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
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

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

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

*ImageDownloader*

*PingPongReceivers*

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

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()
Overview of Java Threads in Android

• Java threads must be given code to run

```java
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

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

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

```java
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

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

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

See [docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html](https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html)

Create/start a Thread using anonymous inner class as Runnable
Overview of Java Threads in Android

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

```java
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

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

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

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

Create/start a Thread using a lambda expression as Runnable
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

```java
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

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

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

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

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

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

See www.youtube.com/watch?v=1YwVH-nhDtc
End of Overview of Android (Part 2): Middleware Infrastructure