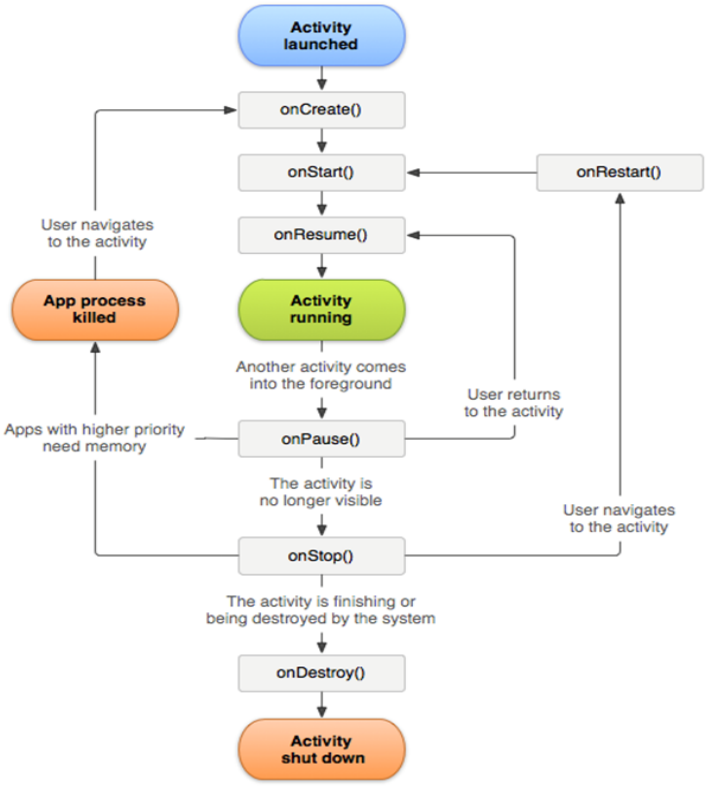
********** Homework Week 1 Day 3 **********

Answer the following questions:

1. **What are the major components in android?**
   The major components in android are android are activities and fragment Managers, views, Notification Manager, services, intents, content providers, Android Manifest.xml
   - **Activity and Fragment Manager** are used to define the user interface of the app. They control the life cycle of the activities and fragments respectively, including management of the Activity stack.
   - **Views** is used to construct the user interfaces controls within your Activity and Fragments
   - **Notification Manager** Provides
   - **Services** (Resource Manager) enables non-code resources, such as strings and graphics to be externalized.
   - **Intents** provides a mechanism for transferring data between applications and their components
   - **Content providers** lets your applications share date
   - **Android Manifest.xml** contains information about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

2. **What are all the activity lifecycles and explain what each detail?**



Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.
onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.

3. **Explain each step of the sprint in agile.**
   a) **Backlog:** Make a list of all the things you and your team need to do.
   Think about your projects and goals. What specific tasks need to be completed to reach them? Don't hold back. Get down to the dirty details, and try to break down tasks into their smallest parts.
   b) **Sprint or iteration:** Create a new task board to list all the things to be accomplish over the next week or two. Agile project management values working by time: what exactly can you and your team achieve in a specified period of time?
   c) **iteration planning" or "sprint planning**: Move tasks from your backlog to your task board. Be ambitious, but realistic, about what your team can achieve over the next two weeks. Go through your various backlogs, if you have more than one, to ensure you have all your bases covered.
   d) **Assign each task in your sprint to someone on your team:** Ownership motivates—that's a guarantee. When a task has a person's face next to it, it motivates the owner to take responsibility and see it through to the end. At the end of the iteration, it gives each individual recognition for their hard work.
   e) **Prioritize the tasks in your upcoming sprint:** Agile project management favors categorizing tasks according to four priorities: critical, high, medium, and low. Since plans can change and things might take longer than you estimate, there's a good chance your team won't be able to complete every single task in your upcoming sprint.
   f) **Story point:** Estimate how long each task will take. When going through each task, try to think about the amount of work that needs to be done, the complexity of the work, and any risk or uncertainty you might encounter when working on the task.
   g) **Start your sprint!** As you go, communicate with your team and update them on where things stand. Scrum respects that things change and unexpected circumstances

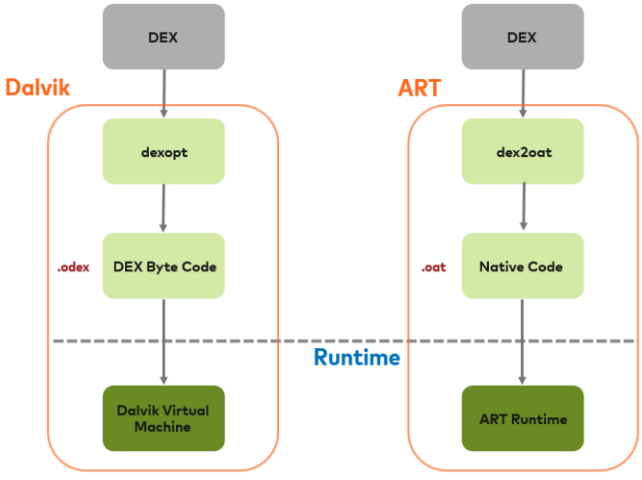
can stop a task from reaching completion. If a task gets stuck or delayed, just update your status column to let everyone else on your team know where thing stand.

h) **Dailies or Standups:** Hold short daily meetings with your whole team. Face-to-face communication is the best way for everyone to stay in sync. Schedule a short 10 to 15-minute scrum daily standup meeting at the beginning of every workday so that everyone on your team can provide a quick overview of what they worked on the day before and what they plan to work on that day.

i) **Retrospection:** You finished your sprint. Great work! Now review and analyze what worked and what didn't. Once your sprint is over, sit as a team to celebrate what you all achieved, troubleshoot what went wrong, and plan how to tackle things in the next iteration. Pull request.

j) **Rinse and repeat:** Move your completed sprint to the bottom of your board. Start a new iteration at the top of the board. Storing all your iterations in one board allow you to keep a clear record of everything you've completed. Didn't finish everything this sprint? Move it to the next one. And should a task from a previous sprint come back to life, you can just drag it back into your current sprint.

4. **What is dalvik and ART?**
   - **Dalvik** is a part of the software stack that makes up the Android platform. According to Google's Android documentation, the Dalvik VM is an interpreter-only virtual machine that executes files in the Dalvik Executable (.dex) format, a format that is optimized for efficient storage and memory-mappable execution.
   - **ART** (Android Run Time) is the next version of Dalvik



Dalvik vs ART

5. **How is the android platform architecture designed?**
   a) The Linux Kernel The foundation of the Android platform is the Linux kernel. For example, the Android Runtime (ART) relies on the Linux kernel for underlying functionalities such as threading and low-level memory management.

b) **Hardware Abstraction Layer (HAL)** The hardware abstraction layer (HAL) provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or Bluetooth module.

c) **Android Runtime** For devices running Android version 5.0 (API level 21) or higher, each app runs in its own process and with its own instance of the Android Runtime (ART). ART is written to run multiple virtual machines on low-memory devices by executing DEX files, a bytecode format designed specially for Android that's optimized for minimal memory footprint.

d) **Native C/C++ Libraries** Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in C and C++. The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps.

e) **Java API Framework** The entire feature-set of the Android OS is available to you through APIs written in the Java language.

f) **System Apps** Android comes with a set of core apps for email, SMS messaging, calendars, internet browsing, contacts, and more. Apps included with the platform have no special status among the apps the user chooses to install.