Android Applications Components

Outline

- Main Android App components
- Android OS Architecture
- Application Architecture
- Dalvik VM
- Application Manifest

App Components

- Essential building blocks of an Android app.

- Each component is an entry point through which the system or a user can enter your app. Some components depend on others.

- The most frequently used components:
  - Activity
  - Service
  - BroadcastReceiver
  - ContentProvider
  - Application

- These components come with a base class that you extend and a section in the application manifest XML file that exposes them to the Android system.

Activity Component

- UIs for an Android app should extend the Activity class, or one of its subclasses.
  - FragmentActivity, ListActivity, ActionBarActivity, AppCompatActivity, ...

- An app can have several Activities for different functions but only one Activity can be displayed at any given time.

- Delegate long-running operations to a **Service** and execute them on a separate thread.

Service Component

- Anything that doesn't involve user interface operations goes into a **Service**.

- All components run on the same main thread => make sure to execute any long-running Service in a new thread using AsyncTask

- It is recommended to create a Service for each task:
  - Service for storing data
  - Service for communicating with an online web service
  - …

BroadcastReceiver Component

- BroadcastReceiver is usefull for one thing only: **listening to system events**. You will be notified about the events after registering
  - Be notified when the battery goes below certain limit
  - Be notified when you download file
  - Be notified when a call is received
  - Be notified when connected to a wireless network
  - …
  - …
  - *Hundreds of events*

ContentProvider Component

- Use a **ContentProvider** component to manage a shared set of app data that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location
  - Through the content provider, other apps can query or modify the data if the content provider allows it

- In many cases, a simple key/value storage using **SharedPreferences** is sufficient

- You can also store data in an SQLite database using SQLite API directly from your **Services** and **Activities**

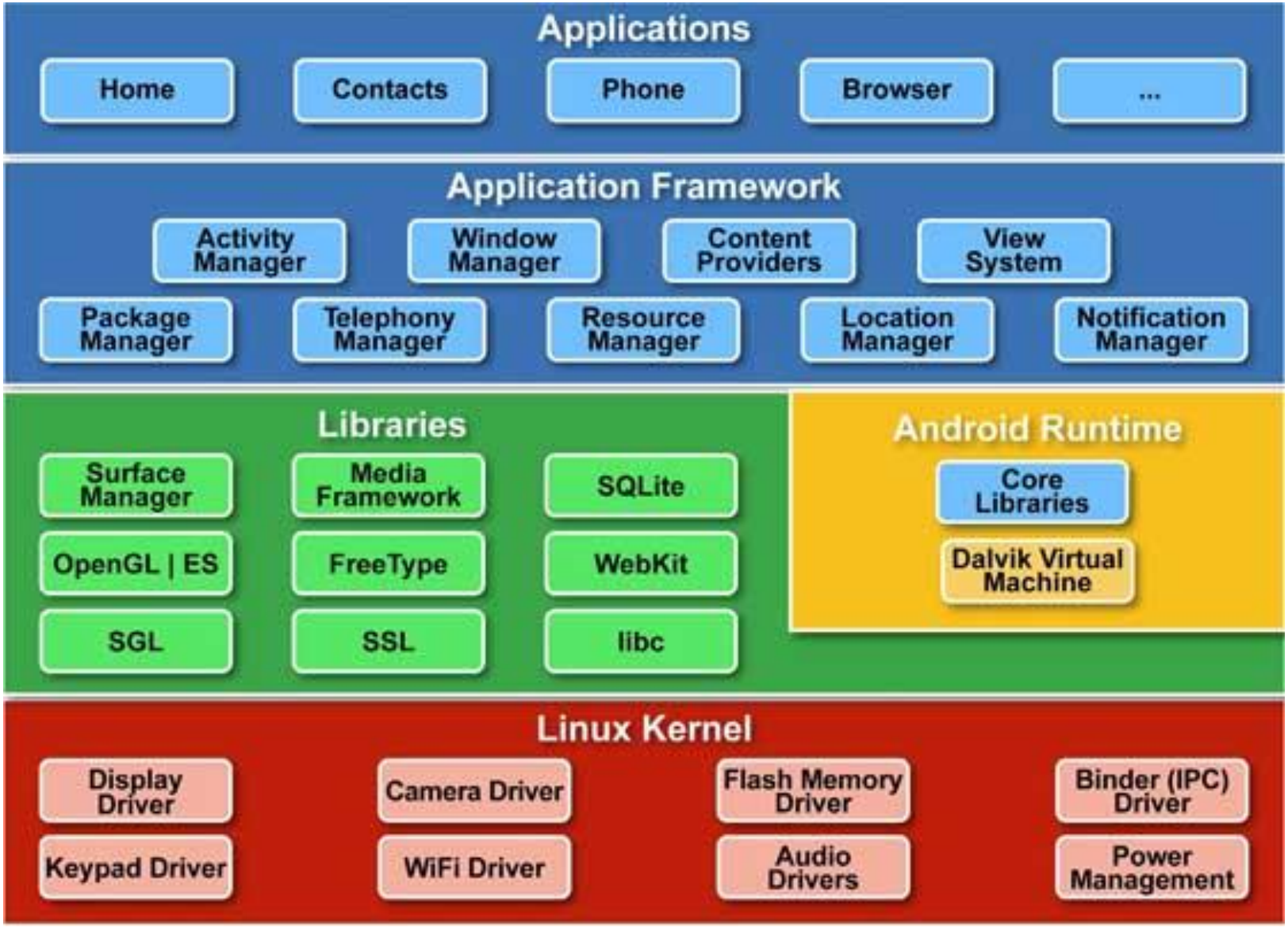
Application Component

- **A top-level component** that's created before other components (Activities, Services …)

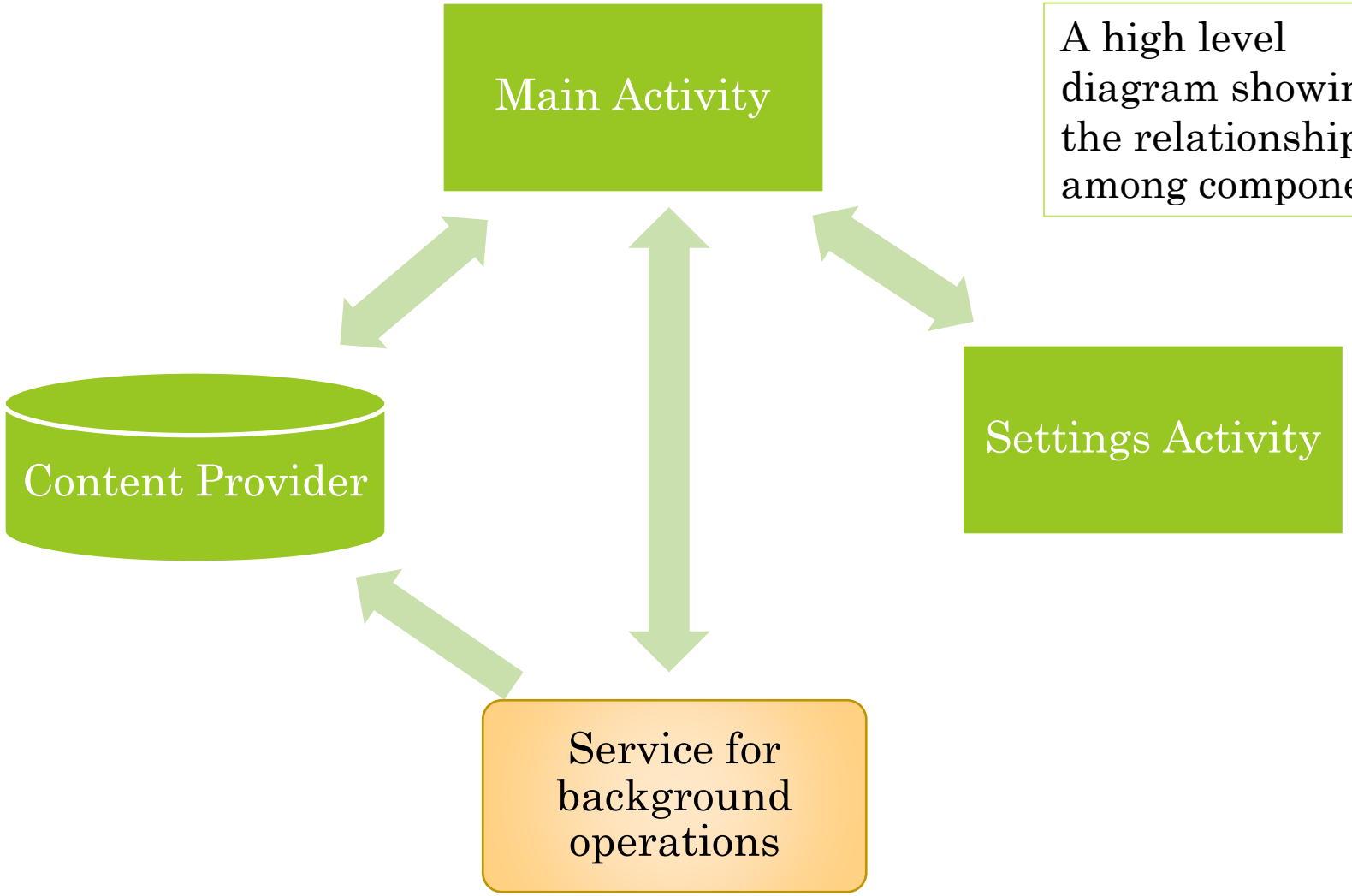
- **An Android app will always have an Application component** (created by default for you)
  - You can create a custom Application component

- You can use it to share variables and communicate across the other components within your app

Android OS Architecture



Application Architecture



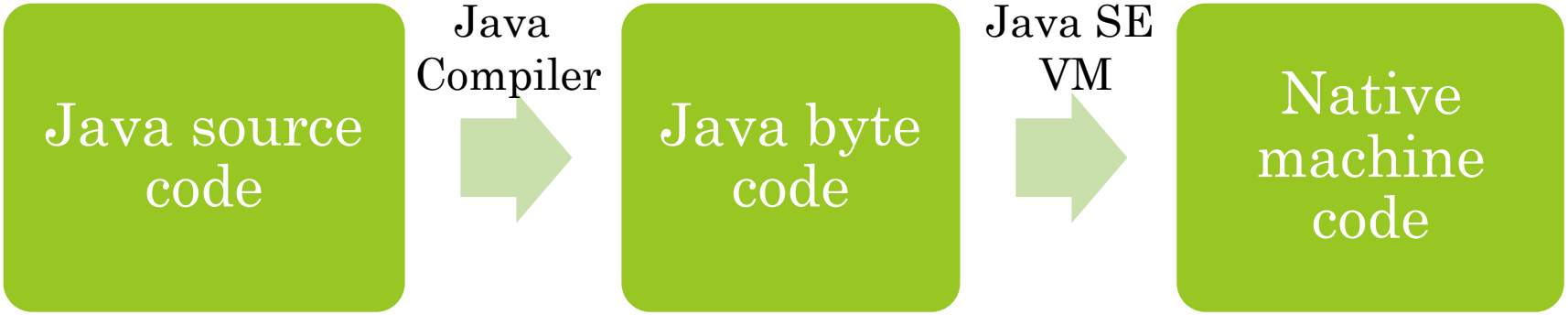
A high level diagram showing the relationships among components

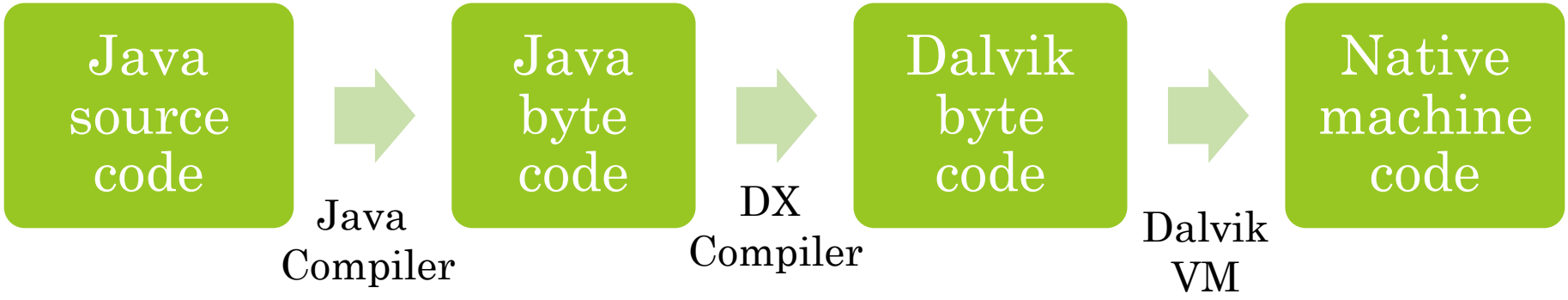
Dalvik Java vs. Java SE

- The VM that runs on an Android device is called Dalvik
  - Developed by Google
  - Suitable for CPU and memory-constrained devices

- Several differences between Java SE and Dalvik Java, mostly regarding the VM
  - Java SE uses a stack machine design
  - Dalvik is designed as a register-based machine
  - Register-based machine is 32% faster than a stack-based machine
  - An Android SDK tool called DX converts the Java SE byte-code to Dalvik byte-code

Dalvik Java vs. Java SE





Application Manifest

- The Manifest is an XML file where you define each component
- The root element is *manifest*. Example with some attributes:

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myfirstapp"
    android:installLocation="auto"
    android:versionCode="1"
    android:versionName="1.0" >
</manifest>
```

- Package name (ID of your app)
- Version information (versionCode is the value that is read by Google Play Store whereas versionName is what is presented to the user)

- installLocation attribute to control where the application is installed (internal or external storage): auto, internalOnly, preferExternal

Application Manifest

- Declare in the manifest the permissions you use.

<uses-permission android:name="*string*"
android:maxSdkVersion="*integer*" />

- **android:name**: The name of the permission
  - android.permission.CAMERA
  - android.permission.READ_CONTACTS
  - android.permission.WRITE_EXTERNAL_STORAGE
  - …

- **android:maxSdkVersion**: The highest API level at which this permission should be granted to your app. For example, beginning with Android 4.4 (API level 19), it's no longer necessary for your app to request the WRITE_EXTERNAL_STORAGE permission.

Application Manifest

- Declare in the manifest the **uses-feature** element
  - Used by Google Play to filter which application will be visible on each device.
  - For example, the following declares that your application needs telephony support
  - `<uses-feature android:name="android.hardware.telephony"/>`

  - A complete list of all the standard features in Android: http://developer.android.com/guide/topics/manifest/uses-feature-element.html#features-reference

Examples:
android.hardware.camera, android.hardware.camera.flash
android.hardware.camera.front, android.hardware.fingerprint
android.hardware.location, android.hardware.wifi,
android.hardware.Bluetooth, …….

Application Manifest

- Declare in the manifest the **supports-screens** element to specify the screen sizes supported by your app

- The following is an example of what to use for a tablet-only application

```
<supports-screens
    android:smallScreens="false"
    android:normalScreens="false"
    android:largeScreens="false"
    android:xlargeScreens="true”/>
```

Application Manifest

- Declare in the manifest the **uses-sdk** element to define the API levels your application supports.
  - minSdkVersion defines the lowest Android version you support
  - maxSdkVersion defines the highest (to avoid!)
  - targetSdkVersion tells a device which API level you're targeting
  - Example:

```
<uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />
```

Application Manifest

- The Application component is represented by the **application** element in the manifest file. Example:

```
<application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:description="@string/app_description"
        android:theme="@style/AppTheme"
        android:allowBackup="true"
        android:backupAgent=".MyBackupAgent">

<!-- Activities, Services, receivers and providers go here ... -->
</application>
```

- You can provide your custom Application class using android:name attribute. If not the system default will be used.