Desenvolvimento para dispositivos móveis

Activity e ciclo de vida da Activity

Professor Msc. Fabio Pereira da Silva
E-mail: fabio.pereira@faculdadeimpacta.com.br

Github

- Link com os códigos gerados na aula passada:
- https://github.com/fabiodasilva500/Aulas-Mobile/tree/Aula-6-Overview-Android

Introdução

- Na aula anterior criamos um aplicativo simples para ver o funcionamento básico de um aplicativo Android
- Ele continha basicamente uma tela (arquivo xml, dentro da pasta layout) e um arquivo Kotlin, onde foram definidos a tela e os eventos.
- A classe no arquivo Kotlin é conhecida como Acitivity

Introdução

- A classe Activity é uma das classes mais importantes no Android
  - Geralmente ela representa uma tela no aplicativo
  - É responsável por definir qual será a View que desenha a interface gráfica
  - É responsável por controlar os eventos da tela
- Usualmente um aplicativo tem mais de uma tela, e consequentemente mais de uma Activity
  - Sempre que for criar uma nova tela no aplicativo, é necessário ter uma Activity relativa a esta tela
- Activity == Atividade == Ações e funcionalidades que o usuário pode fazer no app

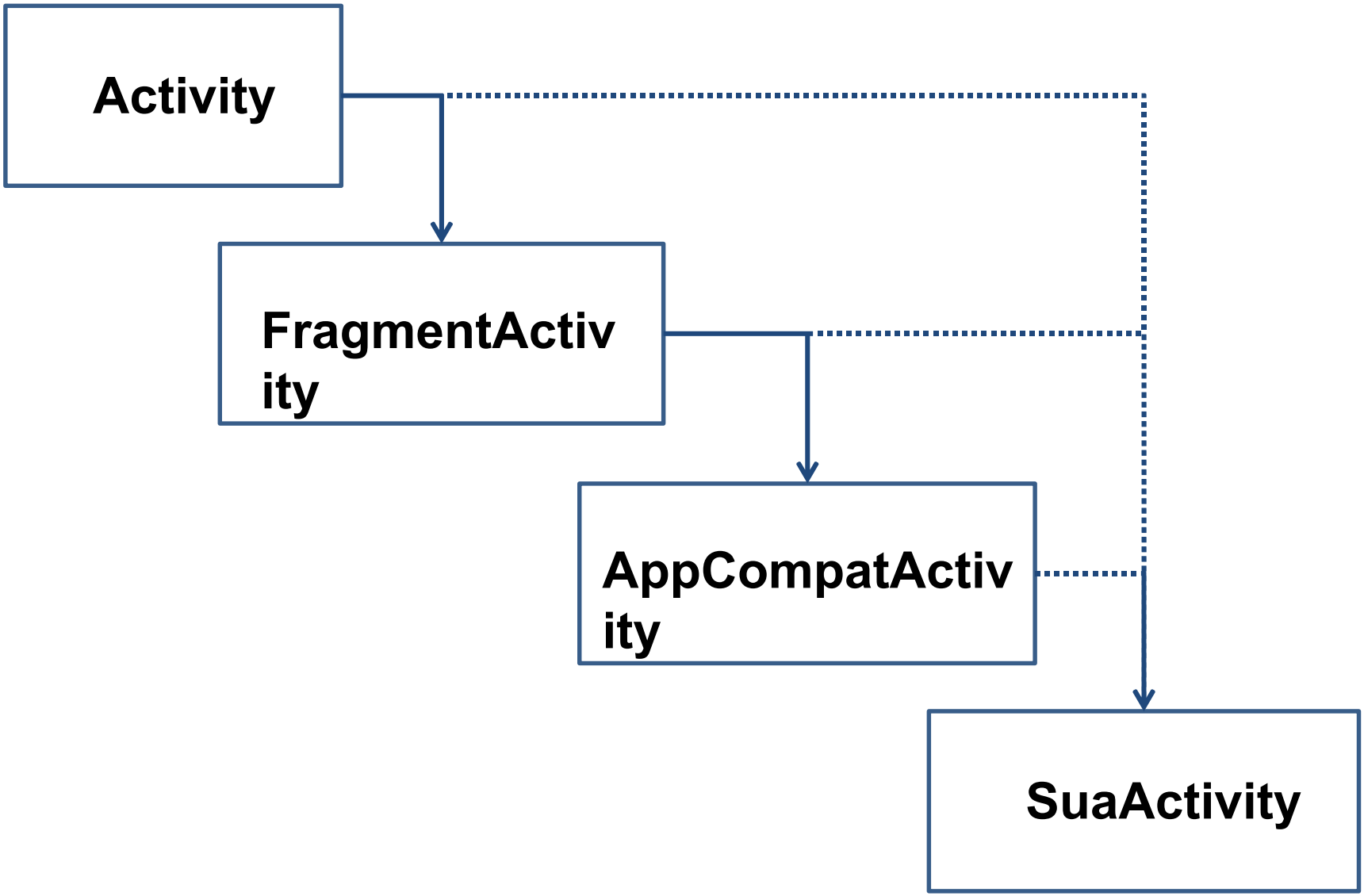
Activity

- Uma atividade representa uma única tela
  - Ex. Aplicação de e-mail
    Uma atividade para a lista de novos e-mails
    - Outra atividade para a tela de composição de e-mail
    - Outra atividade para a tela de leitura de um e-mail
- Cada atividade é independente da outra, mesmo que em conjunto trabalhem para um único proposito
- Aplicações podem iniciar qualquer uma dessas atividades (Mesmo em outra aplicação)
  - Ex. Uma aplicação relacionada com o gerenciamento de fotos pode iniciar a atividade de composição de e-mail para enviar a foto para um amigo

Activity

- Uma classe de Activity deve herdar todas as características (atributos) e comportamento (métodos) da classe adroid.app.Activity, ou alguma subclasse desta
  - Por exemplo, AppCompatActivity ou FragmentActivity (ambas são subclasses de Activity)
  - FragmentActivity Permite utilizar fragments em versões antigas do Android
  - AppCompatActivity possibilita que a ActionBar (barra superior dos aplicativo) funcione e versões antigas do Android
    - É subclasse de FragmentActivity

Activity


Activity
FragmentActivity
AppCompatActivity
SuaActivity

Activity

- Toda Activity deve:
  - Sobrescrever o método onCreate() da superclasse
    - Responsável por realizar a inicialização para executar a aplicação, como definir a interface do usuário
  - Ser declarada no AndroidManifest.xml
    - Similar a declarar um servlet no web.xml
    - A declaração é feita colocando o seguinte marcador
      dentro do marcador <application>

```
<activity android:name=".MainActivity" />
```

Activity

Estado	Descrição
Ativa	A Activity foi iniciada pelo usuário, está em execução e está a frente (foreground)
Pausada	A Activity foi iniciada pelo usuário, está em execução, mas alguma notificação ou outro elemento está a frente
Parada	A Activity foi iniciada pelo usuário, está em execução, mas está escondida (ocultada por outra atividade)
Morta	Atividade nunca foi iniciada (após uma inicialização do telefone ou tablet) ou a atividade foi finalizada por questão de falta de memória

Activity

```
<activity android:name=".MainActivity" />
```

- A declaração da Activity no arquivo de configuração é feita com a sintaxe anterior.
  - Ou seja, o pacote da classe é relativo ao pacote do projeto (definido quando o projeto foi criado), por exemplo br.com.fabiopereira)

Activity

- Caso a Activity esteja em outro pacote, basta colocar o caminho do pacote:
  - Um pacote dentro do pacote principal (br.com.fabiopereira.telas)

```
<activity android:name=".telas.MainActivity" />
```

  - Outro pacote, fora do pacote principal

```
<activity
android:name="br.com.fernandosousa.activities.MainActivi
ty" />
```

Resumo sobre Activity

- Uma Activity geralmente representa uma tela do aplicativo
- Deve implementar o método onCreate()
- Deve ser declarado em AndroidManifest.xml

Conceitos Fundamentais - Pilha

- Uma pilha é uma estrutura de dados em que o acesso é restrito ao elemento mais recente na pilha.
- Uma pilha é uma estrutura de dados que pode ser acessada somente por uma de suas extremidades para armazenar e recuperar dados.
- Por essa razão, uma pilha é chamada de estrutura *LIFO* (*last in first out*).

Pilha

- São listas em que as operações de remoção e inserção ocorrem sempre em locais específicos
- A inserção é feita sempre no final da lista
- A remoção é feita sempre no final da lista
- **Em função disso uma fila assume a condição LIFO**

Pilha

- Dada uma pilha P = (a(1), a(2), ..., a(n)), dizemos que a(1) é o elemento da base da pilha; a(n) é o elemento topo da pilha; e a(i+1) está acima de a(i).
- Em uma pilha “ideal”, operações básicas devem ocorrer em O(1), independentemente do tamanho N da pilha (ou seja, em tempo constante).

Pilha

- O conceito de pilha é usado em muitos softwares de sistemas incluindo compiladores e interpretadores.
- Como exemplo de sua utilização, *A maioria dos compiladores C usa pilha quando passa argumentos para funções).*
- As duas operações básicas – armazenar e recuperar – são implementadas por funções tradicionalmente chamadas de *push e pop*, respectivamente.
- A função *push()* coloca um item na pilha e a função *pop()* recupera um item da pilha.
- A região de memória a ser utilizada como pilha pode ser um vetor, ou uma área alocada dinamicamente.

Pilha

- Operações:
- Adicionar elemento (método push)
- Remover elemento (método pop)
- Verificar se a pilha está vazia
- Verificar se a pilha está cheia

Pilha

- Na implementação de pilha, em apenas uma das extremidades, chamada dė topo, é realizada a manipulação dos elementos, em oposição a outra extremidade, chamada de base.
- Todas as operações em uma pilha podem ser imaginadas como as que ocorre numa pilha de livros, jornais, revistas, papéis e vários outros exemplos de aplicação.

Pilha

- Exemplos de aplicação:
- Calculadora para expressões matemáticas
- Conversão de número decimal para binário
- Retirada de mercadorias de um caminhão de entregas
- Mecanismo de fazer/desfazer do Word
- Mecanismo de navegação de páginas na Internet (avançar e retornar).

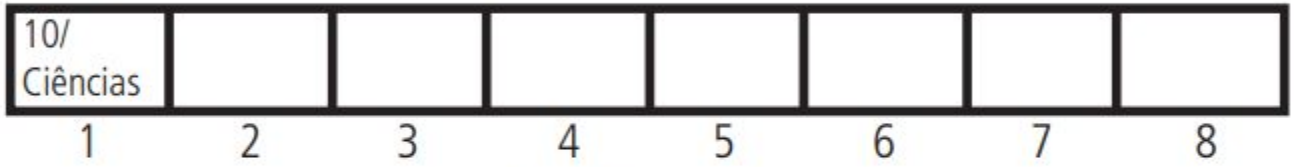
Pilha

- São listas em que as operações de remoção e inserção ocorrem sempre em locais específicos
- A inserção é feita sempre no final da lista
- A remoção é feita sempre no final da lista
- **Em função disso uma fila assume a condição LIFO**

Pilha

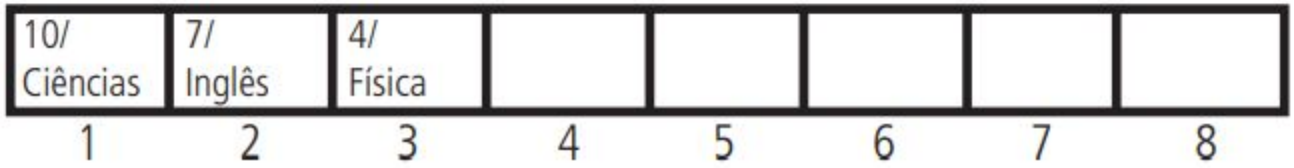
Vetor Vazio

Inserção do Livro com código10 e título Ciências



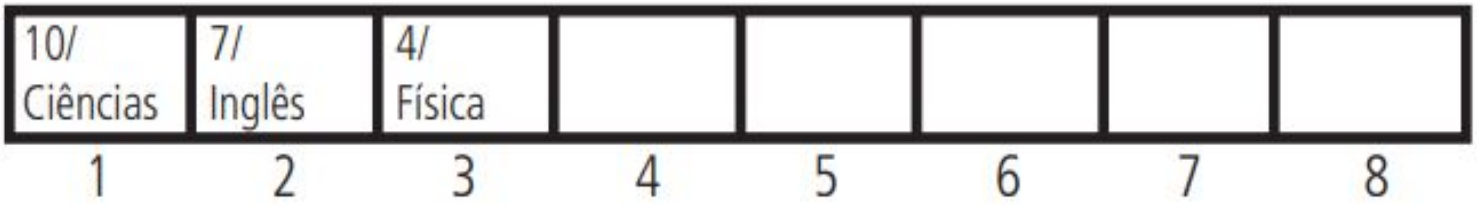
Inserção do Livro com código7 e título Inglês

Inserção do Livro com código4 e título Física

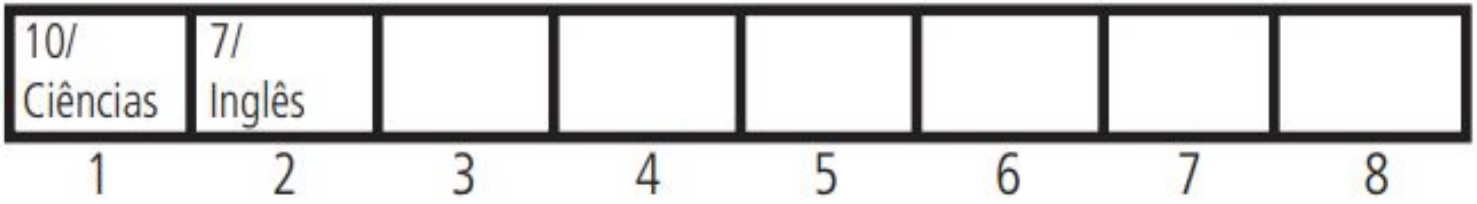


Pilha

Pilha com três elementos



Pilha depois da remoção



Fila

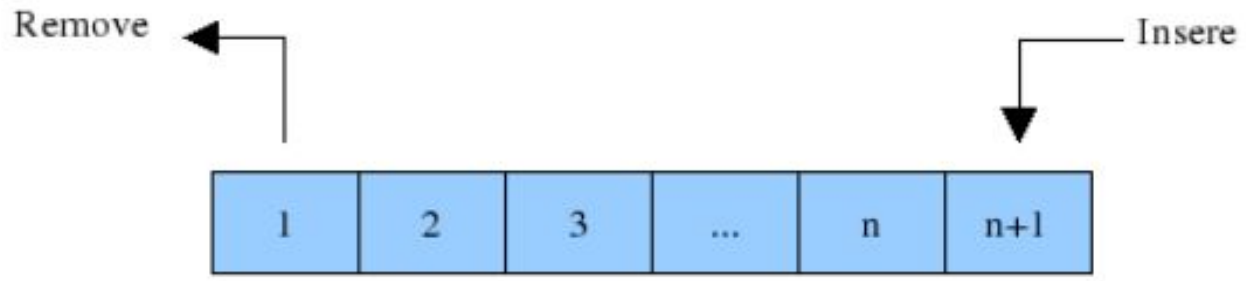
- Tipo abstrato de dados, em que o primeiro elemento inserido é o primeiro elemento retirado.
  - (FIFO – *First in First Out*)
- Aplicação: Sistemas operacionais: processamento, impressão de arquivos.
- Exemplos: Fila bancária, caixa lotérica, fila do cinema.

Fila

- São listas em que as operações de remoção e inserção ocorrem sempre em locais específicos
- A inserção é feita sempre no final da lista
- A remoção é feita sempre no início da lista
- **Em função disso uma fila assume a condição FIFO**

Fila

- São estruturas de dados do tipo FIFO (first-in first-out), onde o primeiro elemento a ser inserido, será o primeiro a ser retirado, ou seja, adiciona-se itens no fim e remove-se do início.



Fila

- Uma fila é caracterizada por ser uma linha de espera que cresce somando elementos ao seu final e que diminui tomando elementos da sua frente.
- Em uma extremidade os nós são somente adicionados, enquanto que na outra extremidade da fila os nós são apenas removidos.

Fila

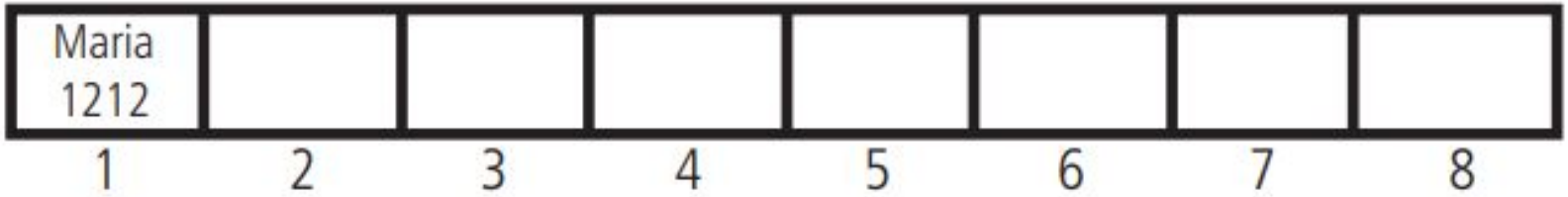
- A estrutura de fila é análoga ao conceito que temos de filas em geral. O primeiro a chegar é sempre o primeiro a sair, e a entrada de novos elementos sempre se dá no fim da fila.
- Em computação vemos este conceito sendo implementado em filas de impressão.
- Assim como as pilhas, uma fila também pode ser implementada por meio de um vetor ou de uma lista encadeada.

Fila

- As filas são frequentemente usadas em simulações, uma vez que existe uma *teoria das filas* bem desenvolvida e matematicamente sofisticada na qual vários cenários são analisados e modelos que usam filas são construídos.

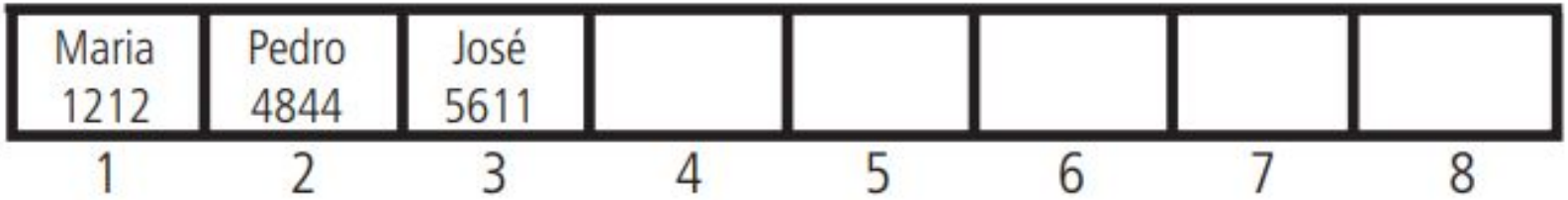
Fila

Inserção da aluna de matrícula 1212 e nome Maria



Inserção da aluna de matrícula 4844 e nome Pedro

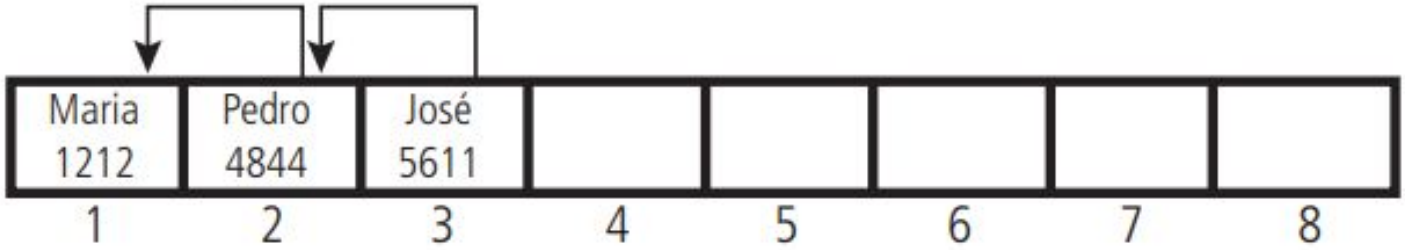
Inserção da aluna de matrícula 5611 e nome José



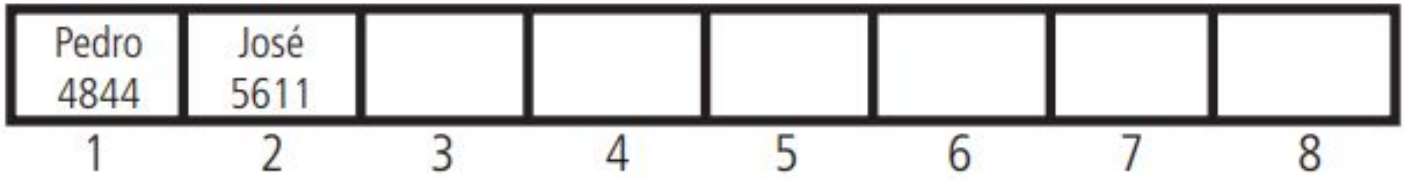
Fila

Fila com 3 elementos

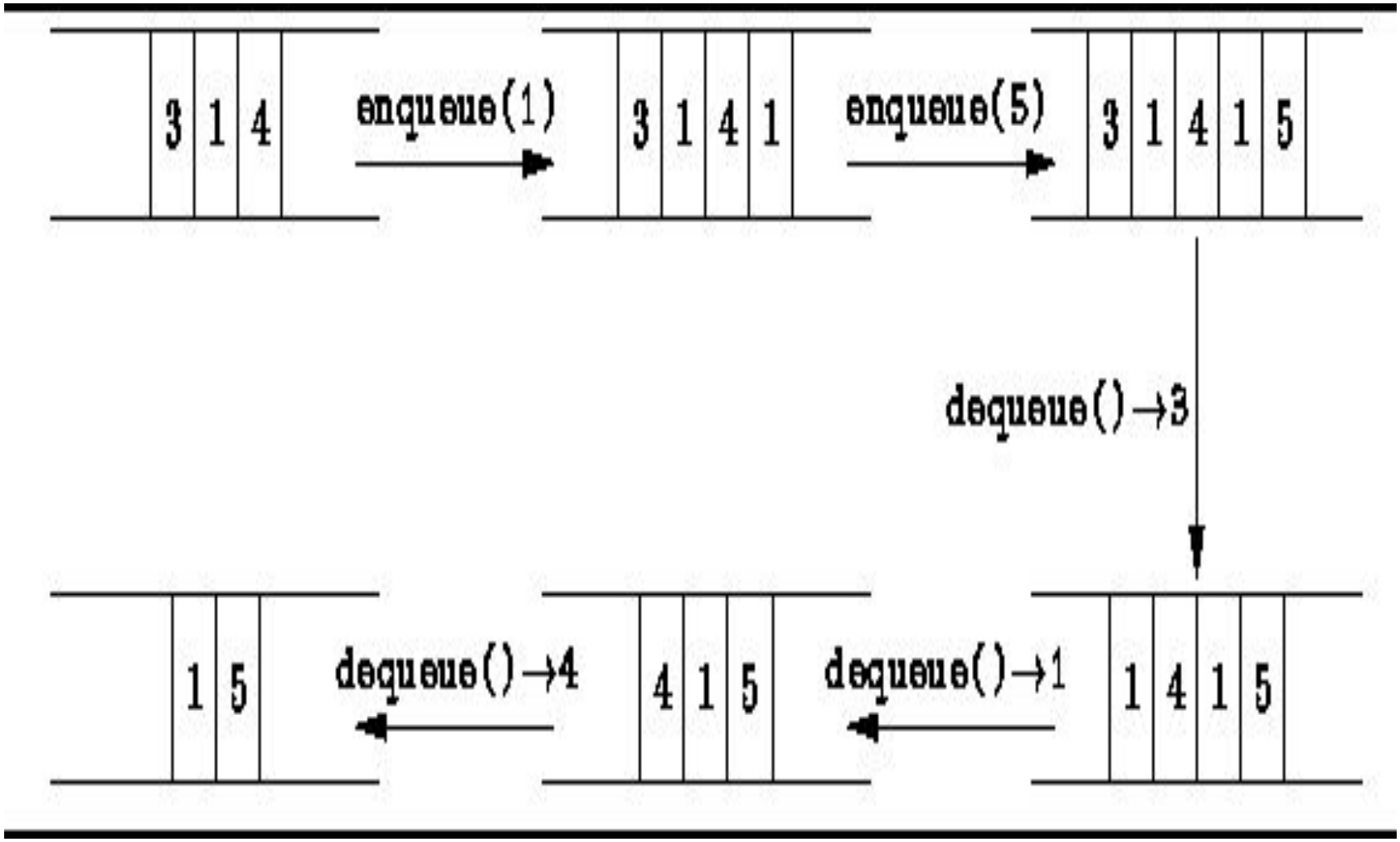
Retirada do primeiro da fila



Fila depois da remoção



Alocação de dados em Filas



Fila

- O acesso aos elementos da fila é realizado através das posições "entrada" e "saída" – as demais posições não são visíveis.
  - Estrutura do tipo FIFO (*first in, first out*).
  - Acesso: através de dois apontadores.
  - Manipulação: apenas a entrada e a saída são visíveis.

Leitura – posição "saída"	**Escrita – posição "entrada"**
• elemento ← f(*j*)	• f(*i*) ← elemento
• atualiza índice de saída *j*	• atualiza índice de entrada *i*

  - Fila cheia: índice de entrada *i* > índice máximo da fila.
  - Fila vazia: índice de entrada *i* = índice de saída *j*.

Fila

- Filas de impressão:
  - Impressoras tem uma fila, caso vários documentos sejam impressos, por um ou mais usuários, os primeiros documentos impressos serão de quem enviar primeiro;
- Filas de processos:
  - Vários programas podem estar sendo executados pelo sistema operacional. O mesmo tem uma fila que indica a ordem de qual será executado primeiro;
- Filas de tarefas:
  - Um programa pode ter um conjunto de dados para processar. Estes dados podem estar dispostos em uma fila, onde o que foi inserido primeiro, será atendido primeiro.

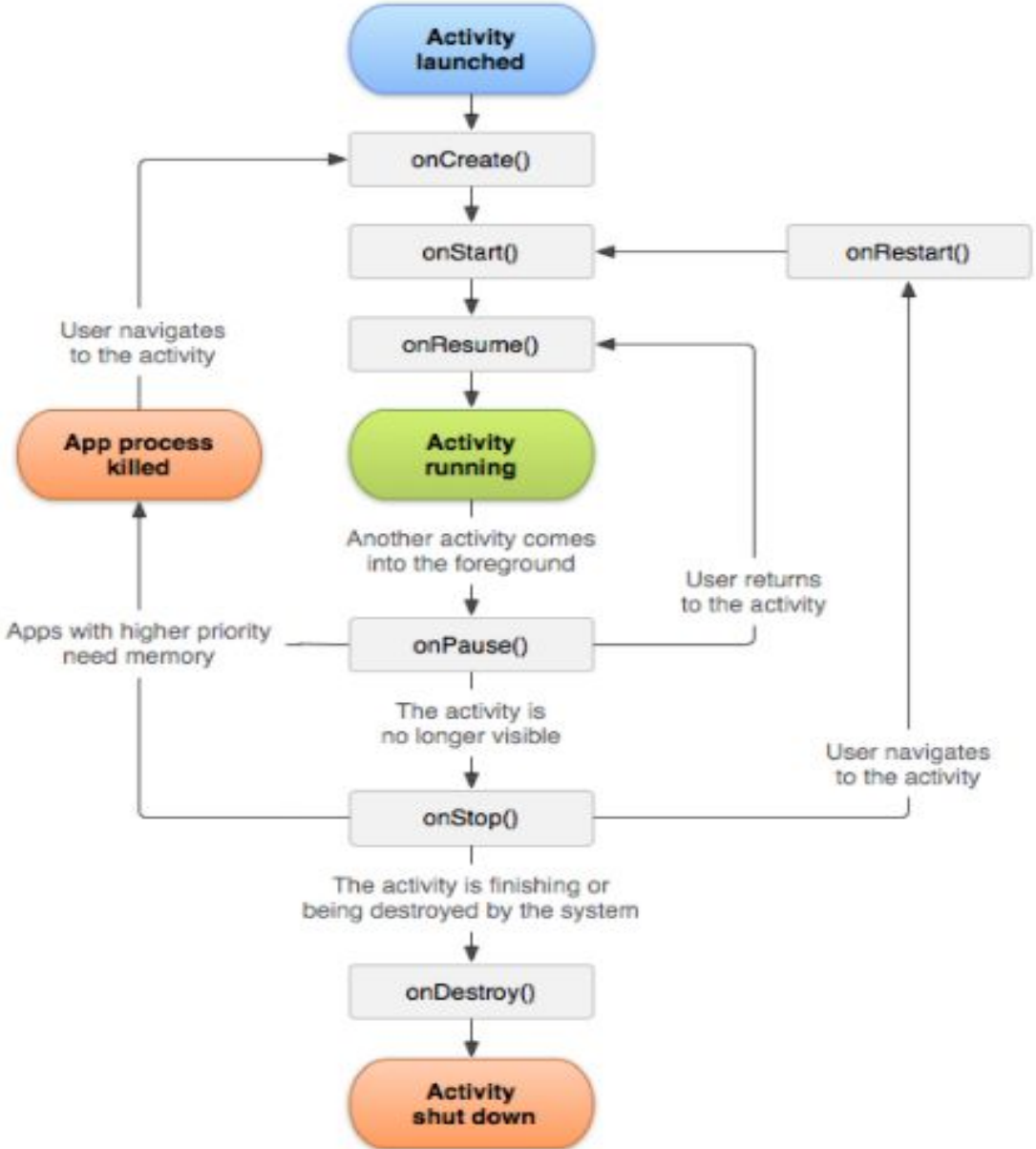
Fila

- Fila de Prioridades:
  - Cada item tem uma prioridade. Elementos mais prioritários podem ser atendidos antes, mesmo não estando no inicio da fila;
- Fila Circular:
  - Neste tipo de fila os elementos nem sempre são removidos ao serem atendidos, mas voltam ao fim da fila para serem atendidos novamente mais tarde.

Ciclo de vida da Activity

- Ciclo de vida está relacionado aos estados que em que uma Activity se encontra
  - Executando
  - Temporariamente interrompida (segundo plano)
  - Destruída
- O sistema operacional é responsável por cuidar deste ciclo de vida
- Entretanto, um aplicativo robusto se preocupa em tratar estes estados

Ciclo de vida da Activity



Ciclo de vida da Activity

- À medida que o usuário começa a sair da atividade, o sistema chama métodos para eliminá-la. Em alguns casos, essa eliminação é somente parcial.
- A atividade ainda reside na memória, como quando o usuário alterna para outro aplicativo, e ainda pode voltar ao primeiro plano. Se o usuário retornar a essa atividade, a atividade será retomada de onde o usuário parou.
- Com algumas exceções, os aplicativos são impedidos de iniciar atividades quando executados em segundo plano.

Ciclo de vida da Activity

- A probabilidade do sistema eliminar um determinado processo, com as atividades nele, depende do estado da atividade no momento.
- Em Estado da atividade e ejeção da memória, há mais informações sobre o relacionamento entre o estado e a vulnerabilidade para ejeção.
- Dependendo da complexidade de sua atividade, não é necessário implementar todos os métodos do ciclo de vida.

Ciclo de vida da Activity

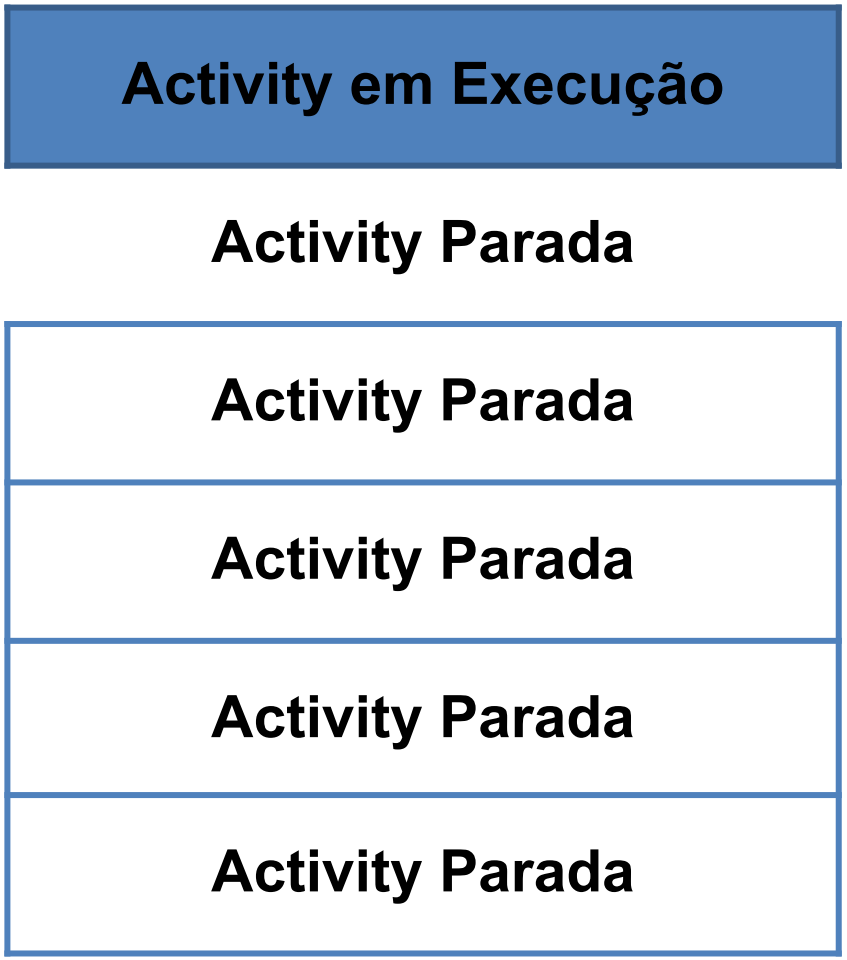
- Por exemplo, o usuário está utilizando um aplicativo de jogo no Android e enquanto isso ele recebe uma ligação
- Ao atender a ligação o SO interrompe o jogo temporariamente e o coloca sem segundo plano
- Quando a ligação termina, o SO reinicia o jogo
  - O jogo vai continuar de onde parou?
  - O estado e informações foram salvos ou foi perdido
- O Android fornece a estrutura necessária para tratar estes casos
  - Para isso é necessário entender o ciclo de vidada Activity

Ciclo de vida da Activity

- Uma Activity tem um ciclo de vida bem definido
- Cada Activity iniciada é inserida no topo da pilha de Activity
- Aquela que está no topo da pilha está em execução
  - As outras abaixo dela podem “pausadas” ou totalmente paradas
- Uma Activity pausada pode ter seu processo encerrado pelo SO para liberar recursos
  - Quando o SO decide encerrar o processo, o aplicativo pode salvar os dados para recuperar depois
- Tudo no Android é uma Activity, inclusive a Tela Inicial (Home)

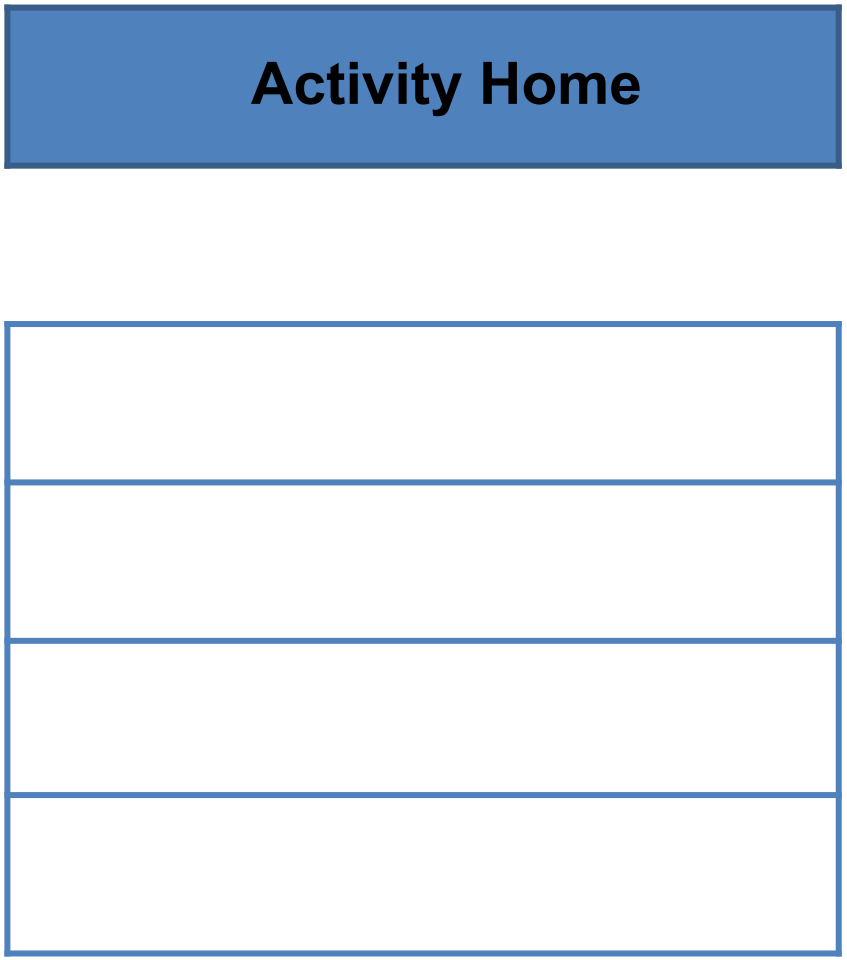
Ciclo de vida da Activity

- Pilha de Activities



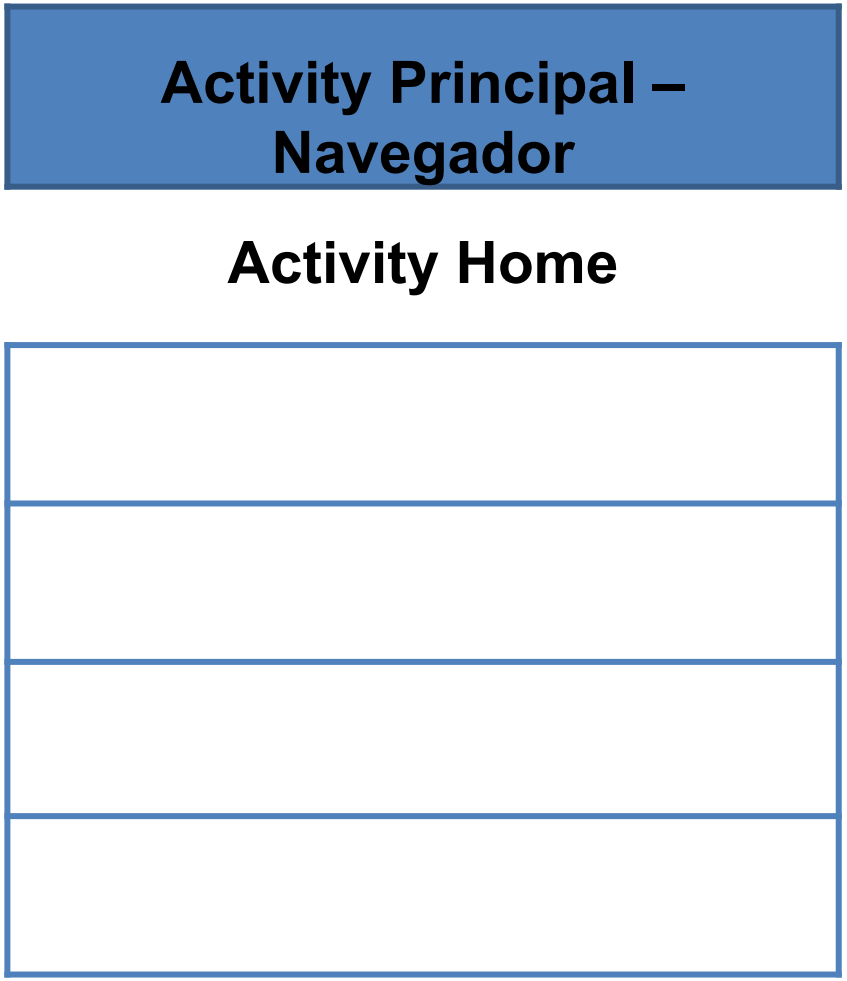
Ciclo de vida da Activity

- Usuário está na tela inicial no Android
  - Activity da Home é colocada to topo da pilha



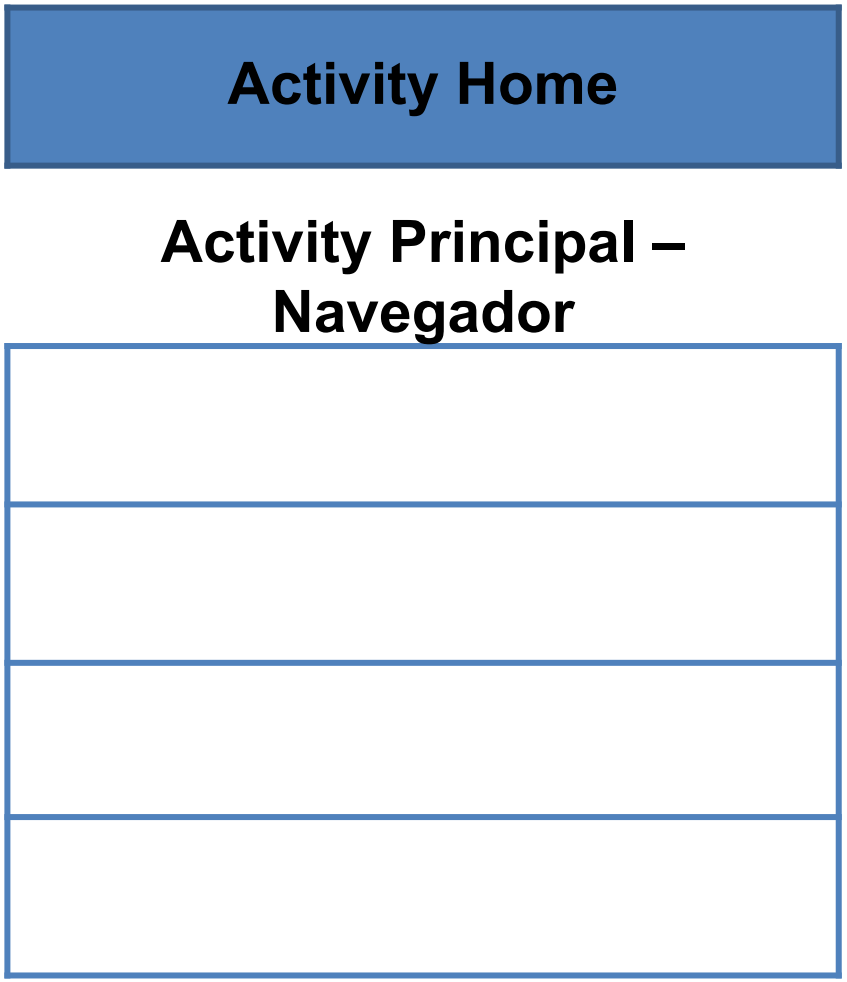
Ciclo de vida da Activity

- Usuário abre o Navegador
  - Activity da Home é parada é parada
  - Activity principal do navegador é colocada no topo da pilha



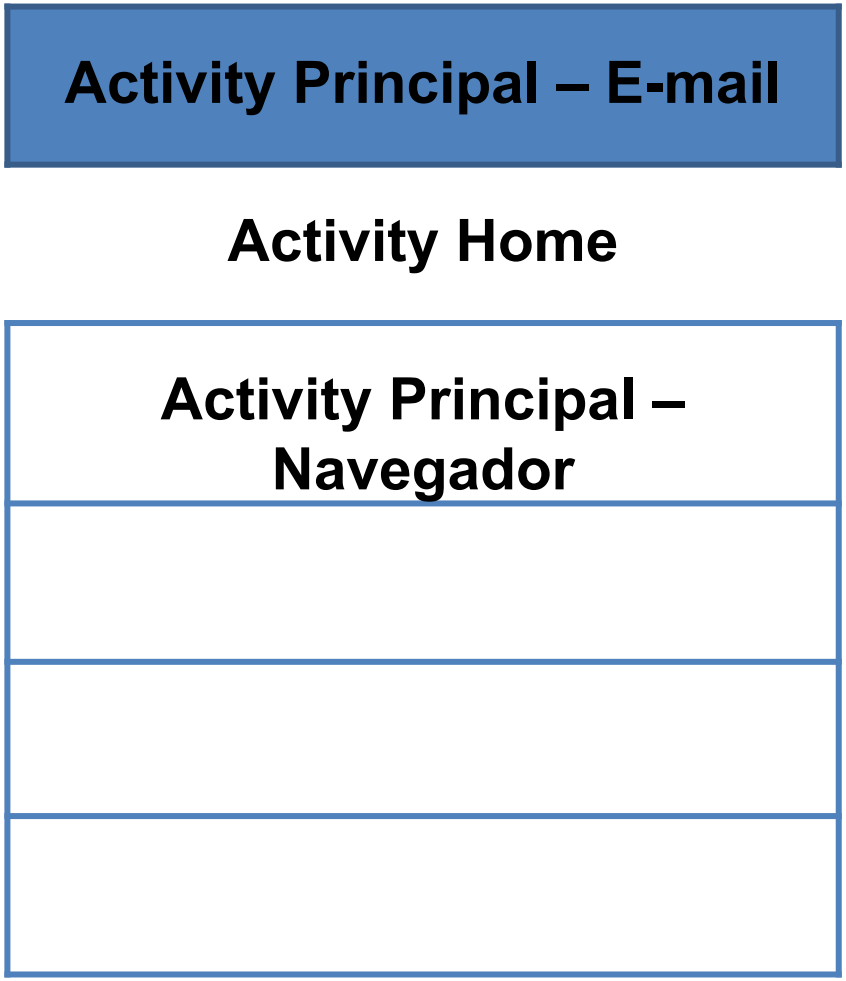
Ciclo de vida da Activity

- Usuário volta para Home
  - Activity principal do navegador é parada
  - Activity Home é colocada to topo da pilha



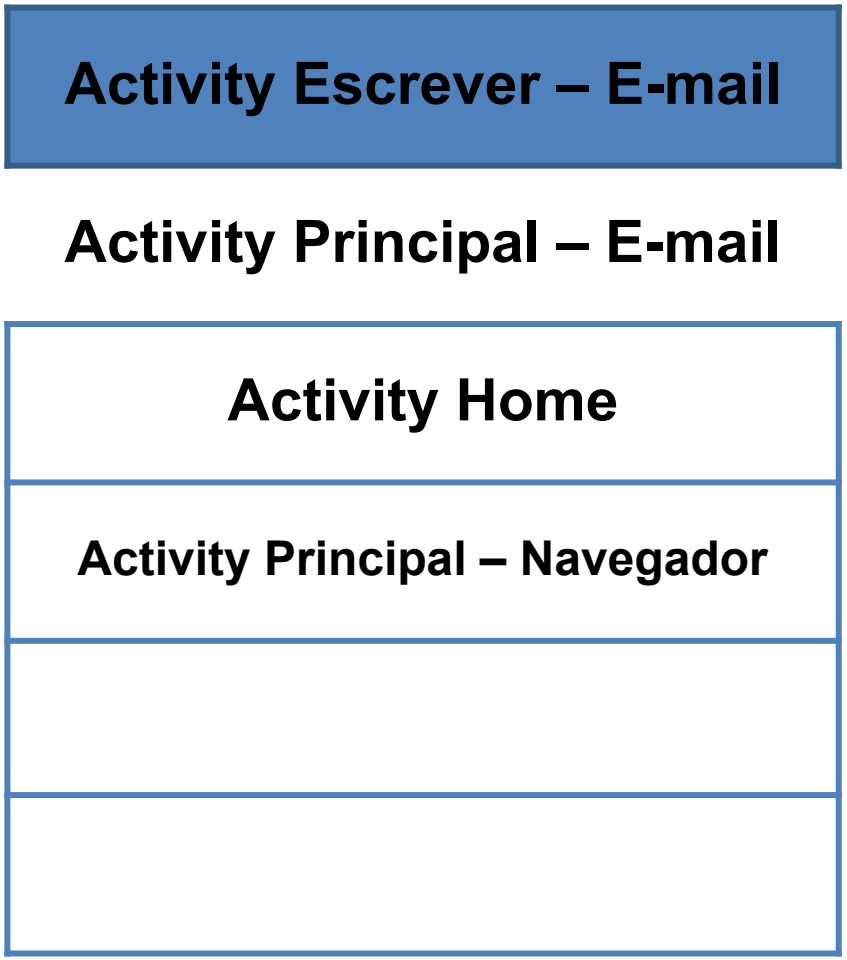
Ciclo de vida da Activity

- Usuário abre o aplicativo de e-mail
  - Activity Home é parada
  - Activity Principal do aplicativo de e-mail é colocada no topo da pilha



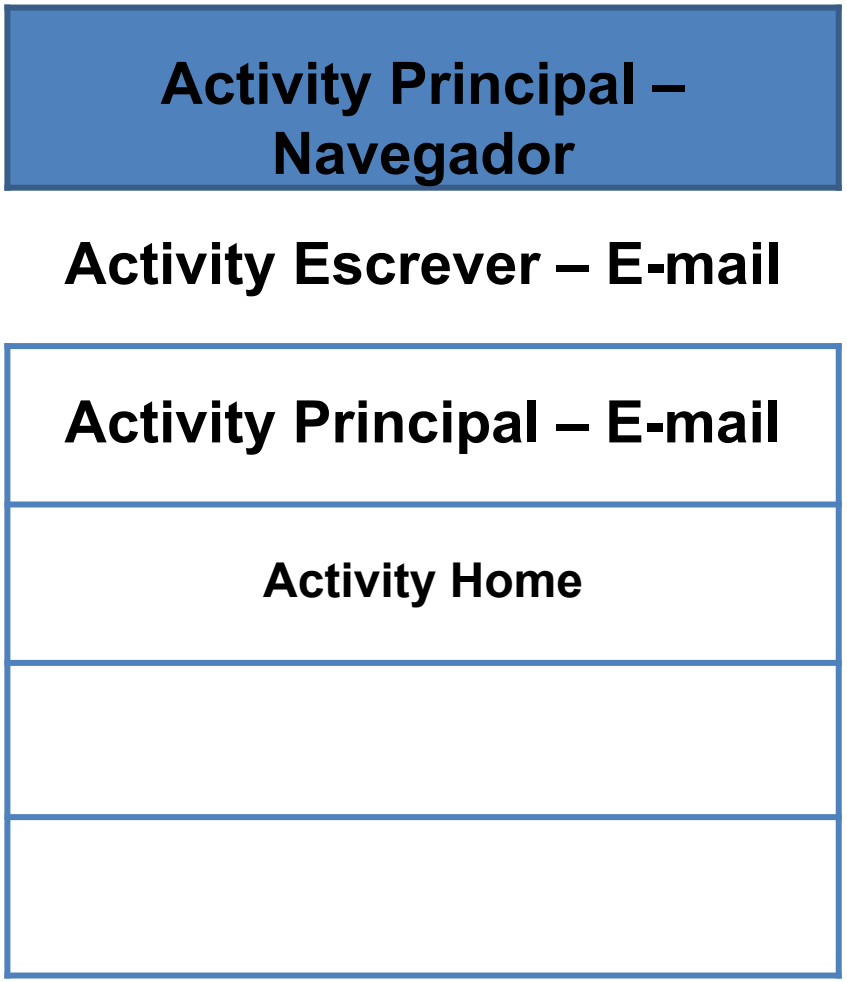
Ciclo de vida da Activity

- Usuário seleciona opção de escrever novo e-mail
  - Activity Principal do aplicativo de e-mail é parada
  - Activity de Escrever e-mail colocada no topo da pilha



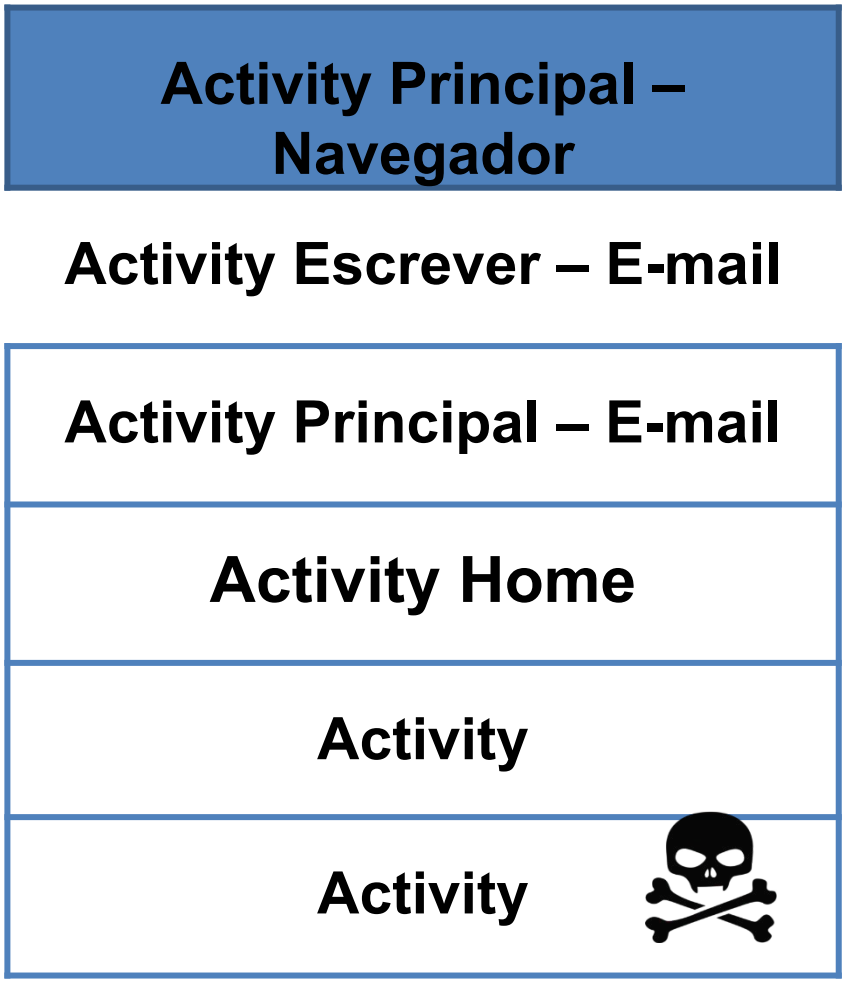
Ciclo de vida da Activity

- Usuário volta para o navegador
  - Activity de Escrever e-mail é parada
  - Activity Principal do navegador volta para o topo da pilha



Ciclo de vida da Activity

- Se a pilha estiver cheia, O SO decide qual Activity destruir para colocar uma nova no topo da pilha



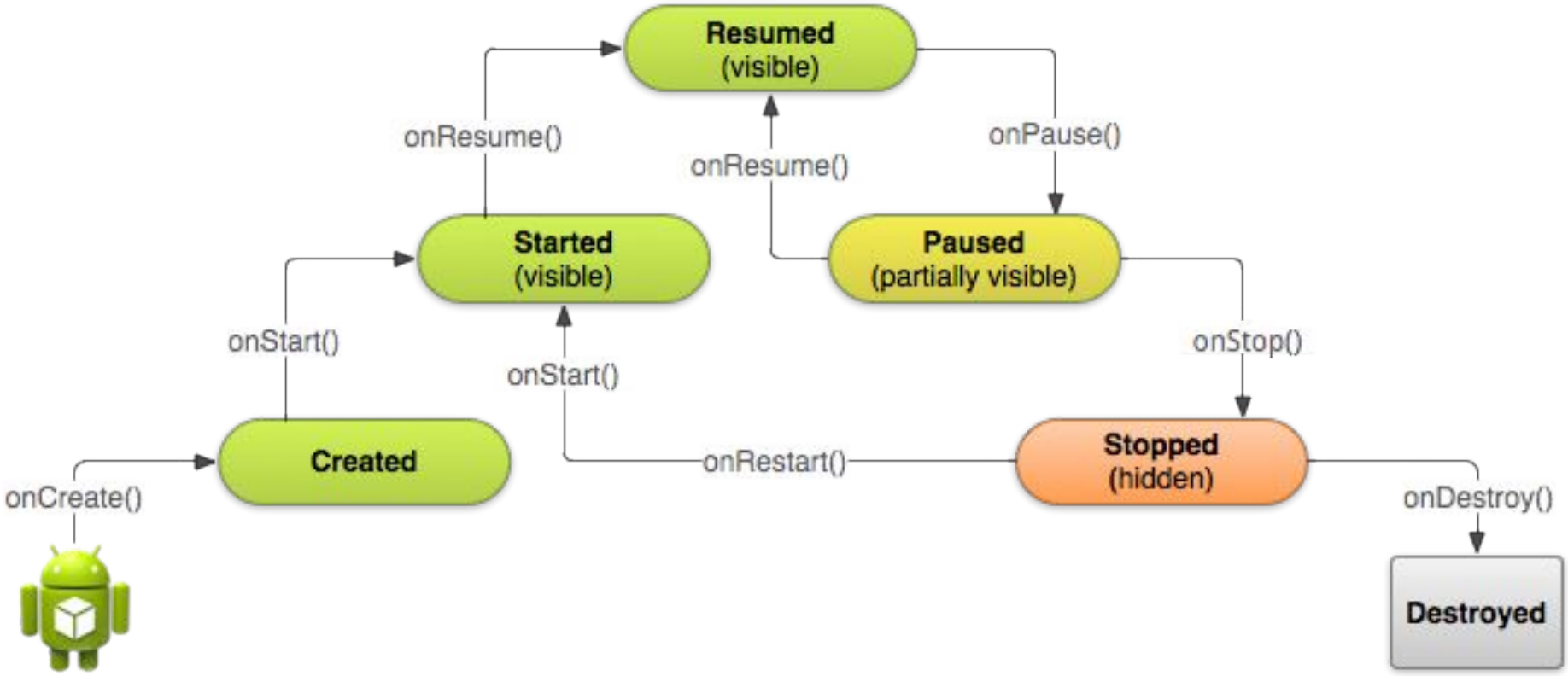
Ciclo de vida da Activity

- Mesmo os aplicativos nativos do dispositivo, como navegador, tela inicial, agenda e telefone são definidos por uma Activity
  - Estas vão ser inseridas na mesma pilha de atividades que uma Activity de um aplicativo instalado ou desenvolvido por você
  - Lembrando: todos os aplicativos funcionam da mesma forma, sobre uma mesma arquitetura

Ciclo de vida da Activity

- A superclasse Activity define métodos para controlar estes estados do aplicativo
  - onCreate(bundle?)
  - onStart()
  - onRestart()
  - onResume()
  - onPause()
  - onStop()
  - onDestroy()
- Estes métodos são sobrescritos (override) pela Activity do seu projeto

Ciclo de vida da Activity



http://developer.android.com/training/basics/activity-lifecycle/starting.html

Ciclo de vida da Activity

- onCreate(bundle?)
  - Obrigatório em toda Activity
  - É chamado somente uma vez, até o ciclo de vida ser encerrado
  - Se houver uma View, é neste método que deve ser criada exibida
  - Assim que é finalizado, chama automaticamente onStart()
- onStart()
  - É chamado quando a tela está ficando visível, após o método onCreate() ou onRestart()

Ciclo de vida da Activity

- onRestart()
  - Chamado quando a Activity foi parada temporariamente e foi reiniciada
  - Chama automaticamente o método onStart()
- onResume()
  - Chamado quando a Activity está no topo da pilha de execução
  - A Activity está executando e pronta para interação
  - Normalmente utilizado para disparar Threads que consultam WS ou BD para atualizar a tela, por exemplo

Ciclo de vida da Activity

- onPause()
  - Chamado quando algum evento ocorrer no celular
  - Activity é temporariamente interrompida
  - Neste método que os dados devem ser salvos para recuperar depois em onResume()
- onStop()
  - Chamado quando a Activity for encerrada, não mais visível pelo usuário
  - Pode ser reiniciada e chama onRestart()
  - Caso fique muito tempo parada e o SO precise de recursos, pode chamar onDestroy() para remover da pilha

Ciclo de vida da Activity

- onDestroy()
  - Encerra a execução da Activity e remove da pilha
  - Processo no SO é encerrado
  - Pode ser chamado pelo SO (liberar recursos) ou pelo aplicativo (método finish() da Activity)

Ciclo de vida da Activity

- Existem três subníveis do ciclo principal que ficam repetindo durante a execução do aplicativo.

Subnível	Descrição
Entire lifetime	É o ciclo completo entre o inicio e a destruição do activity, onde o tempo de vida é definido, está presente entre os métodos onCreate() e onDestroy(), que são chamados apenas uma vez.
Visible lifetime	Ocorre entre o onStart() e onStop(), onde a partir do método onStart() o ciclo visible lifetime é executado entre os métodos onResume(), onPause(), onStop() e onRestart(). Durante este período a activity está visível para usuário, ou esta sendo executada em segundo plano.
Foreground lifetime	Está no topo da pilha, interagindo com o usuário, ocorre entre o onResume e onPausa().

Estado da instância

- Há alguns casos em que a atividade é destruída devido ao comportamento normal do aplicativo, como quando o usuário pressiona o botão "Voltar" ou a atividade sinaliza a própria destruição chamando o método finish().
- Quando a atividade é destruída porque o usuário pressionou o botão "Voltar" ou a atividade é finalizada, o conceito do sistema e do usuário dessa instância Activity se perde.
- Nesses cenários, a expectativa do usuário corresponde ao comportamento do sistema, e você não tem trabalho extra.

Declarar atividade

```
<manifest ... >
  <application ... >
      <activity android:name=".ExampleActivity" />
      ...
  </application ... >
  ...
</manifest >
```

Exemplo prático

- Crie um novo projeto
- Crie arquivo chamado DebugActivity.kt
  - Utilize a opção new Activity Empty Activity ao clicar com o botão direito no pacote principal do seu projeto
  - Desmarque a opção que cria um arquivo de layout
  - Neste novo arquivo Kotlin, sobrescreva os métodos da Activity que controlam o ciclo de vida
    - onCreate, onStart, onRestart, onResume, onPause, onStop, onDestroy
    - Todos os métodos devem chamar o método

Exemplo prático

- No corpo de cada método, coloque a seguinte linha para mostrar uma mensagem no LogCat

```
Log.d(TAG, className + ".onMetodoCicloVida() chamado")
```

  - Troque onMetodoCliclоVida pelo nome do método onde está a mensagem
  - Na classe, crie as constants TAG e className da seguinte forma:

```
private val TAG = "LMSApp"
private val className: String
    get() {
        val s = javaClass.name
        return s.substring(s.lastIndexOf("."))
```

Exemplo prático

- Por exemplo, sobrescrever o método onStart()

```
override fun onStart() {
    super.onStart()
    Log.d(TAG, className + ".onStart()
    chamado")
}
```

Exemplo prático

- Faça a MainActivity estender DebugActivity
- Execute o aplicativo olhe o LogCat: quais métodos foram chamados?

Navegação

- Um aplicativo usualmente tem mais de uma tela.
- A navegação entre telas é feita a partir de uma Activity que está sendo executada (tela que está sendo mostrada no app), chamando a Activity que deve ser aberta
- Existem 2 métodos da classe android.app.Activity para iniciar outra Activity:
  - startActivity(intent): inicia a próxima Activity
  - startActivityForResult(intent, codigo): inicia a próxima Activity e envia um código, identificando a Activity de origem
    - Possibilita retornar informações para a primeira Activity

Navegação

```
// criar intent
val intent = Intent(context, TelaInicialActivity::class.java)
// colocar parâmetros (opcional)
val params = Bundle()
params.putString("nome", "Fabio Pereira")
intent.putExtras(params)
// fazer a chamada
startActivity(intent)
```

- O método startActivity(intent), recebe como parâmetro a intent
- Esta chamada delega ao SO a tarefa de encontrar e executar a Activity chamada

Passagem de parâmetros

- Em um aplicativo Android é possível enviar parâmetros entre as telas
- A classe responsável por armazenar os parâmetros é android.os.Bundle
  - Funciona como uma HashTable: chave=valor
  - Os parâmetros são colocados em uma instância dessa classe
- No exemplo, enviamos na chave “nome” o valor “Fabio Pereira”

```
val params = Bundle()
params.putString("nome", "Fabio Pereira")
intent.putExtras(params)
```

Passagem de parâmetros

- O método putString(chave,valor) é responsável por colocar no Bundle um parâmetro do tipo String com a chave “nome” e valor “Fernando Sousa”
- O método putExtras(bundle) da Intent é responsável por colocar os parâmetros (Bundle) na Intent
- Para que o exemplo funcione corretamente, é necessário criar a constante context na MainAcitivity, para ela armazene a instância atual da classe

```
private val context: Context get() = this
```

Passagem de parâmetros

- O objeto Bundle pode receber quantos parâmetros forem necessários, e de vários tipos diferentes
  - putBoolean, putBooleanArray
  - putByte, putByteArray
  - putChar, putCharSequence, putCharArray, putCharSequenceArray
  - putDouble, putDoubleArray
  - putFloat, putFloatArray
  - putInt, putIntArray
  - putLong, putLongArray
  - putShort, putShortArray

Passagem de parâmetros

- Agora é preciso recuperar os parâmetros enviados na próxima tela do aplicativo (TelaInicialActivity)
  - O Parâmetro estará na variável herdada intent
- Volte à TelaInicialActivity e no método onCreate() coloque o seguinte código:

```
val args = intent.extras
val nome = args.getString("nome")
Toast.makeText(context, "Parâmetro: "+ nome, Toast.LENGTH_LONG).show()
```

  - O parâmetro enviado será armazenado na variável nome

Simplificando

- O envio e recuperação dos parâmetros pode ser simplificado utilizando diretamente o objeto da Intent
  - Na criação da intent e envio dos parâmetros

```
// criar intent
val intent = Intent(context, TelaInicialActivity::class.java)

// enviar parâmetros simplificado
intent.putExtra("numero", 10)

// fazer a chamada
startActivity(intent)
```

  - Na recuperação dos parâmetros
    - Para Int, é preciso informar o valor padrão no segundo parâmetro

```
val numero = intent.getIntExtra("numero", 0)
```

Passagem de parâmetros

- Resumo: Para recuperar os parâmetros os passos são os seguintes:
  - Acessar a variável intent
  - Recuperar o objeto de parâmetros (Bundle) pelo atributo da intent extras
  - Recuperar os parâmetros desejados de acordo com o tipo
    - getString(chave)
    - getInt(chave)
    - getDoubleArray(chave)
  - Ou então diretamente da Intent
    - getStringExtra(chave)
    - getIntExtra(chave, padrao)

Resultado de uma Activity

- É possível enviar dados para a Activity anterior quando a Activity atual é finalizada
- Para isso, deve chamar o método startActivityForResult, ao invés de startActivity, quando for navegar entre as telas

```
startActivityForResult(intent, 1);
```

- Repare no segundo argumento. É o requestCode, um valor inteiro utilizado para identificar a chamada.

Resultado de uma Activity

- Na segunda Activity, é preciso informar que algo será retornado.
- Para praticar, vamos criar um botão “Sair” na view da TelaInicialActivity (activity_tela_inicial.xml) e enviar uma mensagem “Saída do LMSApp” para mostrar na tela de login
- Trate o evento de clique no botão e no método onClick

```
override fun onCreate(savedInstanceState: Bundle?) {
    // código do onCreate

    botaoSair.setOnClickListener {cliqueSair()}
}


fun cliqueSair() {
    val returnIntent = Intent();
    returnIntent.putExtra("result","Saída do LMSApp");
    setResult(Activity.RESULT_OK,returnIntent);
    finish();
}
```

Resultado de uma Activity

- Primeiro é preciso criar uma nova Intent
- Nessa intent, coloque os dados que deseja retornar, utilizando o padrão chave□ valor
- Chame o método setResult, passando o tipo de resultado e a Intent
- Por último, chame o método finish()
  - Este método força a destruição da Activity e volta para a Activity anterior

Resultado de uma Activity

- Os dados enviado pela Activity destruída podem ser recuperados pela Activity que fez a primeira chamada
- Para isso, ela deve sobrescreve o método onActivityResult
- Volte para a MainActivity e sobrescreva este método para mostrar os dados enviados no Toast, da seguinte forma

Resultado de uma Activity

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?)
    {  if (requestCode == 1) {
        val result = data?.getStringExtra("result")
        Toast.makeText(context, "$result",
        Toast.LENGTH_LONG).show()
    }
}
```

Github

- Link com os códigos gerados nesta aula:

  https://github.com/fabiodasilva500/Aulas-Mobile/tree/Aula-7-Ciclo-de-vida-Activity

Referências

- LECHETA, R. R. Android Essencial com Kotlin. Edição: 1ª ed. Novatec, 2017.
- https://developer.android.com/guide/components/activities/activity-lifecycle?hl=pt-br
- https://developer.android.com/guide/components/activities/intro-activities?hl=pt-br
- http://diatinf.ifrn.edu.br/lib/exe/fetch.php?media=cursos:superiores:tads:cursos:fic:android:04-activities_i.pdf
- http://fabrica.ms.senac.br/2013/05/ciclo-de-vida-android-activity/