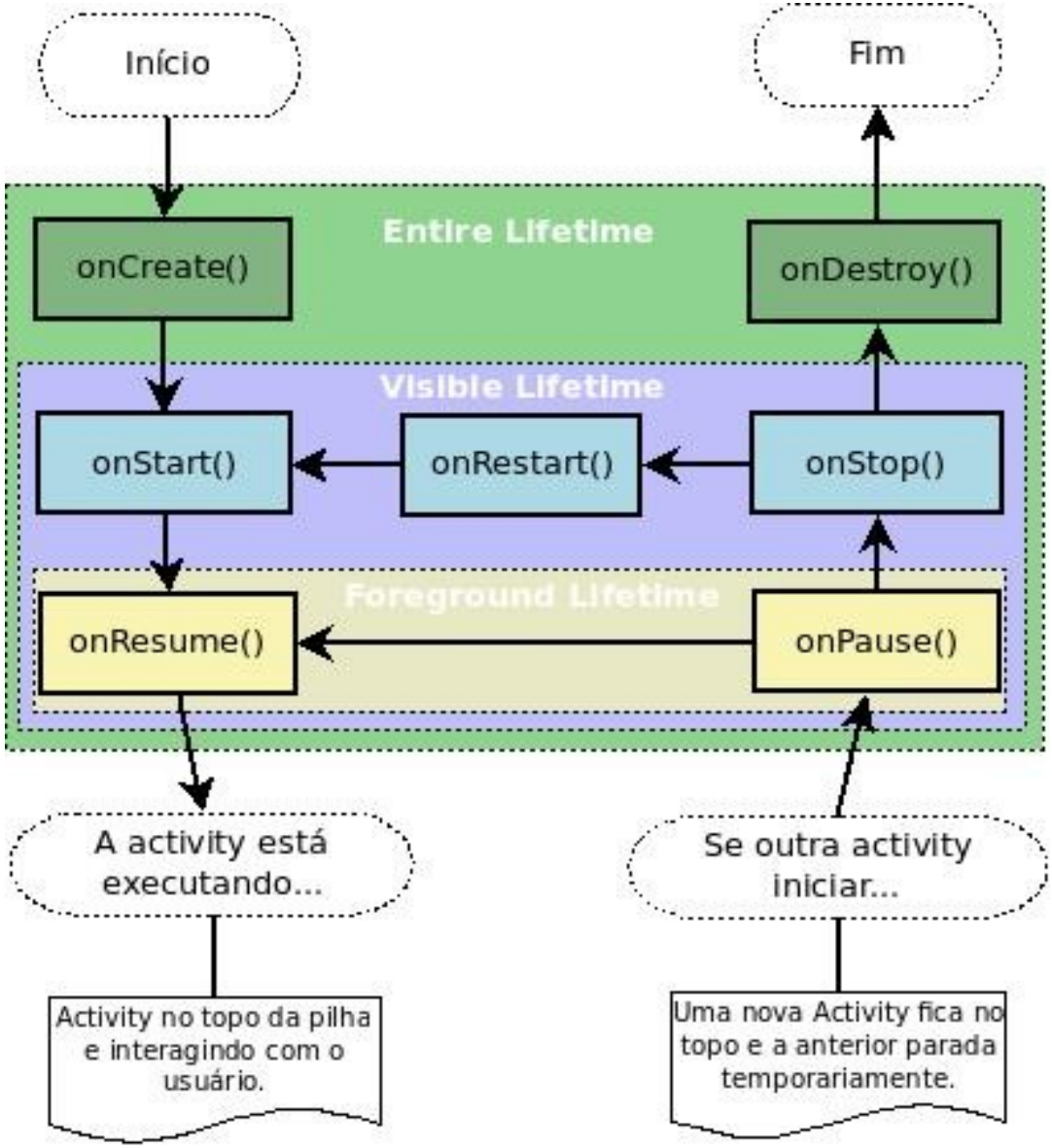
Entendendo cada um dos métodos

- **OnDestroy()** → pode ser chamado de dois modos possíveis:
  - Quando a atividade é finalizada;
  - Quando o sistema Android precisa finalizar uma atividade por falta de recursos.


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

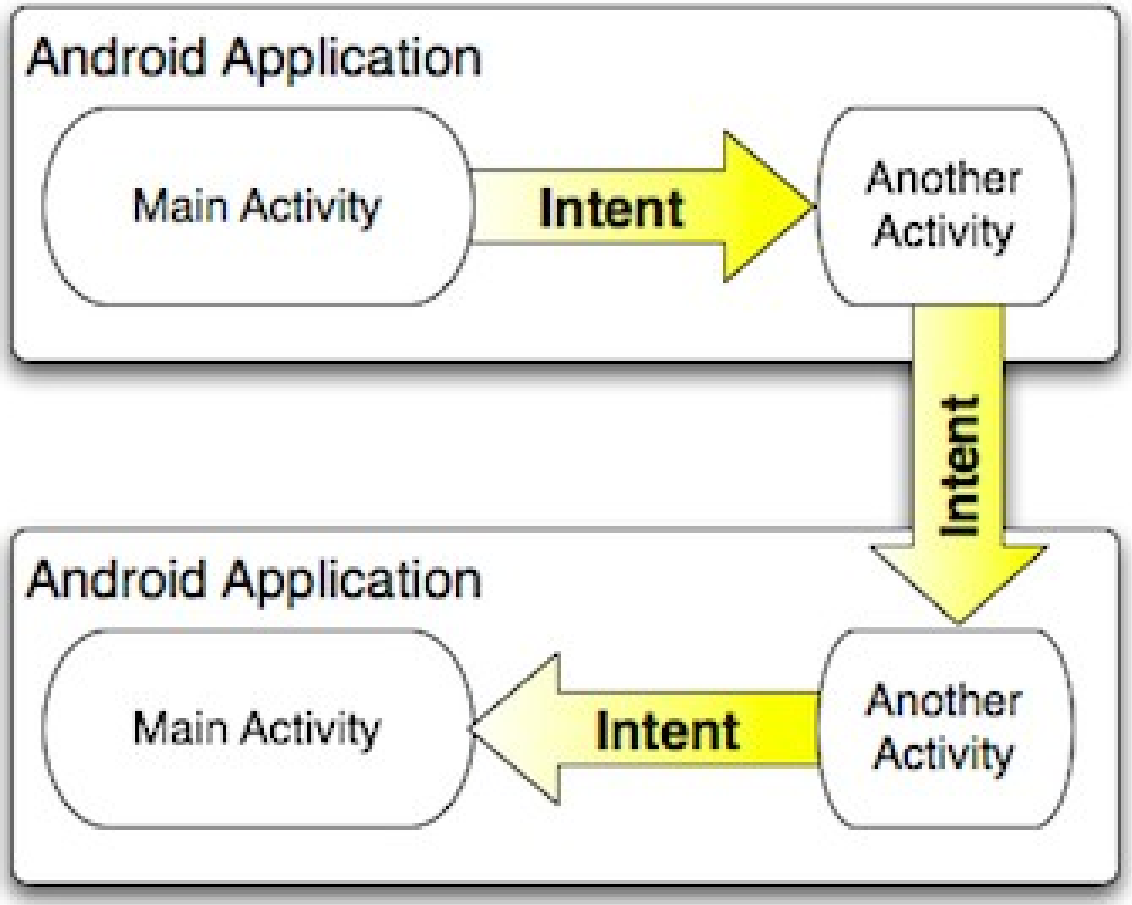
Activity Lifecycle




Início
Fim
Entire Lifetime
onCreate()
onDestroy()
Visible Lifetime
onStart()
onRestart()
onStop()
Foreground Lifetime
onResume()
onPause()
A activity está executando...
Se outra activity iniciar...
Activity no topo da pilha e interagindo com o usuário.
Uma nova Activity fica no topo e a anterior parada temporariamente.

Gerenciando transições de atividades com Intents

- Ao longo do ciclo de vida de uma aplicação, um usuário transita por diferentes atividades;
- Podemos iniciar atividades de vários modos:
  - Definindo pontos de entrada no arquivo AndroidManifest.xml;
  - Usar o contexto da aplicação para iniciar uma atividade. Para isso, usamos o método startActivity(), que deve ser parametrizado com uma **Intent**
- Uma intent é um mecanismo de mensagem assíncrono utilizado pelo sistema Android para compatibilizar uma ação com uma atividade específica ou um serviço. Ex.
  - startActivity(new Intent(getApplicationContext(), MyDrawActivity.class));
  - Podemos usar o Intent para passar dados entre uma atividade e outra.


Android Application
Main Activity
Intent
Another Activity
Intent
Android Application
Main Activity
Intent
Another Activity

Criando Intents com ações e dados

- Intents não precisam especificar um componente ou classe explicitamente para ser inicializada;
- Podemos criar uma espécie de filtro Intent registrá-lo no arquivo AndroidManifest.xml;
- O Android vai buscar qual atividade deve ser lançada com base no filtro escolhido;
- Sabendo que uma Intent tem que receber, por sua sintaxe um par de ação/dado, vejamos o exemplo a seguir...

Criando Intents com ações e dados

- O exemplo mostra como uma atividade pode chamar uma atividade de outra aplicação:

```
Uri numero = Uri.parse(tel:30999999);

Intent ligar = new Intent(Intent.ACTION_DIAL, numero);

startActivity(ligar);
```

Uma lista de filtros prontos do Google:
http://developer.android.com/guide/appendix/g-app-intents.html