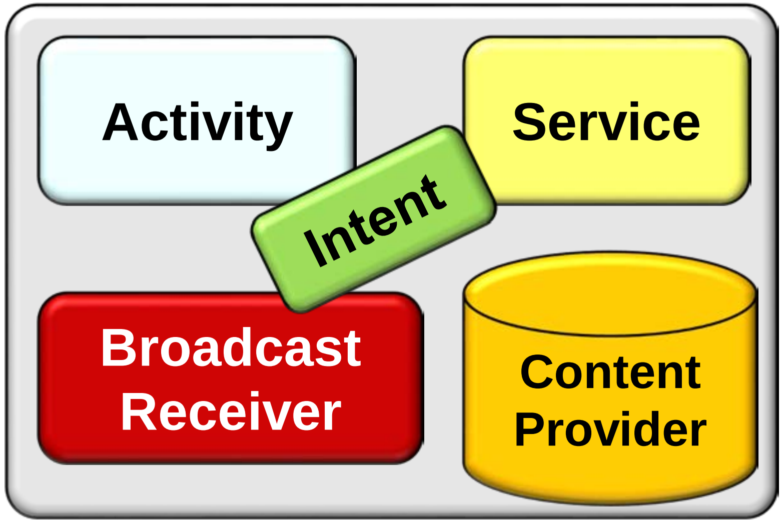
Overview of Key Android App Components

Overview of Key Android App Components

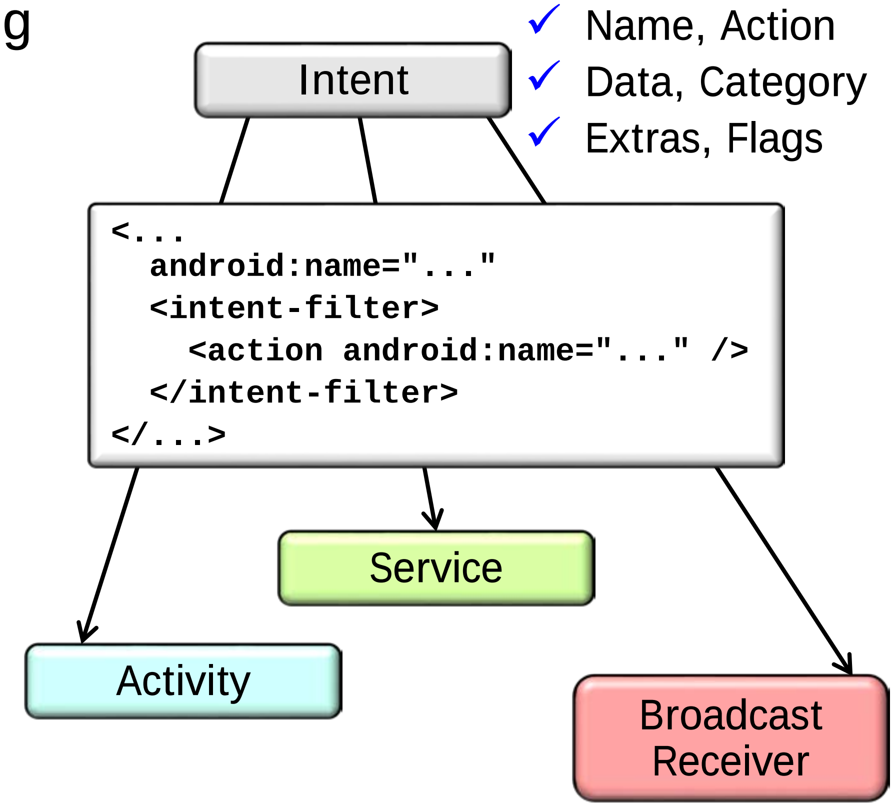
- App components are essential building blocks of mobile apps that provide various hooks via which Android can effect an app's lifecycle



See developer.android.com/guide/components/fundamentals.html#Components

Overview of Key Android App Components

- App components are essential building blocks of mobile apps that provide various hooks via which Android can effect an app's lifecycle, e.g.
  - **Intents**
    - Messages that describe an action to perform or an event that has occurred

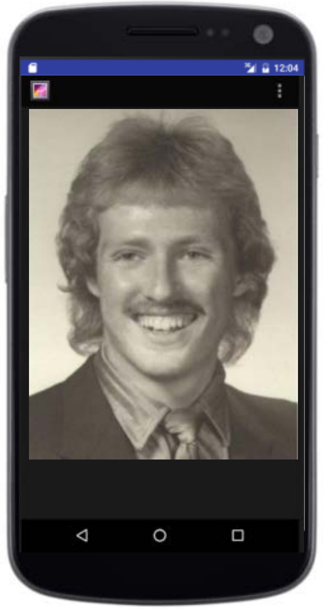


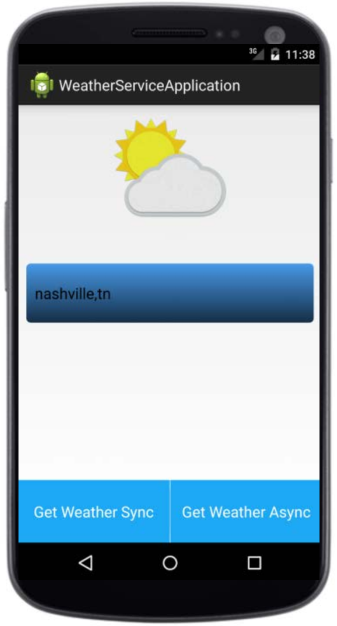
```
<...
  android:name="..."
  <intent-filter>
    <action android:name="..." />
  </intent-filter>
</...>
```

See developer.android.com/reference/android/content/Intent.html

Overview of Key Android App Components

- App components are essential building blocks of mobile apps that provide various hooks via which Android can effect an app's lifecycle, e.g.
  - Intents
  - **Activities**
    - Provide a screen within which users can interact in order to do something



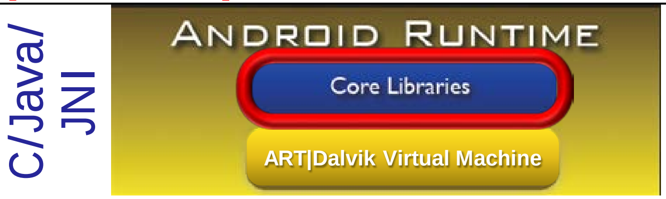


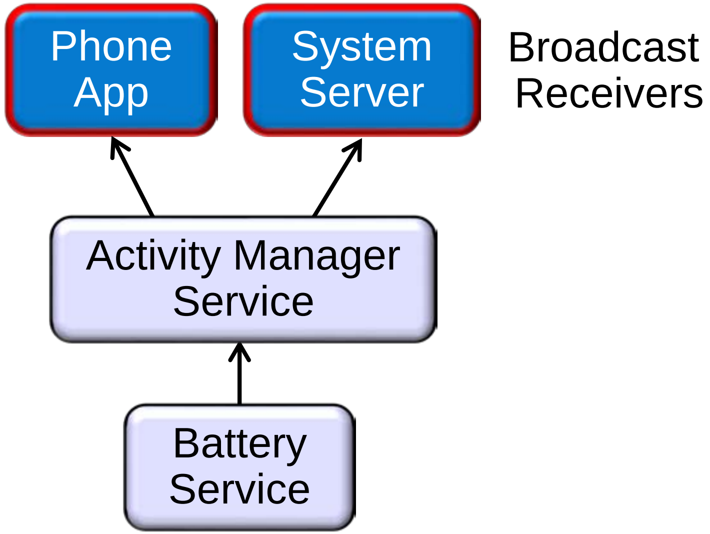
1600 Pennsylvania Avenue Nort...
General Lafayette Statue
The White House North Lawn
Eisenhower Executive Office Bldg
The White House
US Department of The Treasury
The White House South Lawn
South Lawn Fountain
President's Park
Google
The White House
1600 Pennsylvania Ave NW, Washington, DC 2...

See developer.android.com/guide/components/activities.html

Overview of Key Android App Components

- App components are essential building blocks of mobile apps that provide various hooks via which Android can effect an app's lifecycle, e.g.
  - Intents
  - Activities
  - **Broadcast Receivers**
    - Event handlers that respond to broadcast announcements

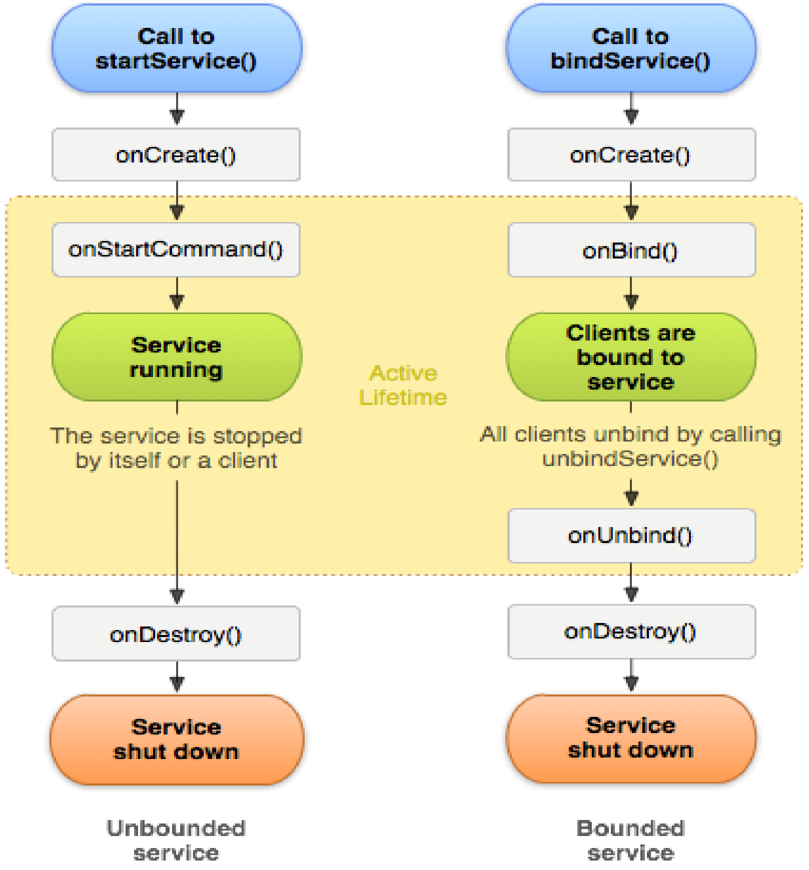




See developer.android.com/reference/android/content/BroadcastReceiver.html

Overview of Key Android App Components

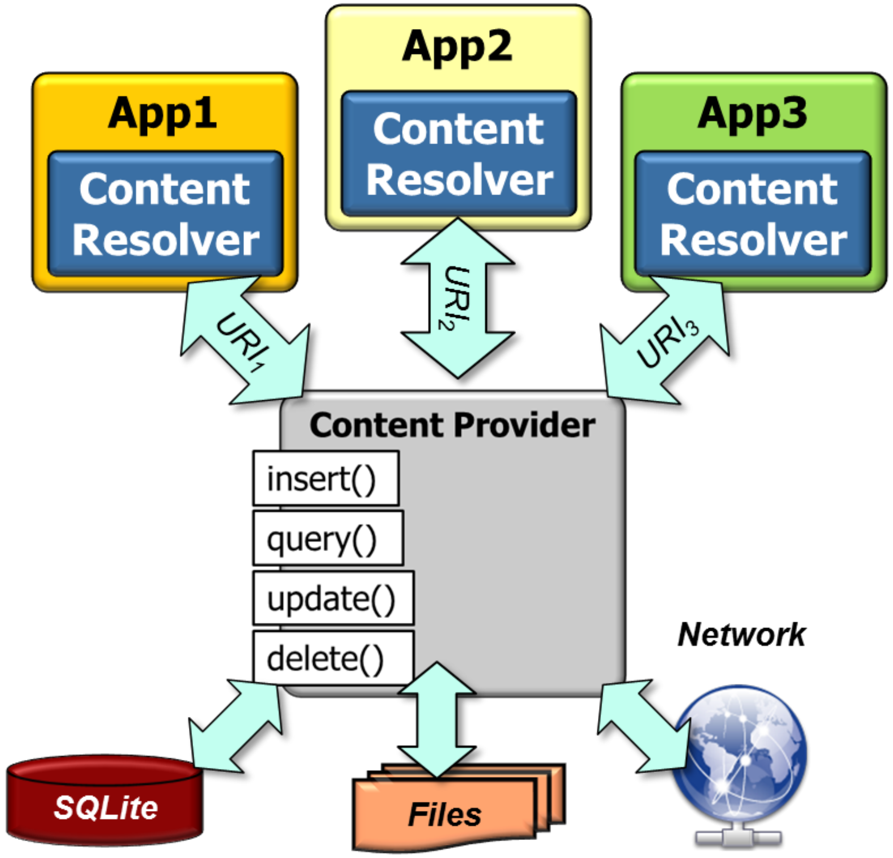
- App components are essential building blocks of mobile apps that provide various hooks via which Android can effect an app's lifecycle, e.g.
  - **Intents**
  - **Activities**
  - **Broadcast Receivers**
  - **Services**
    - Run in background to perform long-running operations or access remote resources



See developer.android.com/guide/components/services.html

Overview of Key Android App Components

- App components are essential building blocks of mobile apps that provide various hooks via which Android can effect an app's lifecycle, e.g.
  - **Intents**
  - **Activities**
  - **Broadcast Receivers**
  - **Services**
  - **Content Providers**
    - Manage access to structured data & provide data security mechanisms



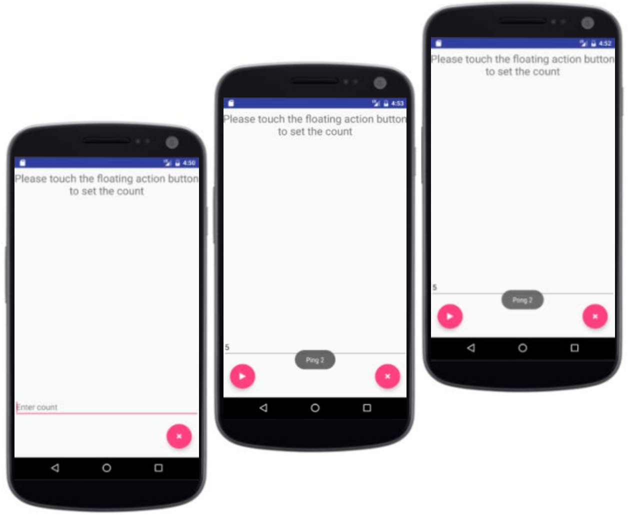
See developer.android.com/guide/topics/providers/content-providers.html

Overview of Java Threads in Android

Overview of Java Threads in Android

- Many example apps in this course use Java threads

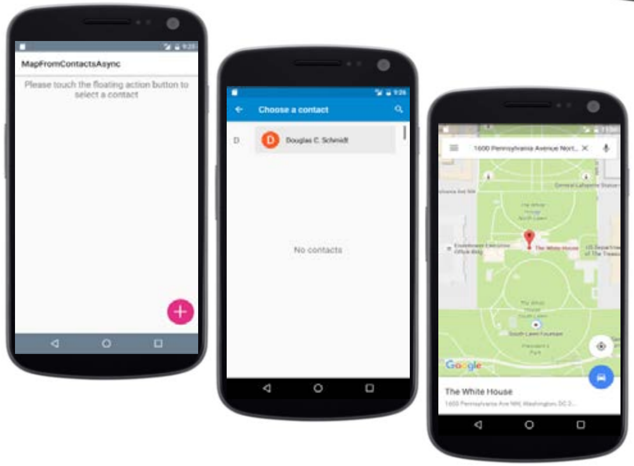
*PingPongReceivers**


Please touch the floating action button to set the count

Please touch the floating action button to set the count

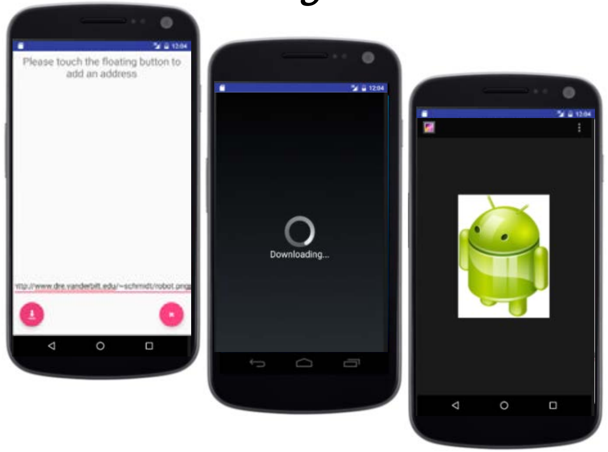
Please touch the floating action button to set the count

*MapFromContacts**


MapFromContactsAsync

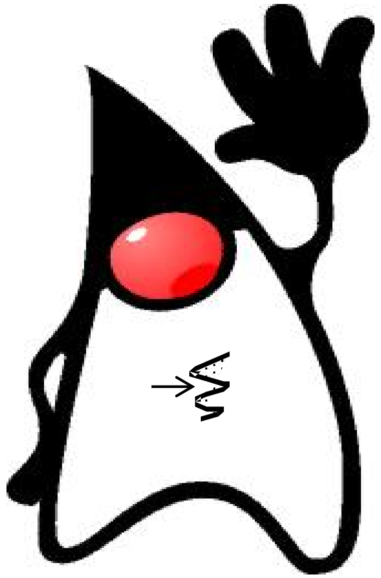
Please touch the floating action button to select a contact

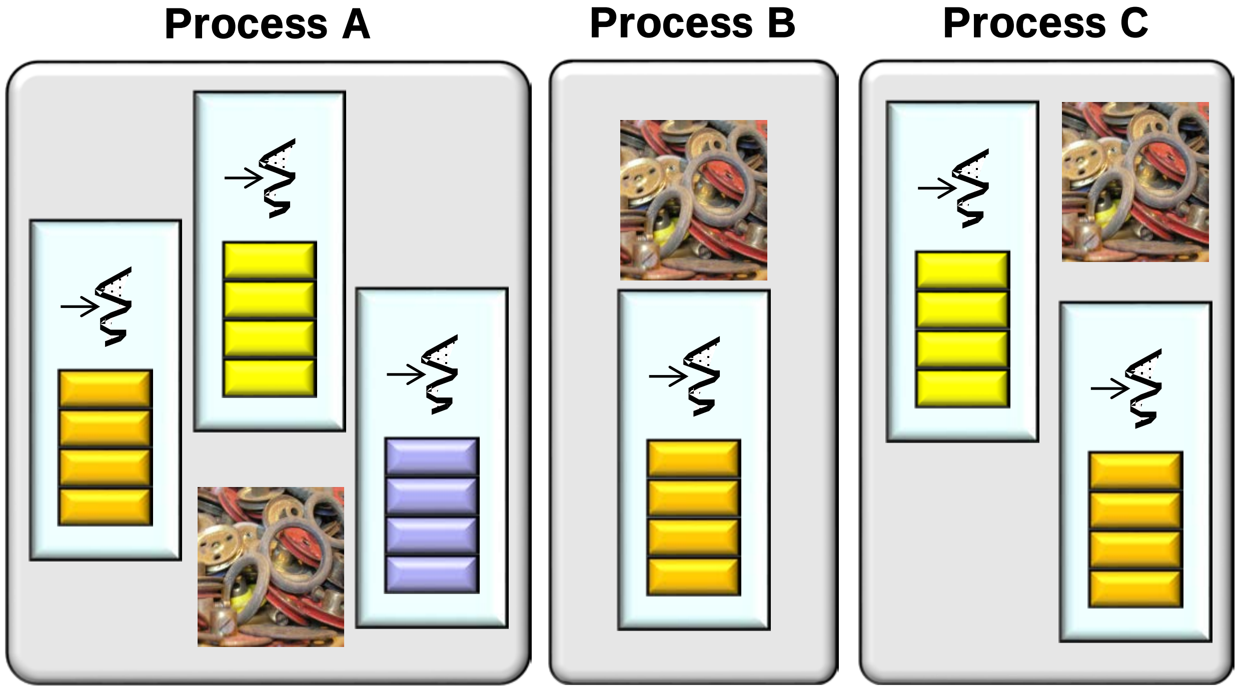
*ImageDownloader**


Please touch the floating button to add an address

Overview of Java Threads in Android

- Java threads are the smallest unit of execution for sequences of programmed instructions

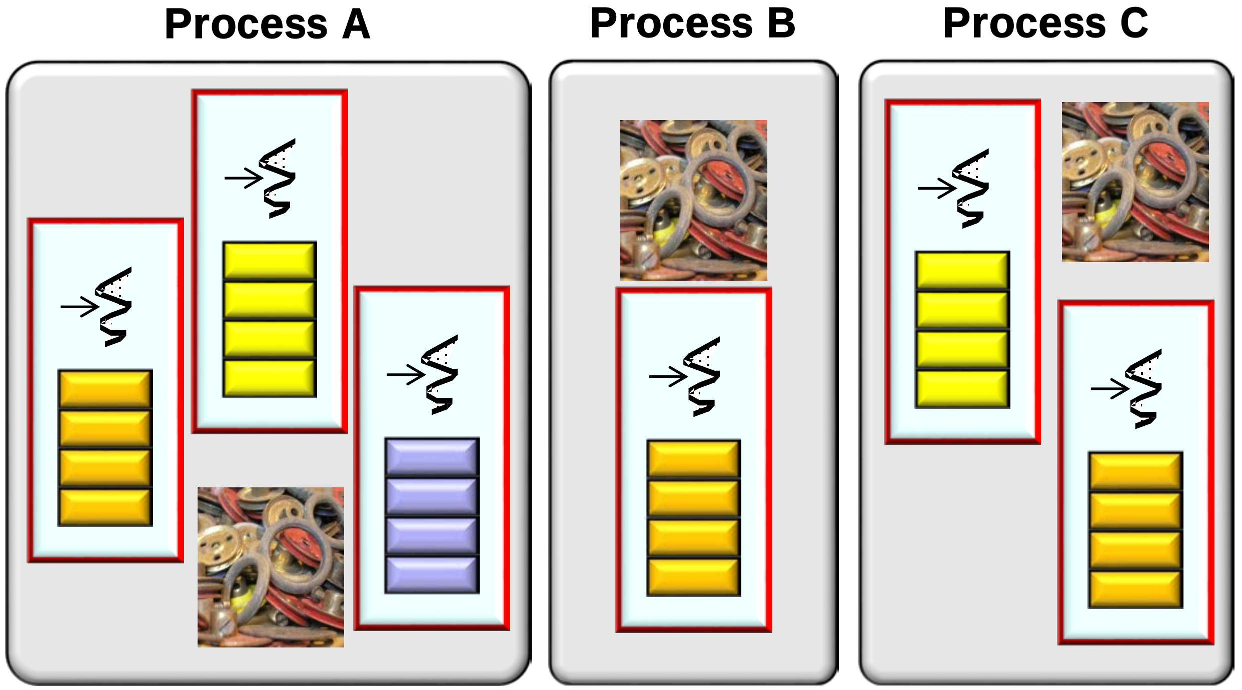




See docs.oracle.com/javase/8/docs/api/java/lang/Thread.html

Overview of Java Threads in Android

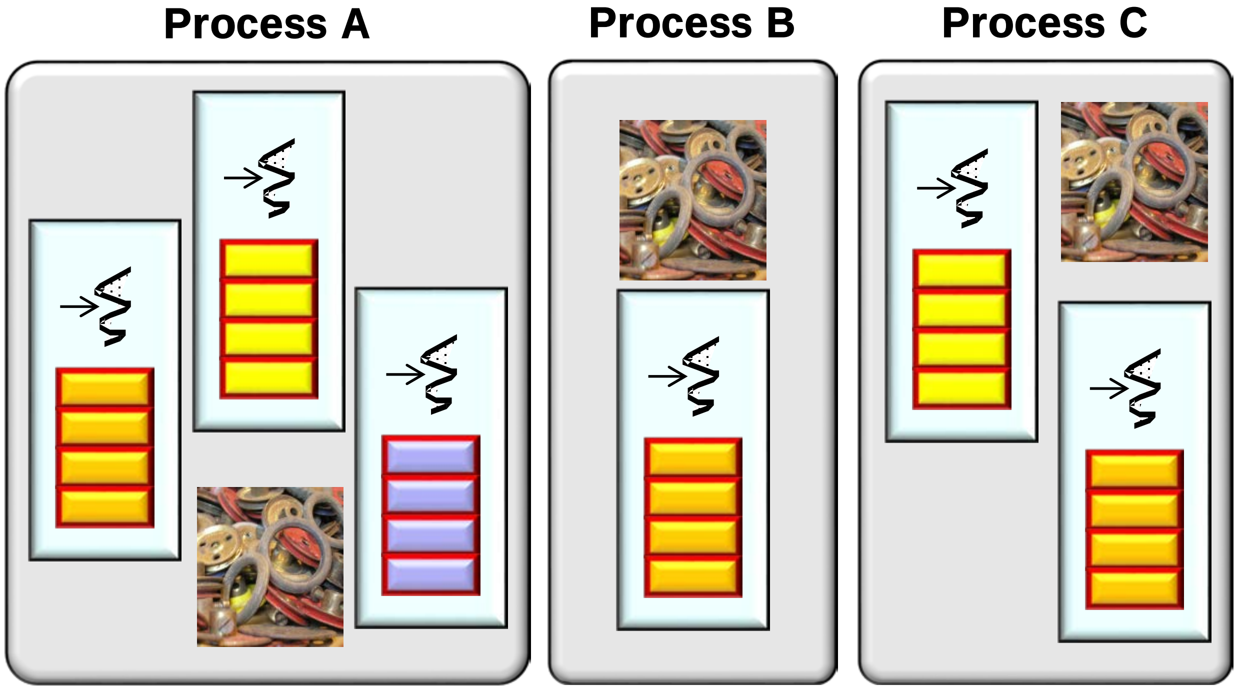
- Java threads are the smallest unit of execution for sequences of programmed instructions
  - Each process can have multiple threads that run concurrently



See docs.oracle.com/javase/tutorial/essential/concurrency/runthread.html

Overview of Java Threads in Android

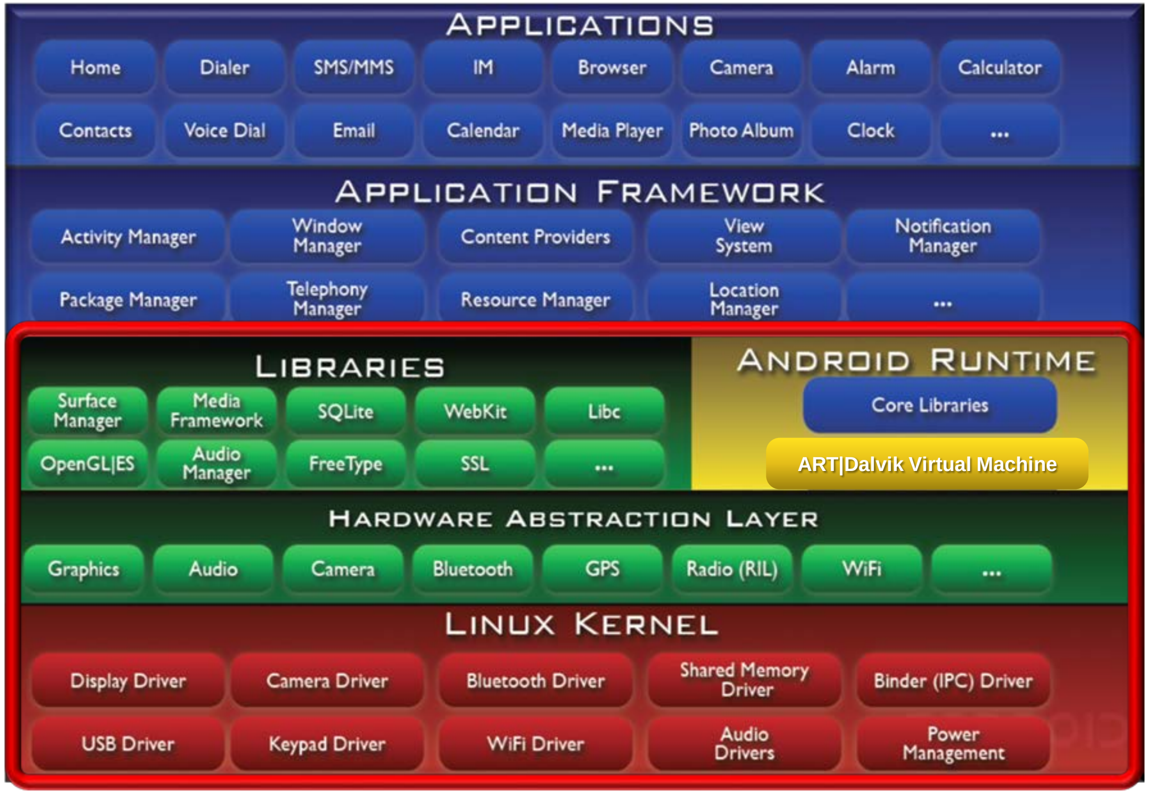
- Java threads are the smallest unit of execution for sequences of programmed instructions
  - Each process can have multiple threads that run concurrently
  - Each thread contains a call stack to keep track of method state



See en.wikipedia.org/wiki/Call_stack

Overview of Java Threads in Android

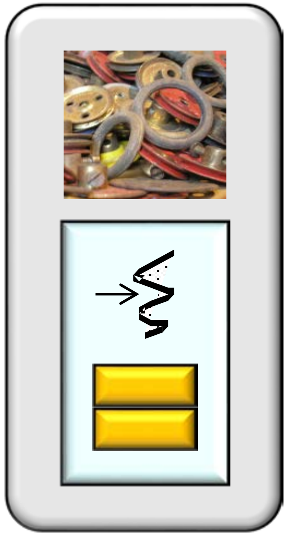
- Java threads are the smallest unit of execution for sequences of programmed instructions
  - Each process can have multiple threads that run concurrently
  - Each thread contains a call stack to keep track of method state
  - Android implements Java threads using mechanisms in various layers



Overview of Java Threads in Android

- Java threads are the smallest unit of execution for sequences of programmed instructions
  - Each process can have multiple threads that run concurrently
  - Each thread contains a call stack to keep track of method state
  - Android implements Java threads using mechanisms in various layers

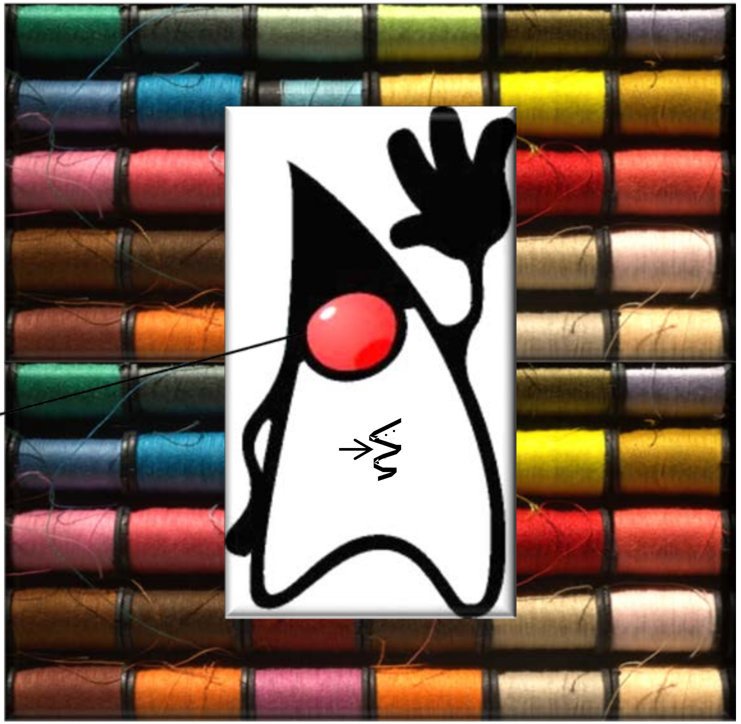
1. **`MyThread.start()`**
2. **`Thread.start()`**
3. **`VMThread.create()`**
4. **`Dalvik_java_lang_VMThread_create()`**
5. **`dvmCreateInterpThread()`**
6. **`pthread_create()`**
7. **`interpThreadStart()`**
8. **`dvmCallMethod()`**
9. **`MyThread.run()`**



- Starting a Java thread takes a non-trivial amount of time & system resources

Overview of Java Threads in Android

- Java threads must be given code to run



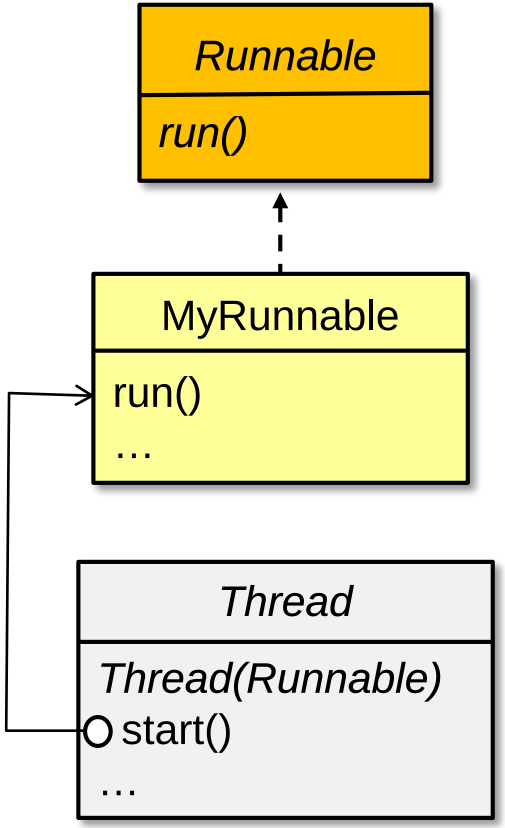
```
public void run() {
  // code to run goes here
}
```

Overview of Java Threads in Android

- Java threads must be given code to run, e.g.
  - Implement the Runnable interface

```
public class MyRunnable
       implements Runnable {
  public void run() {
    Log.d(TAG, "hello world"); ...
  }
}
final Runnable myRunnable =
  new MyRunnable();
new Thread(myRunnable).start();
```

Create/start Thread using named class object as Runnable



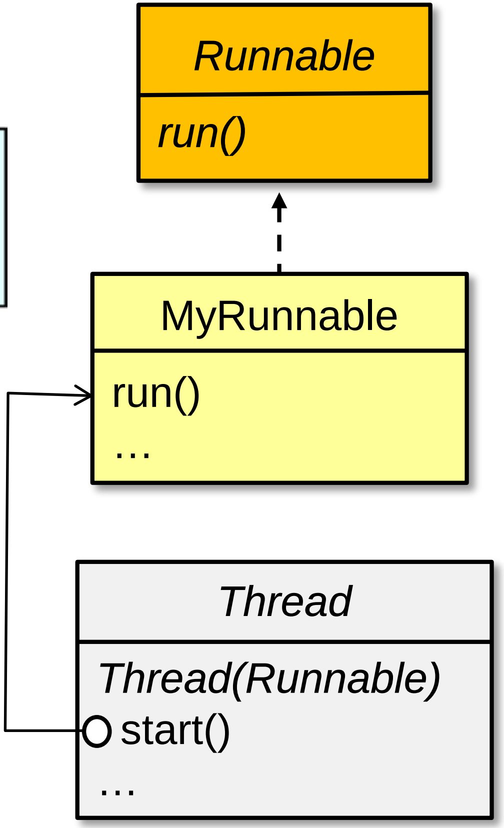
See docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html

Overview of Java Threads in Android

- Java threads must be given code to run, e.g.
  - Implement the Runnable interface

```
public class MyRunnable
       implements Runnable {
  public void run() {
    Log.d(TAG, "hello world"); ...
  }
}
final Runnable myRunnable =
  new MyRunnable();
new Thread(myRunnable).start();
```

This hook method is called back at runtime

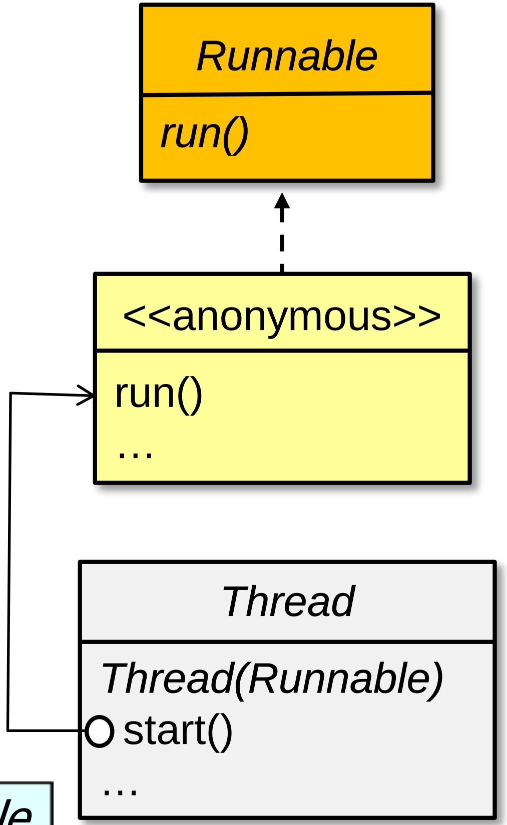


Overview of Java Threads in Android

- Java threads must be given code to run, e.g.
  - Implement the Runnable interface

```
public interface Runnable {
   public void run();
}

new Thread(new Runnable() {
  public void run(){
    Log.d(TAG, "hello world"); ...
  }
}).start();
```



Create/start a Thread using anonymous inner class as Runnable

See docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html

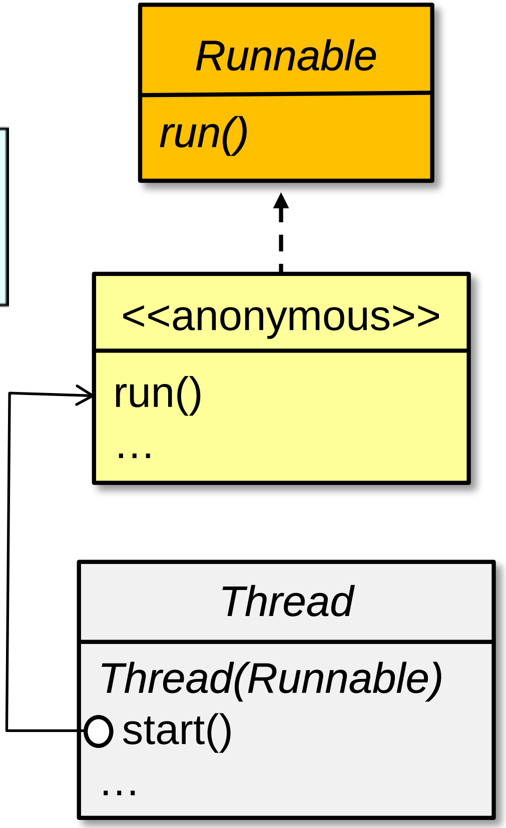
Overview of Java Threads in Android

- Java threads must be given code to run, e.g.
  - Implement the Runnable interface

```
public interface Runnable {
   public void run();
}

new Thread(new Runnable() {
  public void run() {
    Log.d(TAG, "hello world"); ...
  }
}).start();
```

This hook method is called back at runtime



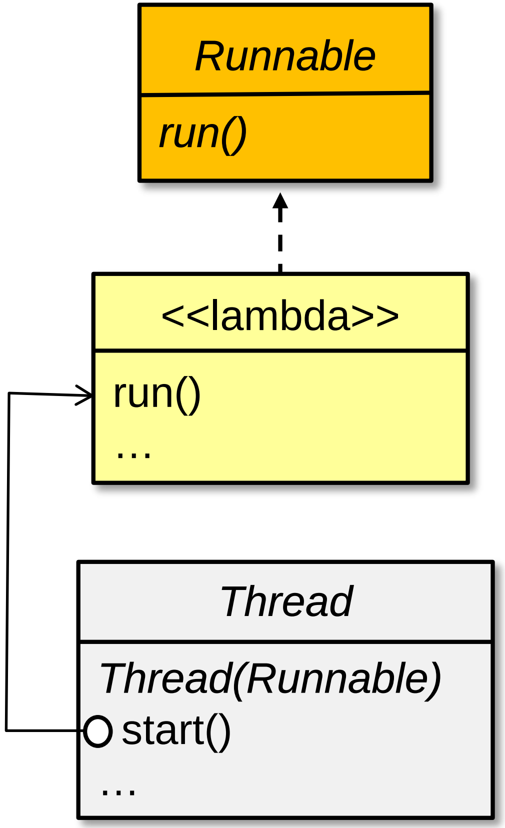
Overview of Java Threads in Android

- Java threads must be given code to run, e.g.
  - Implement the Runnable interface
  - Use Java 8 lambda expressions

```
public interface Runnable {
   public void run();
}

new Thread(() -> {
  Log.d(TAG, "hello world"); ...
}).start();
```

Create/start a Thread using a lambda expression as Runnable



See docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

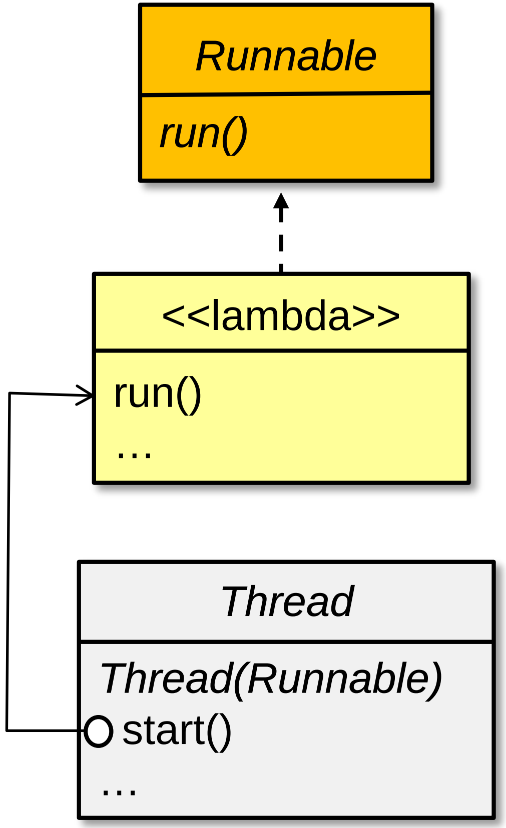
Overview of Java Threads in Android

- Java threads must be given code to run, e.g.
  - Implement the Runnable interface
  - Use Java 8 lambda expressions

```
public interface Runnable {
   public void run();
}

new Thread(() -> {
  Log.d(TAG, "hello world"); ...
}).start();
```

A lambda expression is an unnamed block of code that can be passed around & executed later



See www.drdobbs.com/jvm/lambda-expressions-in-java-8/240166764

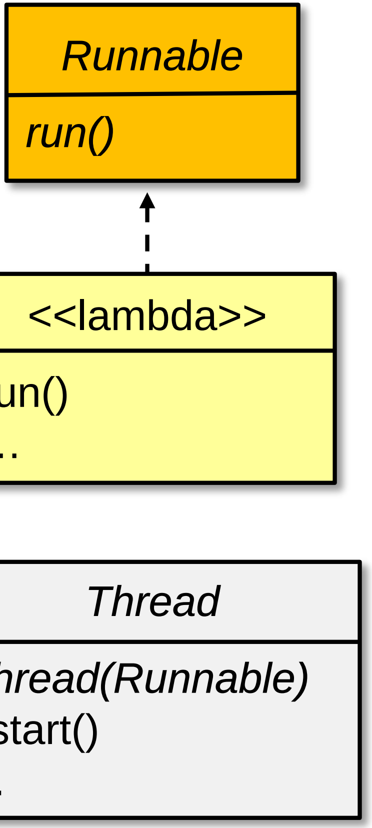
Overview of Java Threads in Android

- Java threads must be given code to run, e.g.
  - Implement the Runnable interface
  - Use Java 8 lambda expressions

```
public interface Runnable {
   public void run();
}

new Thread(() -> {
  Log.d(TAG, "hello world"); ...
}).start();
```

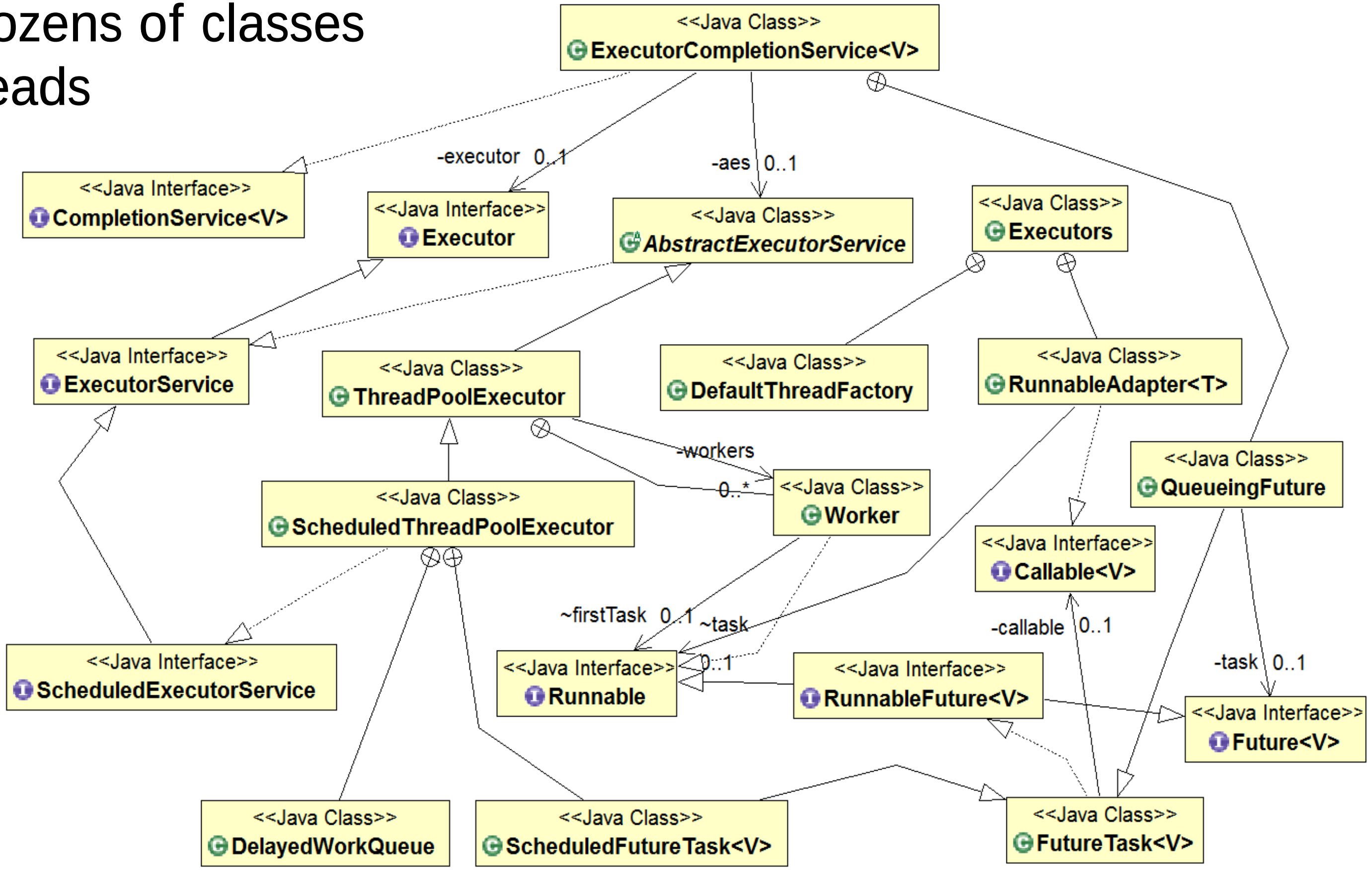
This lambda is called back at runtime



Java 8 lambda expressions are supported in Android API level 24 & beyond

Overview of Java Threads in Android

- Android contains dozens of classes related to Java threads

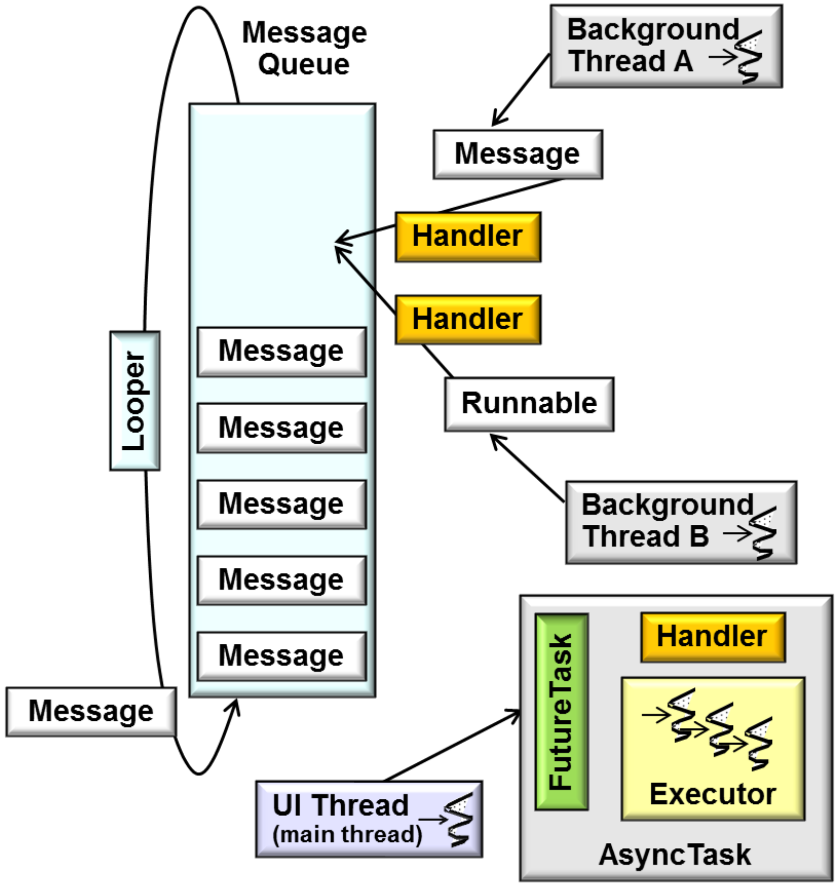


<<Java Class>> ExecutorCompletionService<V>
<<Java Interface>> CompletionService<V>
-executor 0..1
<<Java Interface>> Executor
-aes 0..1
<<Java Class>> AbstractExecutorService
<<Java Class>> Executors
<<Java Interface>> ExecutorService
<<Java Class>> ThreadPoolExecutor
<<Java Class>> DefaultThreadFactory
<<Java Class>> RunnableAdapter<T>
-workers
0..*
<<Java Class>> Worker
<<Java Class>> QueueingFuture
<<Java Class>> ScheduledThreadPoolExecutor
<<Java Interface>> Callable<V>
~firstTask 0..1
~task
-callable 0..1
-task 0..1
0..1
<<Java Interface>> ScheduledExecutorService
<<Java Interface>> Runnable
<<Java Interface>> RunnableFuture<V>
<<Java Interface>> Future<V>
<<Java Class>> DelayedWorkQueue
<<Java Class>> ScheduledFutureTask<V>
<<Java Class>> FutureTask<V>

See www.dre.vanderbilt.edu/~schmidt/LiveLessons/CPiJava

Overview of Java Threads in Android

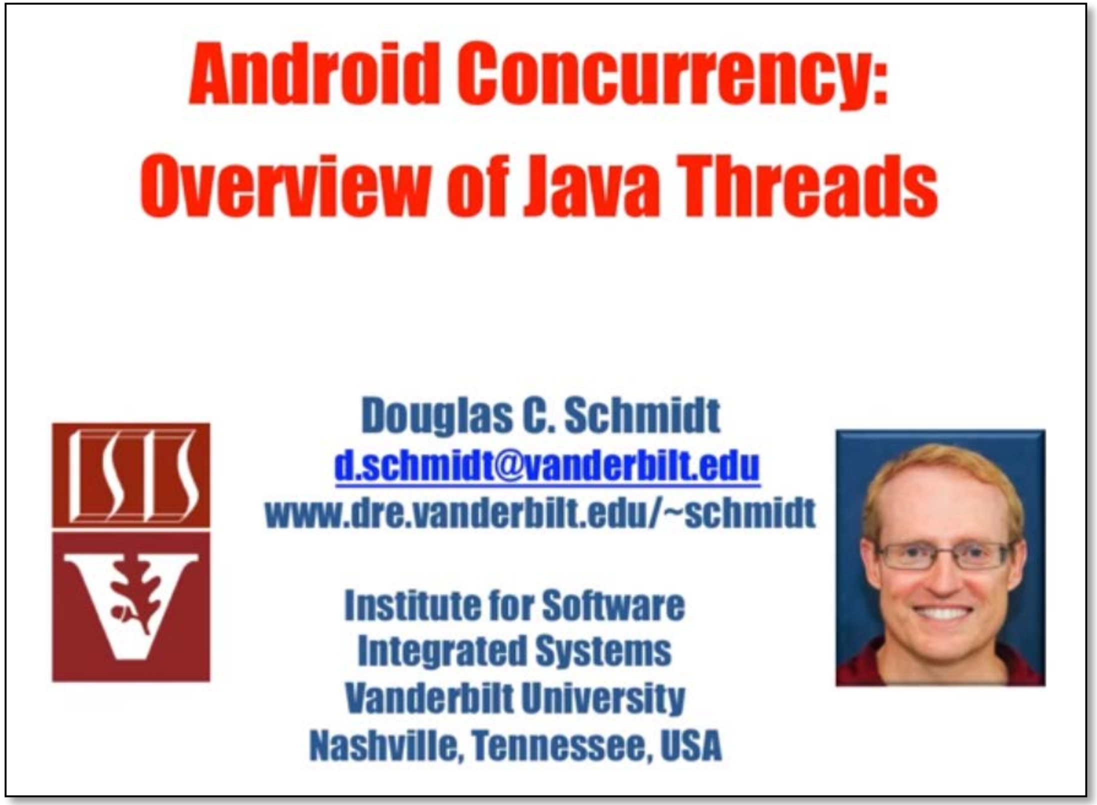
- Android contains dozens of classes related to Java threads
  - Fortunately, Android encapsulates the bulk of these Java threads classes within its concurrency frameworks



See upcoming module on Android Activities for more on its concurrency frameworks

Overview of Java Threads in Android

- More information on Java threads is available online



Android Concurrency:
Overview of Java Threads

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software
Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA

See www.youtube.com/watch?v=1YwVH-nhDtc

End of Overview of Android (Part 2): Middleware Infrastructure