



# LEAK CANARY

Biblioteca de detecção de memory  
leak para Android.

---

**01** MEMORY  
LEAK

---

**02** GARBAGE  
COLLECTOR

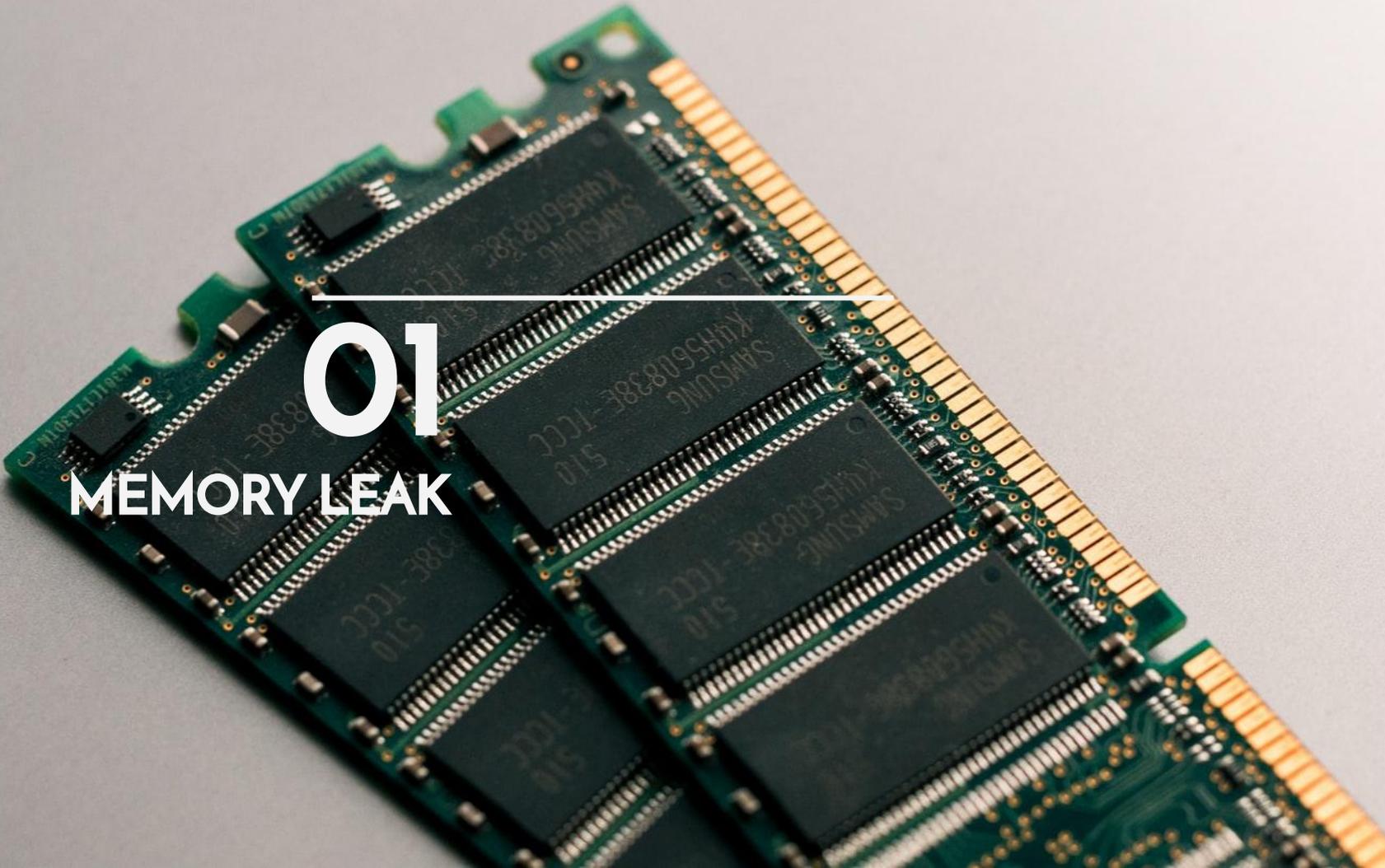
---

**03** LEAK  
CANARY

---

**04** DEMO





01

MEMORY LEAK

# MEMORY LEAK

**Memory Leak é uma falha na liberação de objetos não utilizados da memória.**

É um erro de programação que faz com que um aplicativo mantenha uma referência a um objeto que não é mais necessário. Como resultado, a memória alocada para esse objeto não pode ser recuperada, levando a uma falha no `OutOfMemoryError`.

Por exemplo, uma instância de `Activity` do Android não é mais necessária depois que o método `onDestroy()` é chamado, e armazenar uma referência a essa atividade em um campo estático impediria que ela fosse coletada como lixo.

**A maioria dos vazamentos de memória é causada por erros relacionados ao ciclo de vida dos objetos.**

- Armazenando um contexto de Activity como um campo em um objeto que sobrevive à recreação de atividades devido a alterações na configuração.
- Registrando um listener, broadcast receiver or RxJava subscription que faz referência a um objeto com o ciclo de vida e esquecendo de cancelar o registro quando o ciclo de vida chegar ao fim.
- Armazenando uma view em um campo estático e não limpando esse campo quando a view é desanexada.

# CONSEQUÊNCIAS

Apesar do programa continuar sendo executado, as consequências mais comuns são:

- Paginação em disco
- Crash da aplicação
- OutOfMemoryError

**O que potencialmente pode levar o usuário a desinstalar o aplicativo.**



---

02

GARBAGE COLLECTOR

# GARBAGE COLLECTOR

**Em Java, o Garbage Collector realiza a desalocação automática de memória, sem necessidade de ação explícita da pessoa desenvolvedora.**

Essa abordagem detecta objetos não utilizados para obter mais espaço na memória. Ou seja, se houver um objeto na heap que não contenha nenhuma referência a ele, ele será liberado da memória.





Stars: 24K  
Forks: 3.5K

**LeakCanary é uma biblioteca de detecção de vazamento de memória para Android.**

O conhecimento do LeakCanary sobre os componentes internos do Android Framework oferece uma capacidade única de diminuir a causa de cada vazamento, ajudando os desenvolvedores a reduzir drasticamente as falhas do **OutOfMemoryError**.

# POR QUE USAR

- Vazamentos de memória são muito comuns em aplicativos Android.
- OutOfMemoryError (OOM) é a principal falha na maioria dos aplicativos da loja de jogos.
- \*Case: Quando o LeakCanary foi ativado pela primeira vez no aplicativo Square Point Of Sale, vários vazamentos foram encontrados e corrigidos e, assim, a taxa de falhas do OutOfMemoryError reduziu **94%**.

# COMO FUNCIONA

1. Detectando instâncias retidas
  - Conecta-se ao ciclo de vida do Android para detectar automaticamente quando activities e fragmentos são destruídos
  - `ObjectWatcher` Android lib
2. Esvaziando a heap
  - Quando o número de instâncias retidas atinge um limite, o LeakCanary despeja o heap Java em um arquivo `.hprof`
3. Analizando a heap
  - Analisa o `.hprof` e localiza a cadeia de referências que impede que as instâncias retidas sejam garbage collected: **leak trace**. Com o rastreamento de vazamento determinado, deduzir quais instâncias estão vazando
4. Agrupando heaps
  - Reduz a cadeia de referência a uma sub-cadeia de possíveis causas de vazamento e exibe o resultado.

---

# 04

## DEMO

[Repositório no Github](#)

chrome

SAMSUNG

# COMO INICIALIZAR

Add LeakCanary to build.gradle:

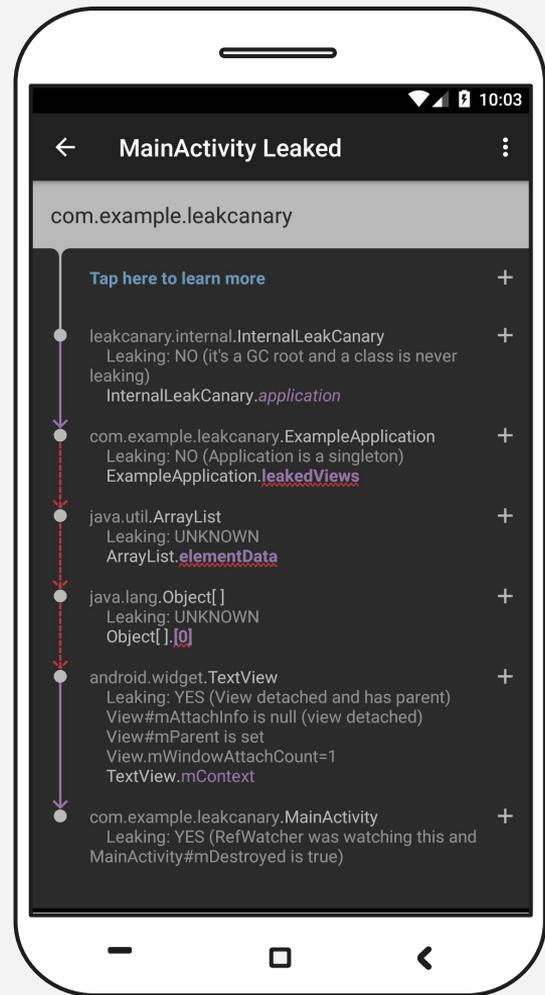
```
dependencies {  
    // debugImplementation because LeakCanary should only run in debug builds.  
    debugImplementation 'com.squareup.leakcanary:leakcanary-android:2.0-beta-3'  
}
```

Pronto, não é necessário alterar o código! LeakCanary mostrará automaticamente uma notificação quando um vazamento de memória for detectado nas **builds de debug**.

# COMO CORRIGIR

Para cada instância com vazamento, LeakCanary computa um rastreamento de vazamento e o exibe em sua interface.

- Objetos e referências
- GC Root
- Leaking instance
- Cadeia de referências
- Reduzindo a causa de um vazamento
- Heurísticas e labels



# CONFIGURAÇÃO

You can create a debug application class in your `src/debug/java` folder.

```
class DebugExampleApplication : ExampleApplication() {  
  
    override fun onCreate() {  
        super.onCreate()  
        AppWatcher.config = AppWatcher.config.copy(watchFragmentViews = false)  
    }  
}
```

Para customizar a detecção de objetos retidos em tempo de execução, atualize `AppWatcher.config`:

```
AppWatcher.config = AppWatcher.config.copy(watchDurationMillis = 6000)
```

Para customizar a heap dumping e a análise da heap, atualize `LeakCanary.config`:

```
LeakCanary.config = LeakCanary.config.copy(retainedVisibleThreshold = 3)
```

# OBRIGADO (A)



*thanks!*

Dúvidas?

Pamella Bezerra  
Victor Aguiar

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.

Outubro, 2019