9. Android is an open-source operating system for mobile devices. Nowadays, it has more than 1.4 billion monthly active users (statistic from September 2015) and the largest share on the mobile device market compared to all operating systems, more than 87% in the 2nd quarter of 2016. Google made it easy for people to develop their own applications and make them available on the official market Google Play.

10-13. Now lets see how Android architecture looks like.
Android runs on top of a Linux kernel. Due to the Android Mainlining Project and the more recent Android Upstreaming project, it runs on top of a vanilla kernel with little modifications, which are called Androidisms. The advantages of using a Linux kernel are that it can use some of the security mechanisms already implemented in Linux and allows manufacturers to easily develop device drivers.

On top of the Linux kernel, we have the Hardware Abstraction Layer (HAL), which provides standard interfaces that expose hardware capabilities to the Java API framework. It includes multiple library modules, each one implementing an interface for a hardware component, for example camera, or sensors. When the framework wants to access a hardware component, the Android system will load the appropriate library module.

The native userspace, includes the init process, a couple of native daemons and hundreds of native libraries. The init process and native daemons have a different functionality than the ones on a standard Linux. For example, we will see that the init process is not the parent of all processes in the system, but Zygote is.

The native libraries are implemented in C/C++ and their functionality is exposed to applications through Java framework APIs. For example, applications can access the OpenGL ES library by using the Java OpenGL API. The native libraries can also be accessed through the Android NDK, from applications that include native code.

A large part of Android is implemented in Java and until Android version 5.0 the default runtime was Dalvik. Recently a new, more performant runtime, called ART has been integrated in Android (first made available as an option in version 4.4, default runtime from version 5.0). ART supports Ahead-Of-Time compilation.

Java runtime libraries are defined in java.* and javax.* packages, and are derived from Apache Harmony project (not Oracle/SUN, in order to avoid copyright and distribution issues.), but several components have been replaced, others extended and improved. Native code can be called from Java through the Java Native Interface (JNI).

Many fundamental features of Android are implemented in the system services: display, touch screen, telephony, network connectivity, and so on. A part of these services are implemented in native code and the rest in Java. Each service provides an interface which can be called from other components.

Android Framework libraries (also called "the framework") include base components for implementing Android applications: base classes for activities, services, content providers, GUI widgets, file access, database access, and so on. It also includes classes for the interaction with the hardware and the higher level services. Practically, both normal and system applications have access to the same framework APIs.

14. The Linux kernel used in Android is modified to work on mobile devices. Android Mainlining Project and the more recent Android Upstreaming project, are dealing with integrating these modifications in the mainline. Many Androidisms have been already integrated in mainline, such as WakeLocks in kernel version 3.5 (a similar mechanism), Low-Memory Killer in 3.10, Binder in 3.19, Alarm and Logger in 3.20.

Another difference is that the kernel includes only the suspend to memory mechanism, and not the suspend to hard disk, which is used on PCs.

15. Until version 5.0, Android Runtime included the Dalvik virtual machine, which is used to run the bytecode of both applications and system services, which are implemented in Java.
Dalvik runs .dex files (this comes from Dalvik Executable) instead of .class. These files are with 50% smaller than the jar containing the associated class files.
From Android 2.2, Dalvik provides Just-In-Time compilation, which means that short segments of bytecode that are executed most often are compiled in native machine code. This brings performance improvements. The rest of the bytecode is interpreted by Dalvik.

16. On this slide we have the process through which an Android application package (.apk) is created. The java files of the application are compiled into .class files, then the dx tool is used to aggregate these .class files into a single .dex file. Each application includes a .dex file in the .apk archive.

17. ART is a more advanced runtime architecture from Android 5.0. It includes Ahead-Of-Time compilation, which means that, at installation time, it will translate the entire DEX bytecode into native machine code and will store it for the future execution of the application. This is done only once, at install time, so it is more efficient and reduces the power consumption associated with the compilation and interpretation. AOT replaces JIT compilation and Dalvik interpretation. The disadvantage is that it occupies more storage space with the compiled code and app installation takes more time.
In addition, ART provides improved memory allocation and garbage collection mechanisms, new debug features and more accurate high-level application profiling. ART relies on the Linux kernel for functionalities such as low memory management and threading.

18. On top of the Linux kernel, we have hundreds of native libraries. The most important one is bioniC, which is the C library of Android, a replacement for glibc. It has BSD license and it is much smaller and faster than glibc.
SQLite is a library for SQL database management.
OpenGL ES is a version of OpenGL for embedded devices. OpenGL is a standard software interface for the hardware that performs 3D graphical processing.
WebKit is a library for displaying web pages, used in many operating systems for mobile devices: Android, Apple iOS, Blackberry, Tizen.
SSL includes the implementation of security protocols used for securing the communication over the Internet.

19. Application Framework includes services and managers used by the applications, for example, Telephony Manager for phone calls, Location Manager for obtaining the location, Activity Manager for handling activity lifecycle (which are application components), Package Manager for handling the application packages, Notification Manager for generating and handling notifications. The framework also includes the implementation of the system Content providers (the default ones): contacts, calendar, dictionary, media store, settings, and so on.

20. When an application wants to access a system service, it will call a method from the public API, which in turn will call an RPC stub which communicates through the Binder with the system service from the framework. In addition, we can observe that applications can call the native libraries through JNI.

22. An Android application has 4 types of components - activities (for the interaction with the user), services, broadcast receivers and content providers (that run in background without the interacting directly with the user).

23. Activities represent the graphical user interface of the application. An activity is similar with a window from the window-based graphical interfaces.
The user can interact with a single activity at a certain moment. From an activity can be started another activity and so on, this way a stack of activities is created. Similar with using a browser, the user can press back in order to come back to the previous activity. However, there is no forward button. When the user presses back, the current activity is removed from the stack and destroyed. But you can start it again from the graphical interface.
An activity can be started using an intent: either from the launcher or from another application component.

24. A service is an application component that executes operations in background and it does not have a graphical interface. It usually runs in the same process with the application, but it can be configured to run in a different process.
A service can perform tasks for the current application or can provide services to other applications. The communication with a service is always done through the Binder.

25. A broadcast receiver is used for receiving broadcast messages that include announcements or notifications. For example, when the battery is depleted, the screen is turned off, the phone is rebooting, but also custom application notifications.
Some notifications are global (throughout the system) and others are local (just in the current application).
The broadcast messages are delivered through intents.
In our application, we can choose which broadcasts to receive and treat using intent filters.
A broadcast receiver is active only when it receives a broadcast message.

26. Content Providers are used for storing, managing and sharing the data of an application.
If an application wants to share a set of data with other applications, it will have to use a content provider.
Content providers use relational databases (SQLite) or files for storing the data.
URIs will be used for identifying the provider and the table. The Content Resolver (which is a client object) will use this URI to send queries to the provider (which is the server object).
The content provider is also active only when it receives a request.

27. Intents are similar to signals in Linux. They are used for sending a message in order to determine the execution of an action.
They are used with the following main purposes: for starting activities, for starting and binding services and for sending broadcast messages to the receivers.
Intents are managed and delivered by the system.
An intent has two main components: the action that has to be executed and the data which is used by that action (for example an URI). Here we have an example: the action is dial and the data is an URI to a certain entry in the contacts.
There are two types of intents: explicit and implicit. Explicit intents are sent directly to a receiver and for the implicit intents, the system will find the most appropriate application that can perform that action.

28. The Binder is an RPC mechanism (remote procedure call) through which remote objects can be invoked. It is used for the communication between two components from the same process or from different processes.
The data must be parceled and sent through the Binder from one entity to another.

The call is synchronous, so the first component will block until it receives a response from the second component. We will discuss the Android SDK in more detail in the next lectures.