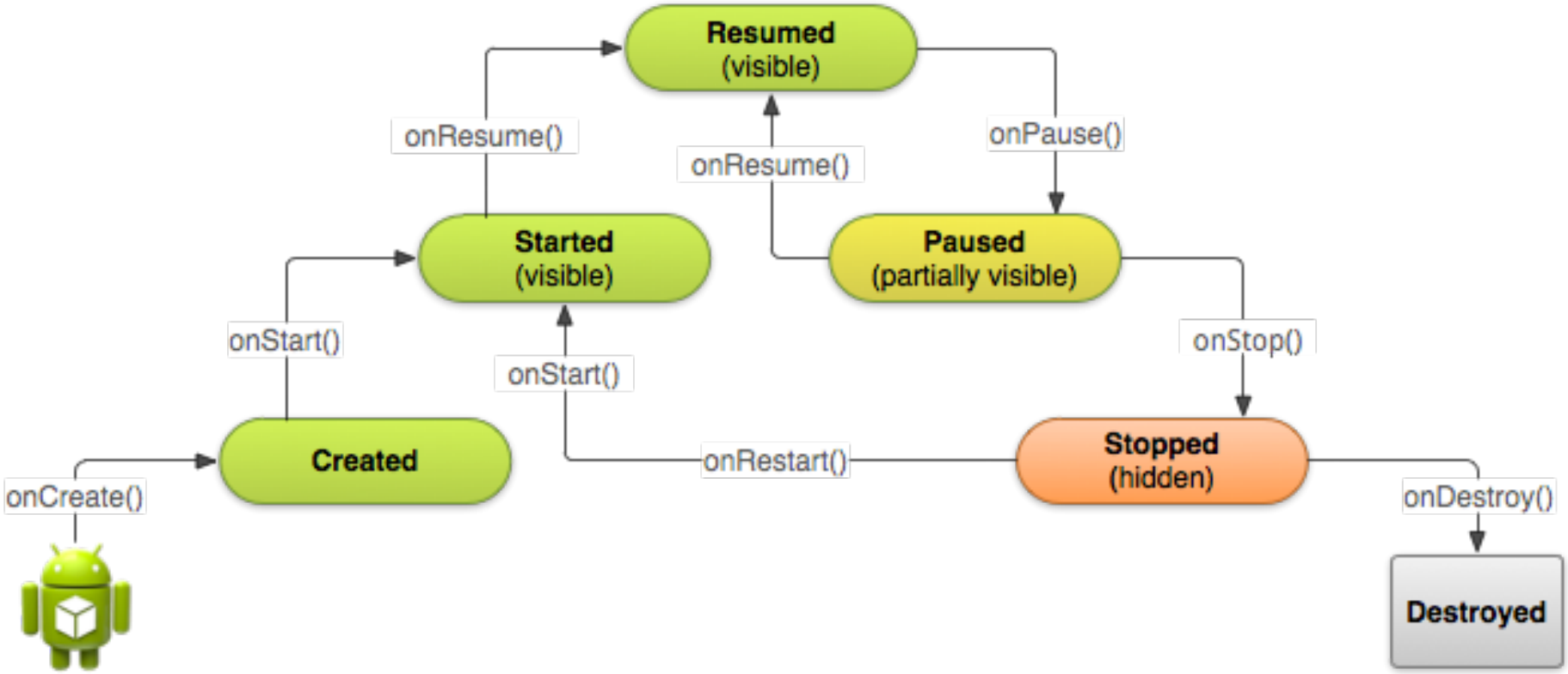
Options for storing application data permanently

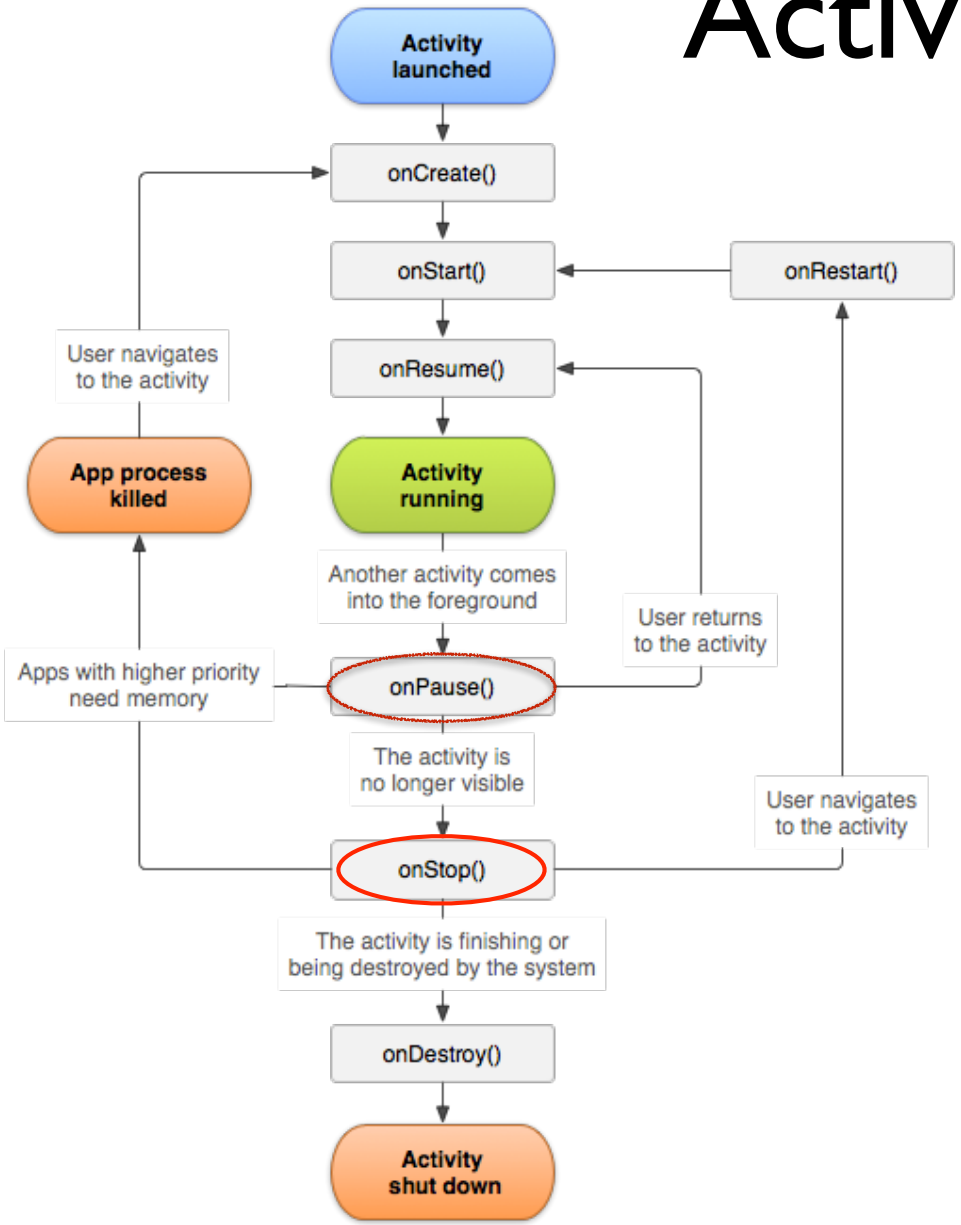
- Shared preferences
  - Key-Value -pairs, use specific API (see http://developer.android.com/reference/android/content/SharedPreferences.html)
- Internal and external storage
  - Read/write files using normal Java file I/O into either device memory or external memory (memory card). (Note permission needed for the latter option)
- (Network)
- (SQLite database)
- See http://developer.android.com/guide/topics/data/data-storage.html for overview

Activity Lifecycle

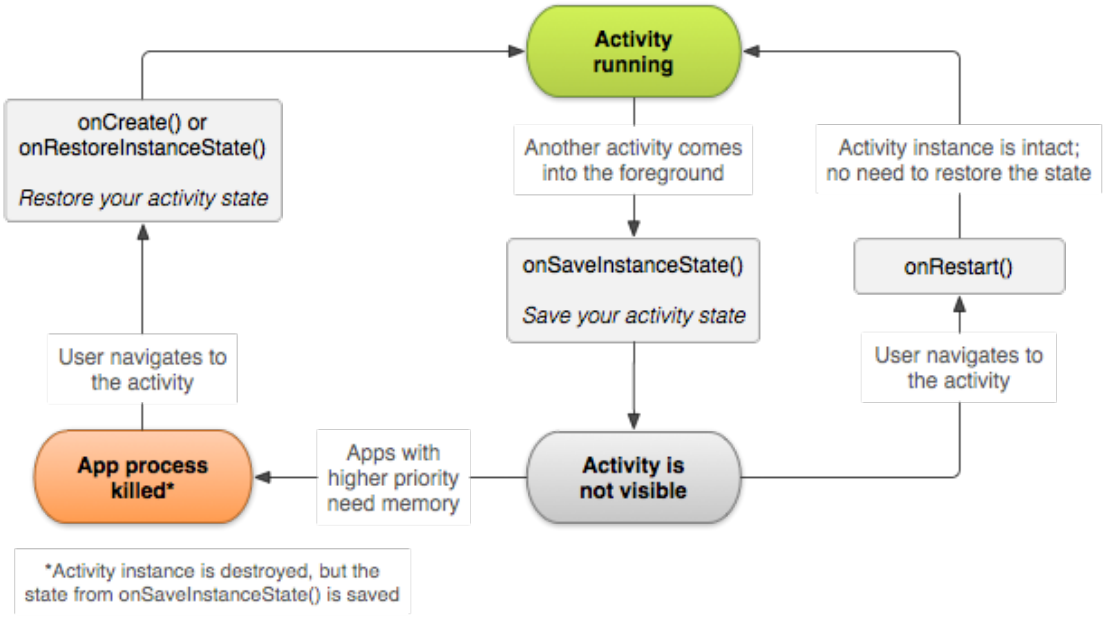


- http://developer.android.com/training/basics/activity-lifecycle/starting.html

Activity lifecycle



instance state callbacks are not part of the activity lifecycle, do not implement persistence there



So, when to restore & save?

- `onResume()` — `onPause()`
  - should be fairly quick operation, saving to db not feasible
  - but last sure callback in pre-3.0 devices
- `onStart()` [`onRestart()`] — `onStop()`
  - longer-lasting ops possible
  - last sure callback in 3.0 and later
- `onDestroy()`
  - is not guaranteed to be called before app vanishes

Intent mechanism

- Intent mechanism is intended for late (run-time) binding of application components
  - An activity does not necessarily need to know the exact activity it invokes to, for example, show some data. Instead, decision can be made at run-time
- Intent object holds the information needed to activate another component (the are some other uses for Intents, more on those later)
- Activities, services, and broadcast receivers are activated through intents.
- One activity often starts the next one by calling `startActivity()` or `startActivityForResult()` (in both cases passing Intent object as parameter) , latter if it expects a result back from the activity it's starting
- Two main types
  - *Explicit* - intent targeted to specific, named, component
  - *Implicit* – intent targeted to any component that can handle it

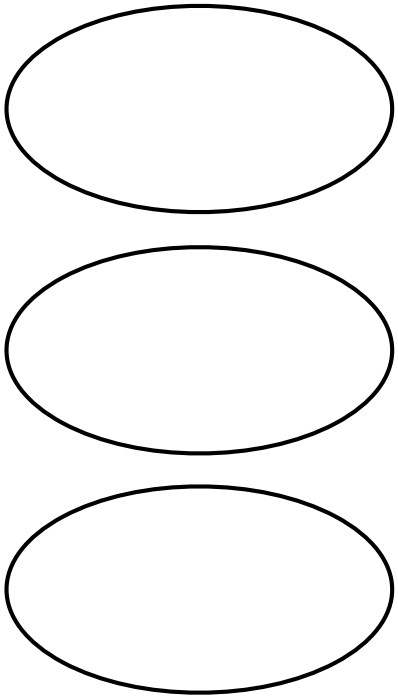
Intent object

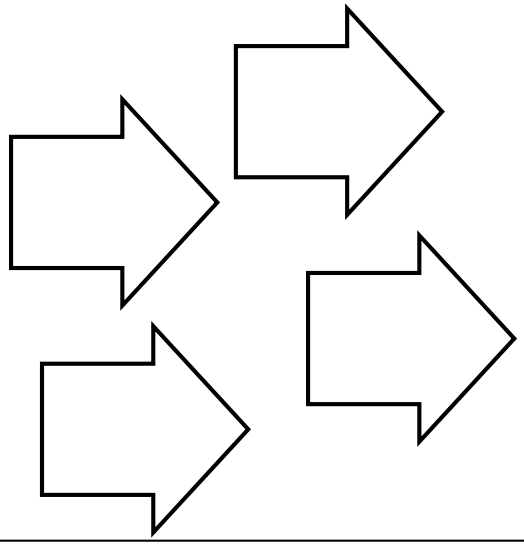
- Intent object contains information to the component that receives the intent, and information to the Android system to help it deliver the intent to the most fitting recipient (such as the category of component that should handle the intent)
- Intent objects contain some of the following attributes:
  - *Component name* - name of the component that should handle the intent (explicit intent). If component name is present in an intent it is the only needed information to address intent.
  - *Action* - action to be performed (ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.)
  - *Category* - A string containing additional information about the kind of component that should handle the intent (CATEGORY_BROWSABLE, etc.)
  - *Data* - The URI of the data to be acted on and/or the MIME type of that data.
  - *Extras* - This is a Bundle of any additional information (key-value pairs). Extras do not have an effect on intent addressing

Intent filters

- To inform the system which *implicit* intents they can handle, activities, services, and broadcast receivers can have one or more *intent filters*.
- Each filter describes a capability of the component - specifying the intents that the component is willing to receive.
  - Filters are specified using action, category and data attributes
- Since the Android system must know about the capabilities of a component before it can launch that component, intent filters are normally set up in the `AndroidManifest.xml`
  - When a new application is installed in the device, the system updates its internal intent filter list to reflect the new capabilities published by the application

Intent resolution (implicit intents)





component specified in matched intent filter

Intent objects:

- action
- possibly category(ies)
- possibly data (scheme, host, port,path)

Match with known intent filters:

- action(s) (at least one) - must include the action in intent
- possibly category(ies) - all categories in intent must be included
- data (scheme, host, port,path) - if specified, intent must have data that matches with filter. Matching order is scheme, host, port, path

Activity (or service):

- call `getIntent()` to get the intent that was used for activating the component (to read extras bundle, for example)

Intent filter example

```
<activity android:name=".MainView" android:label="@string/app_name">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>

<activity android:name=".CityInfo" android:label="Second">
  <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:scheme="http" />
  </intent-filter>
</activity>
```

MAIN - this is the initial activity

LAUNCHER - this activity is shown in the application launcher

Reading list

- http://developer.android.com/training/basics/data-storage/shared-preferences.html
- http://developer.android.com/training/basics/data-storage/files.html
- http://developer.android.com/guide/components/intents-filters.html
- http://developer.android.com/reference/android/content/Intent.html