Mobile Computing Applications

Lecture 1 – Introduction

Dr. Sherin Moussa

Associate Professor at Faculty of Computer & Information Sciences,
Ain Shams university

sherinmoussa@cis.asu.edu.eg

Credit

- The slides heavily use the Slide decks Provided by the Android Developer Fundamentals Course By Google which are under a *Creative Commons Attribution 4.0 International License*.

- The contribution to these slides takes the following forms:
  - Re-ordering and re-mixing topics to match the course objectives.
  - Adding different slides, code samples and content.
  - Deleting some slides to minimize some topics.
  - Video tapping the course content on YouTube.

Lab Content

Android Mobile Development

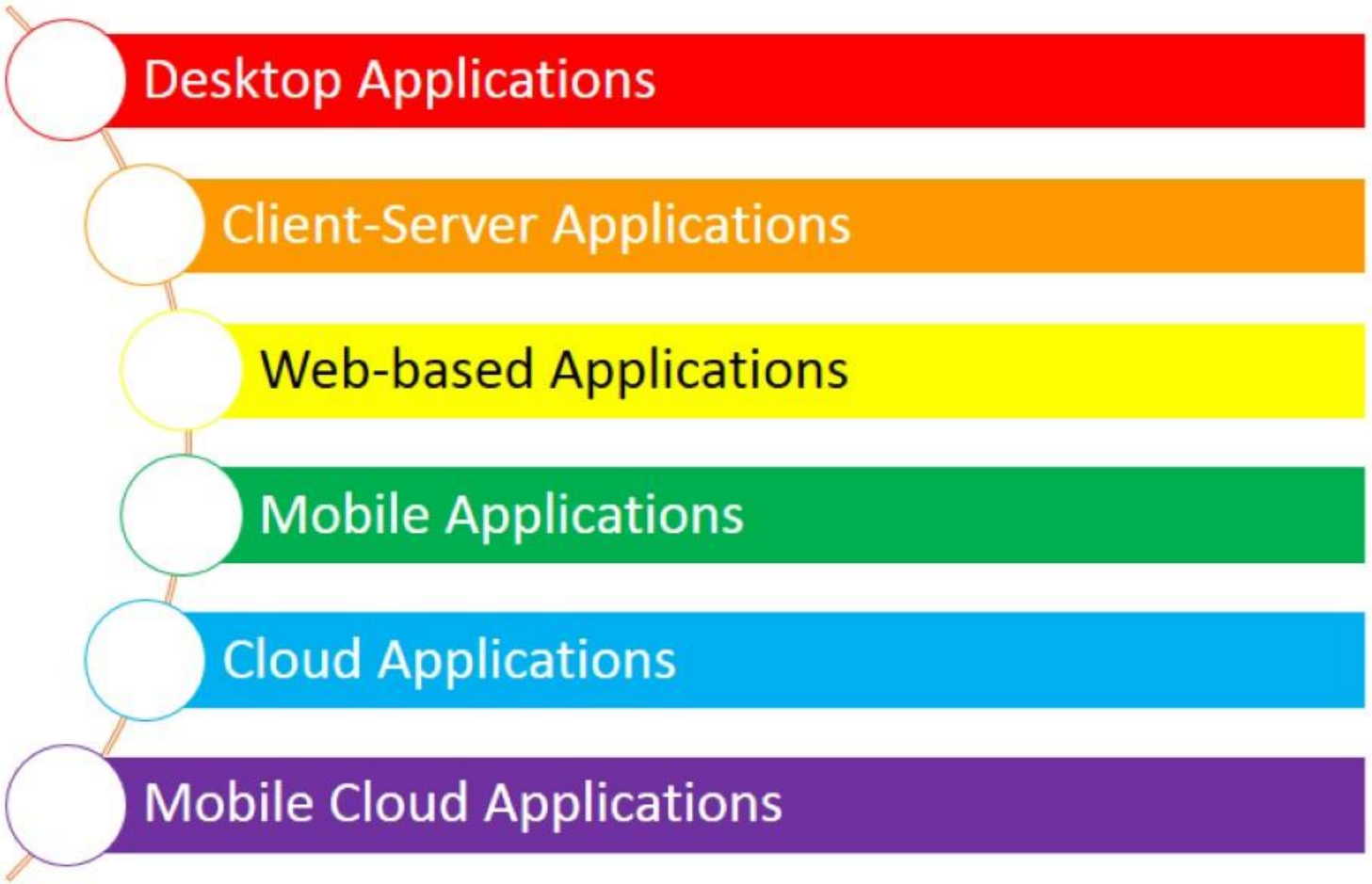
Grading Scheme

- 40 grades for the final exam
- 30 grades for the two midterms
- 18 grades for the lab tasks
- 12 grades for the lab project over 3 main milestones

Topics

1. Introduction and Android Studio.
2. XML and User Interface
3. UI Events and Events Listeners
4. Make App interactive : User input and Interaction
5. Multiscreen Apps: Activities and Intents
6. Activity Lifecycle, Fragments
7. Multimedia: Playing Audio and Videos
8. Networking and APIs
9. Data Storage: Shared Preferences and Files
10. SQLite, Content Providers and Rooming
11. Maps and Locations
12. Sensors

Evolution of Computing



What is Mobile Computing?

- Is the computing that allows continuous access to remote resources, even to small computing devices.
- A form of human–computer interaction by which a computer is expected to be transported during normal usage.
- Thus, Mobile Computing is the ability to use:
- **Computing devices**: publish and/or subscribe to information and connect to the internet.
- That are **Mobile**: changing location; without a pre-defined location.
- Through a **Wireless connection** to a network: that provides wireless transmission to access data and information from wherever location people may be.

Introduction to Android And Building First App

Contents

- Android is an ecosystem
- Android platform architecture
- Android Versions
- Challenges of Android app development
- App fundamentals

What is Android?

- Mobile operating system based on Linux kernel
- User Interface for touch screens
- Used on over 80% of all smartphones
- Powers devices such as watches, TVs, and cars
- Over 2 Million Android apps in Google Play store
- Highly customizable for devices / by vendors
- Open source

Android user interaction

- Touch gestures: swiping, tapping, pinching
- Virtual keyboard for characters, numbers, and emoji
- Support for Bluetooth, USB controllers and peripherals

Android and sensors

Sensors can discover user action and respond

- Device contents rotate as needed
- Walking adjusts position on map
- Tilting steers a virtual car or controls a physical toy
- Moving too fast disables game interactions

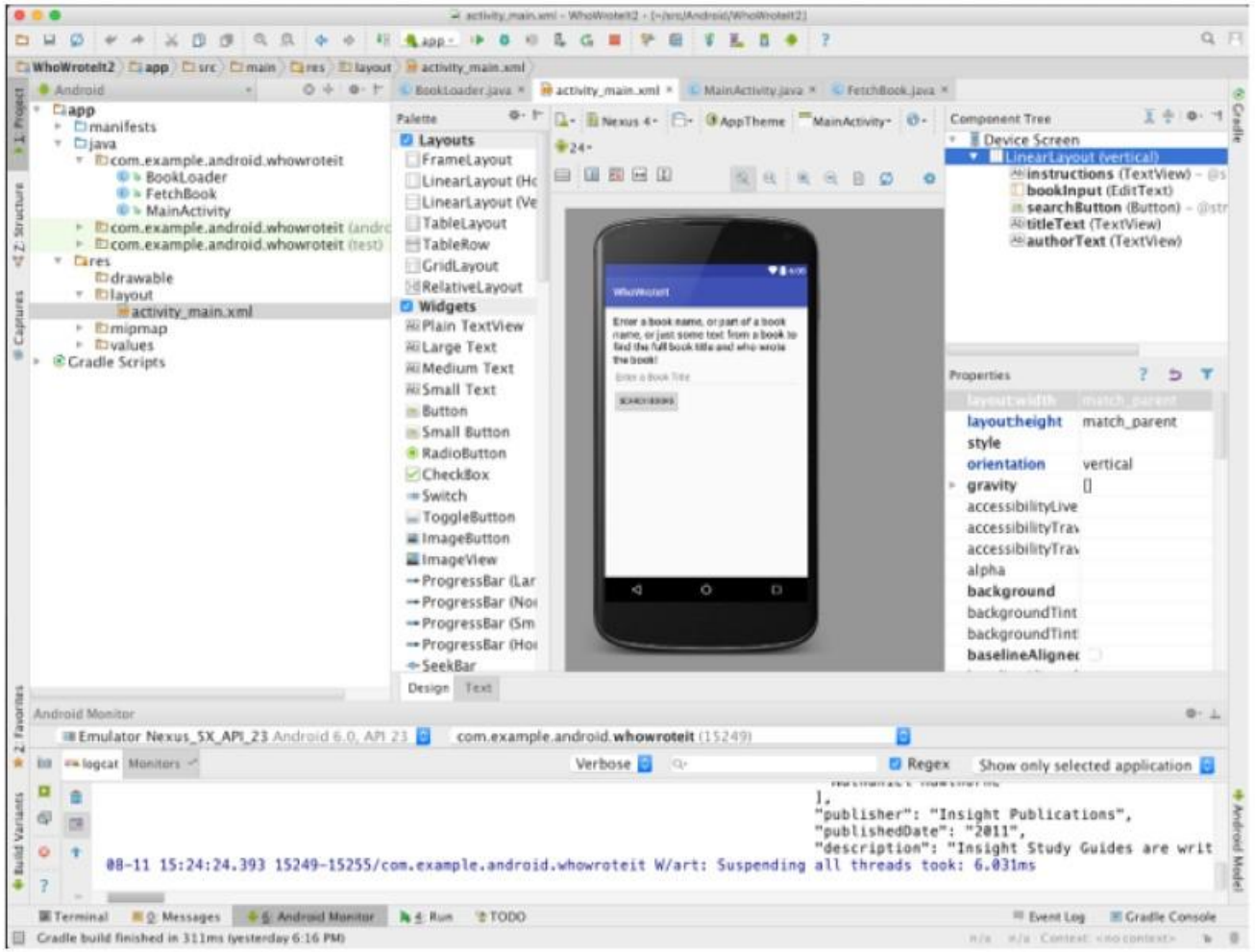
Android home screen

- Launcher icons for apps
- Self-updating widgets for live content
- Can be multiple pages
- Folders to organize apps
- "OK Google"

Android Software Developer Kit (SDK)

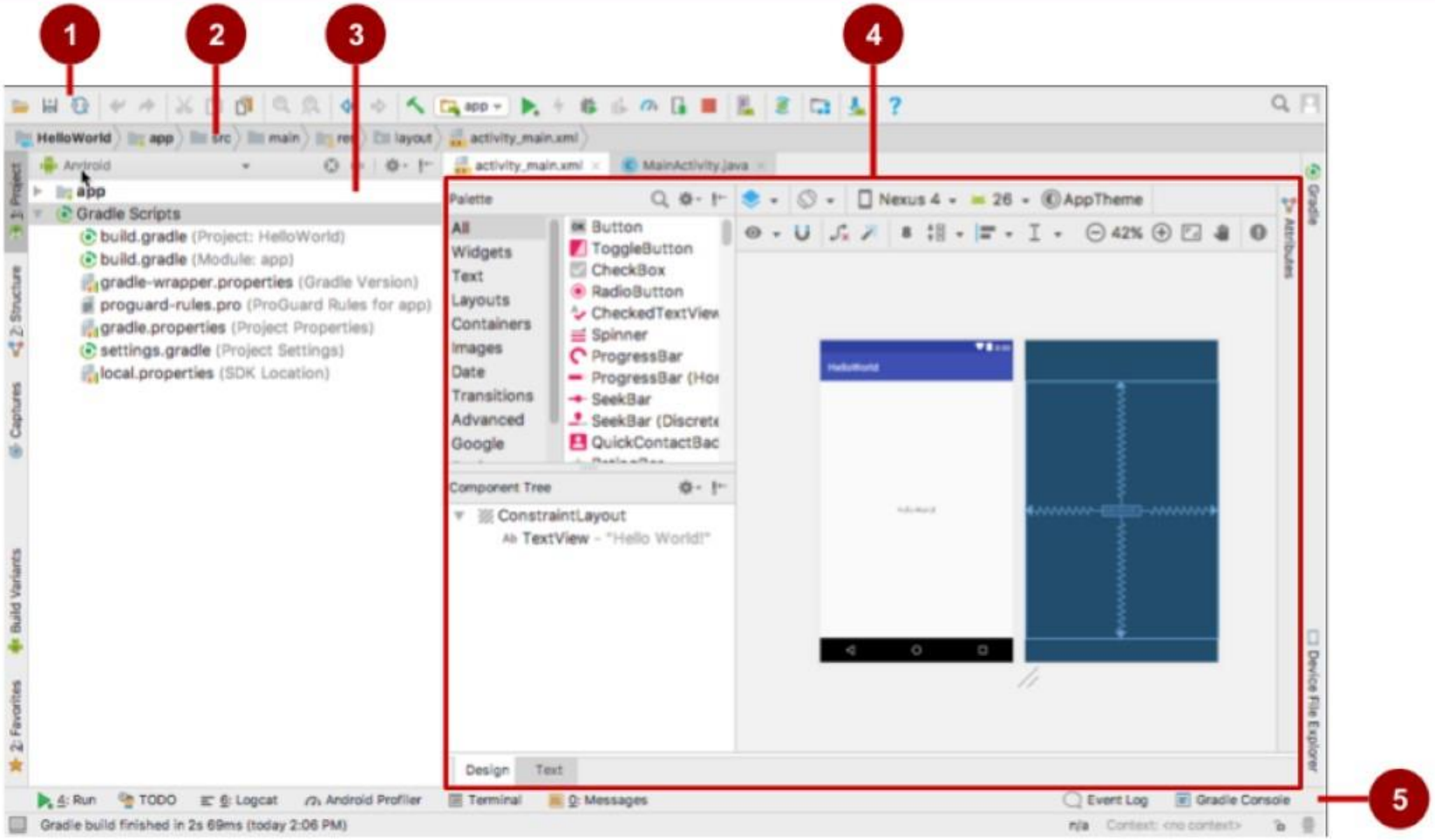
- Development tools (debugger, monitors, editors)
- Libraries (maps, wearables)
- Virtual devices (emulators)
- Documentation (developers.android.com)
- Sample code

Android Studio



- Official Android IDE
- Develop, run, debug, test, and package apps
- Monitors and performance tools
- Virtual devices
- Project views
- Visual layout editor

Android Studio interface



1. Toolbar
2. Navigation bar
3. Project pane
4. Editor
5. Tabs for other panes

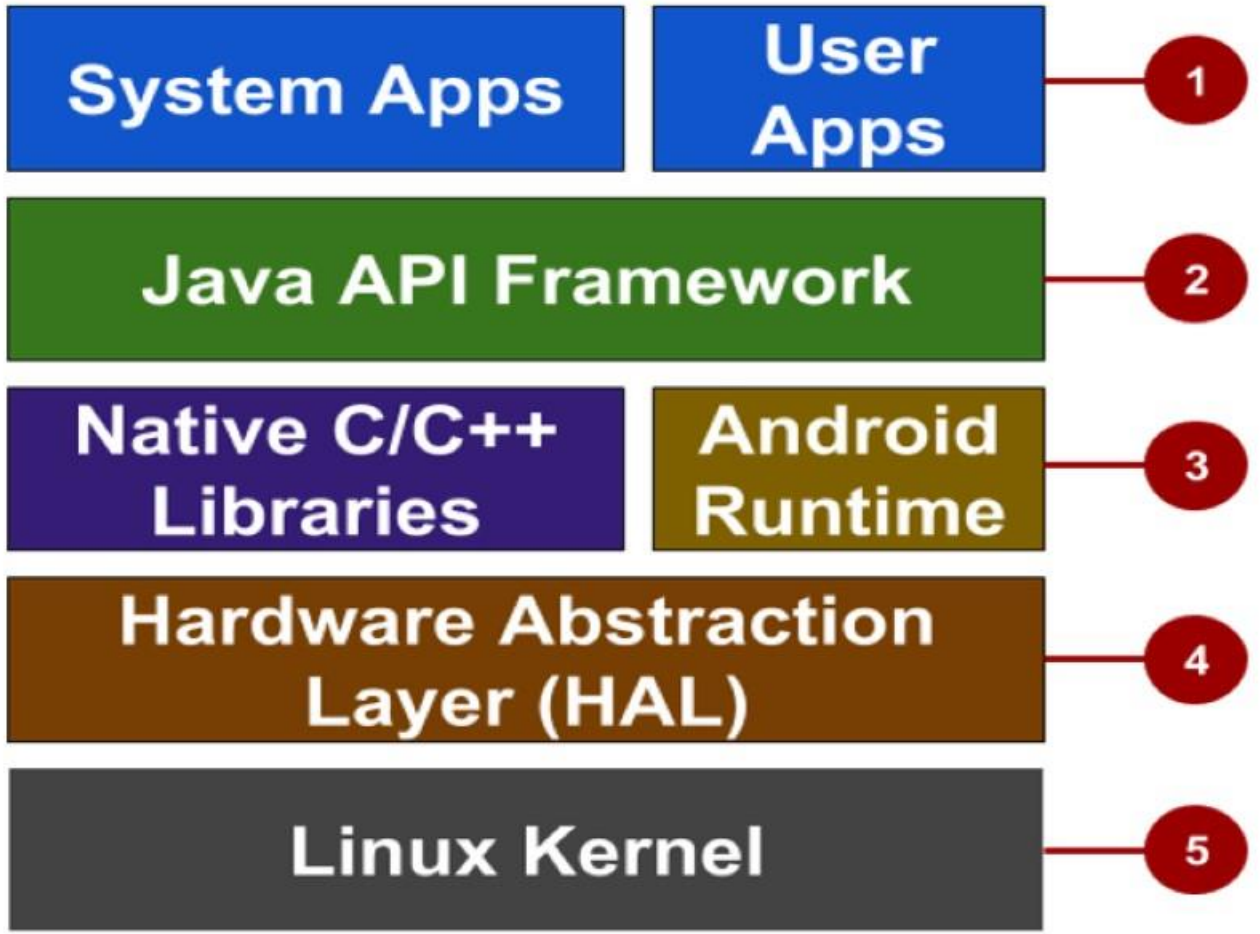
Google Play store

Publish apps through Google Play store:

- Official app store for Android
- Digital distribution service operated by Google

Android stack

1. System and user apps
2. Android OS API in Java framework
3. Expose native APIs; run apps
4. Expose device hardware capabilities
5. Linux Kernel



System and user apps

- System apps have no special status
- System apps provide key capabilities to app developers

Example:

Your app can use a system app to deliver a SMS message.

Java API Framework

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

- View class hierarchy to create UI screens
- Notification manager
- Activity manager for life cycles and navigation

Android runtime

Each app runs in its own process with its own instance of the Android Runtime.

C/C++ libraries

- Core C/C++ Libraries give access to core native Android system components and services.

Hardware Abstraction Layer (HAL)

- Standard interfaces that expose device hardware capabilities as libraries

  Examples: Camera, bluetooth module

Linux Kernel

- Threading and low-level memory management
- Security features
- Drivers

Older Android versions

Codename	Version	Released	API Level
Honeycomb	3.0 - 3.2.6	Feb 2011	11 - 13
Ice Cream Sandwich	4.0 - 4.0.4	Oct 2011	14 - 15
Jelly Bean	4.1 - 4.3.1	July 2012	16 - 18
KitKat	4.4 - 4.4.4	Oct 2013	19 - 20
Lollipop	5.0 - 5.1.1	Nov 2014	21 - 22

Newer Android versions

Codename	Version	Released	API Level
Marshmallow	6.0 - 6.0.1	Oct 2015	23
Nougat	7.0 - 7.1	Sept 2016	24 - 25
Oreo	8.0 - 8.1	Sept 2017	26 - 27
Pie	9.0	Aug 2018	28

What is an Android app?

- One or more interactive screens
- Written using Java Programming Language and XML
- Uses the Android Software Development Kit (SDK)
- Uses Android libraries and Android Application Framework
- Executed by Android Runtime Virtual machine (ART)

Advantages of Android

- The open source nature of Android makes it easier for device manufacturers and developers to use it.
- Java programming language is popular and is already widely used.
- Application can be developed on any operating system using the Android Studio or Eclipse ADT IDE.
- Wide array of devices and manufacturers.
- Multitasking – Android phones can run many applications, it means you can browse, Facebook while listening to a song.
- Android's review process for apps is fairly simple and it takes less time than iOS for an app to get approved for publishing on the play store.

Challenges of Android development

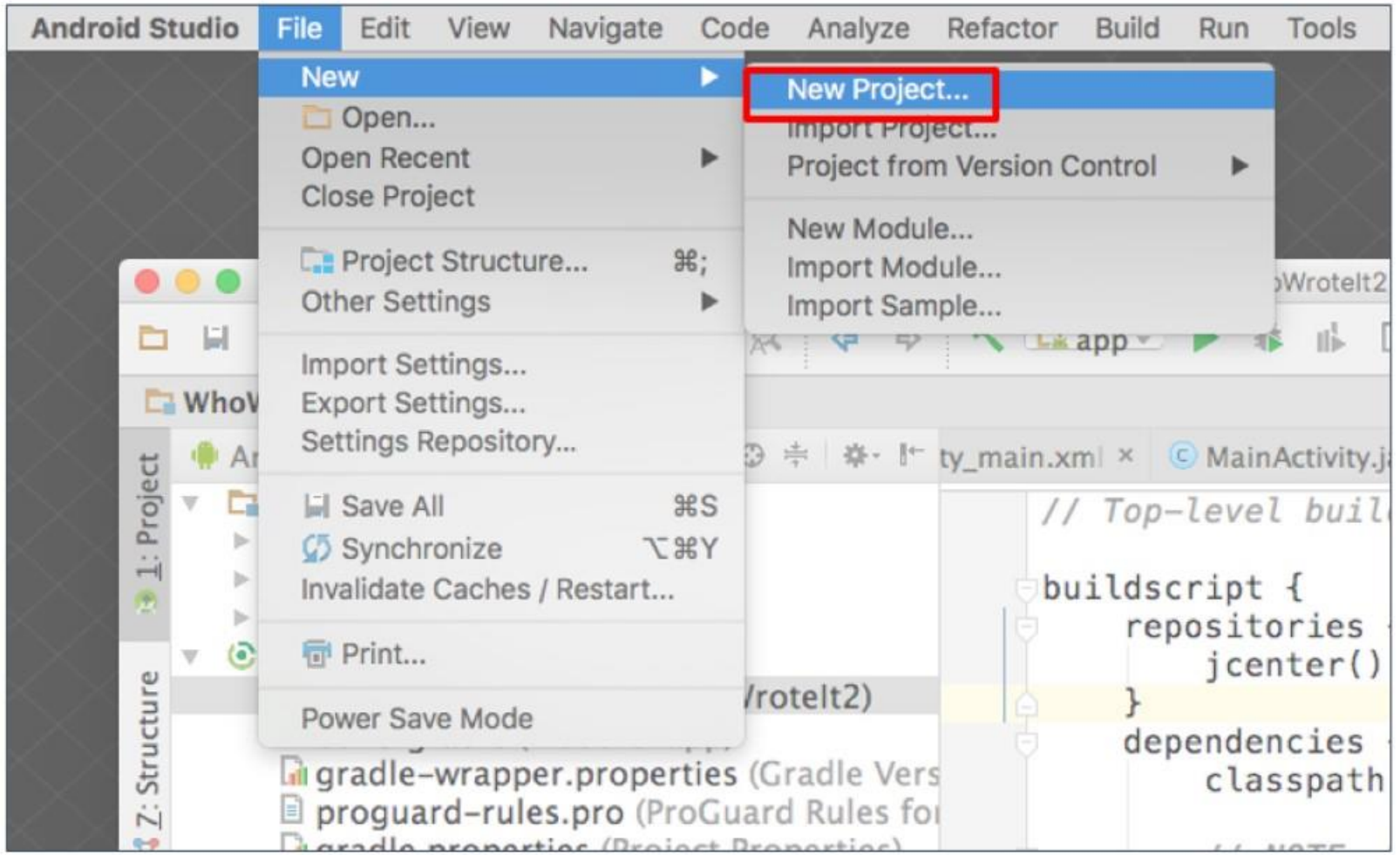
- Multiple screen sizes and resolutions
- Performance: make your apps responsive and smooth
- Security: keep source code and user data safe
- Compatibility: run well on older platform versions
- Marketing: understand the market and your users

  (Hint: It doesn't have to be expensive, but it can be.)

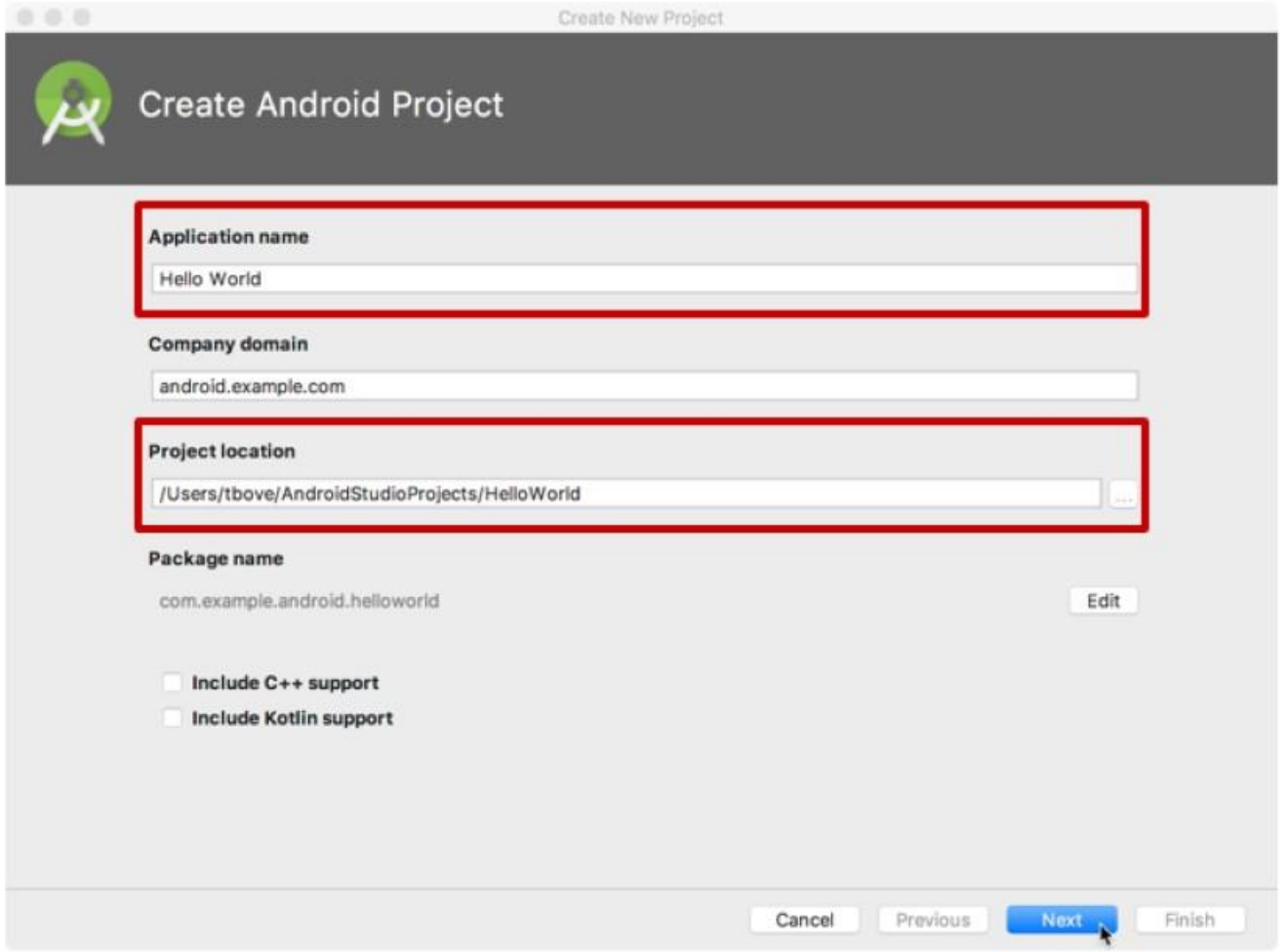
App building blocks

- **Resources**: layouts, images, strings, colors as XML and media files
- **Components**: activities, services, and helper classes as Java code
- **Manifest**: information about app for the runtime
- **Build configuration**: APK versions in Gradle config files

Create a project inside Android Studio

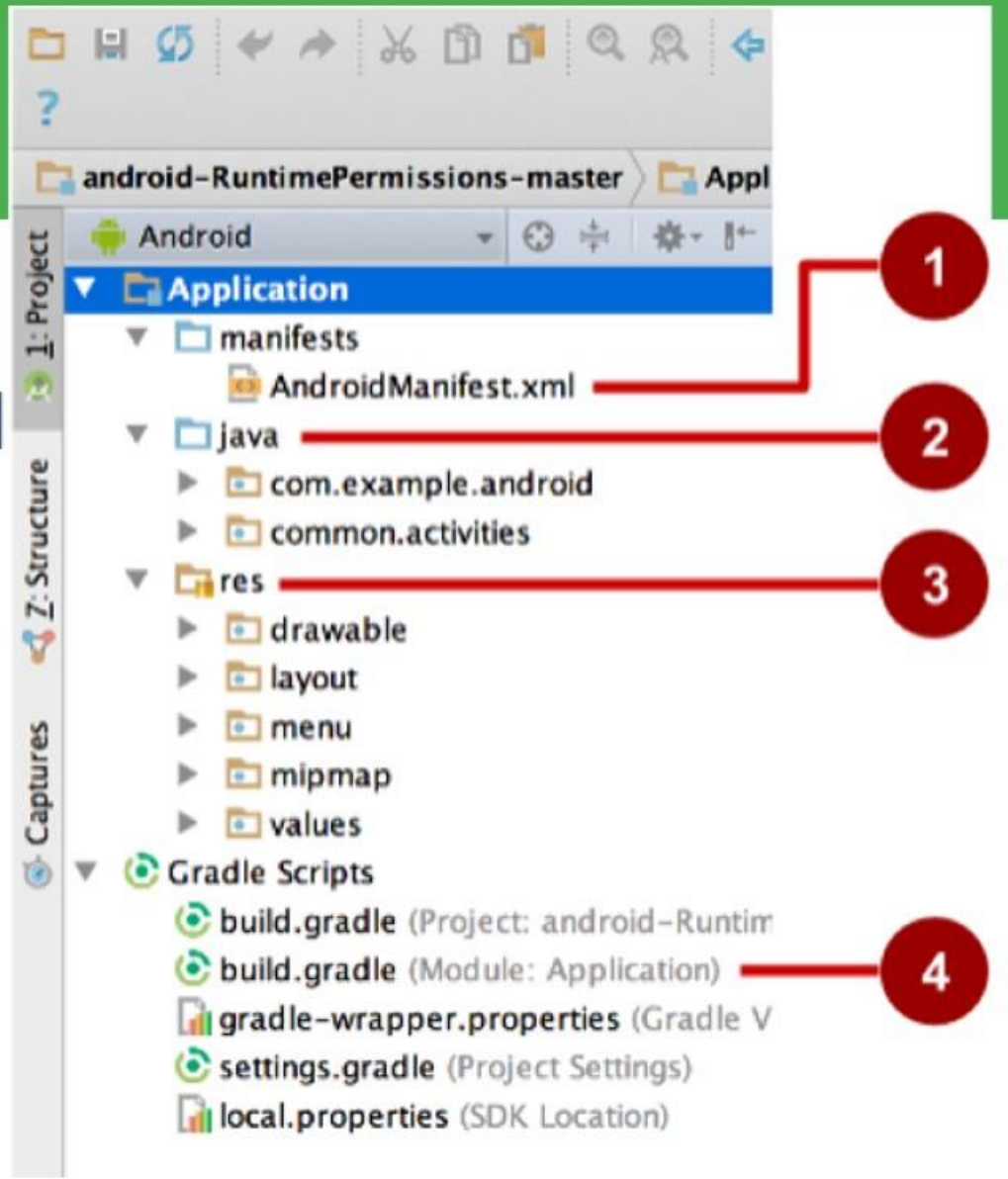


Name your app

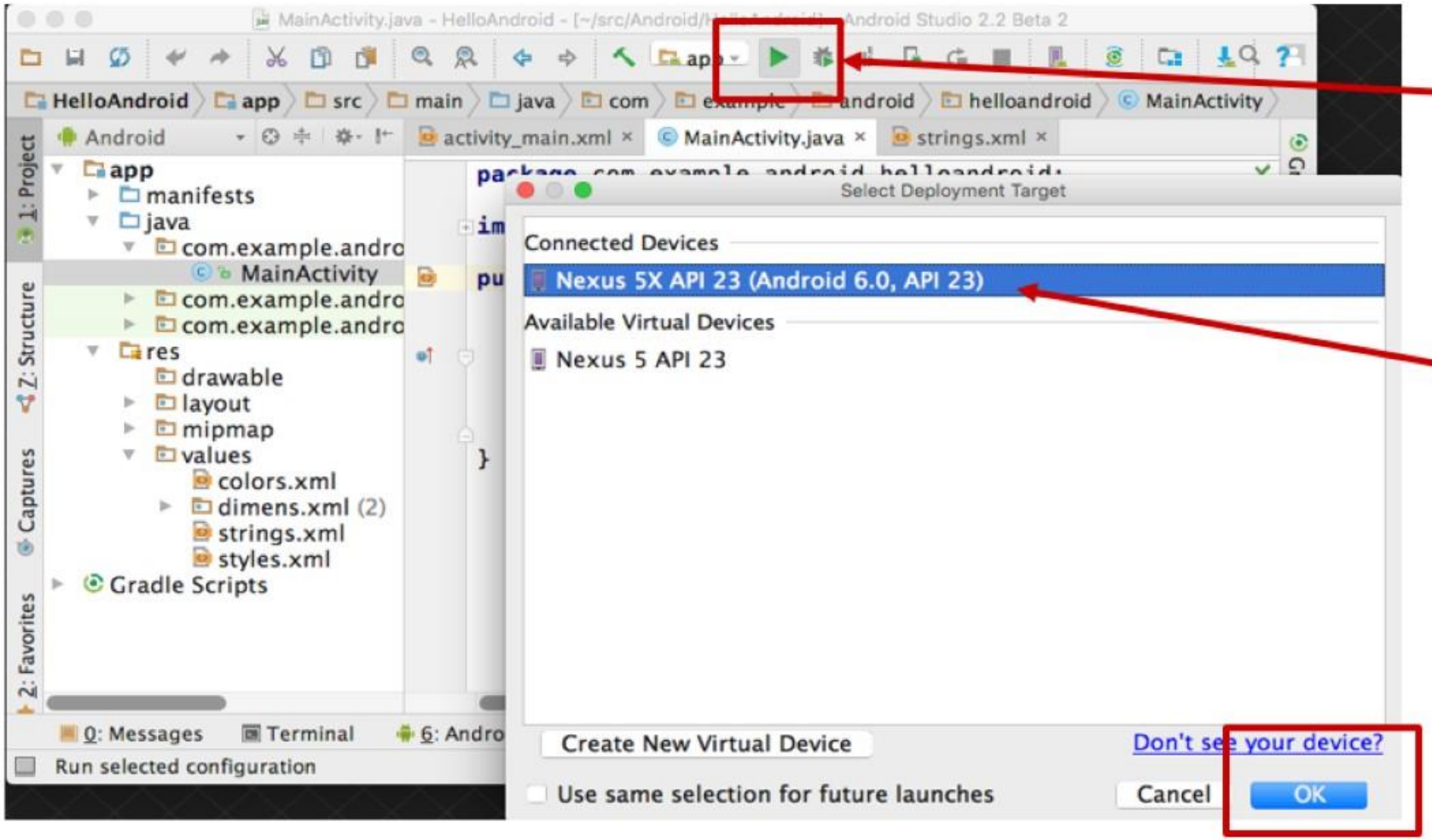


Project folders

1. **manifests**—Android Manifest file - description of app read by the Android runtime
2. **java**—Java source code packages
3. **res**—Resources (XML) - layout, strings, images, dimensions, colors...
4. **build.gradle**—Gradle build files



Run your app

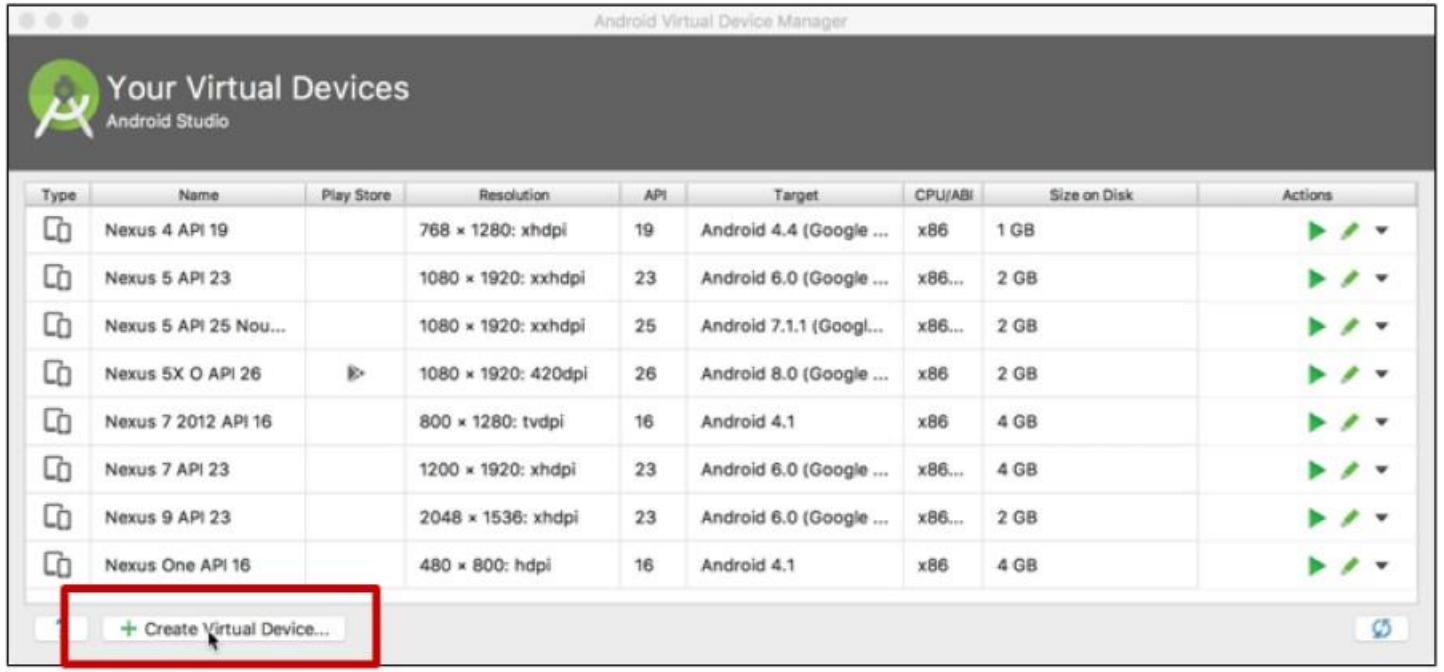


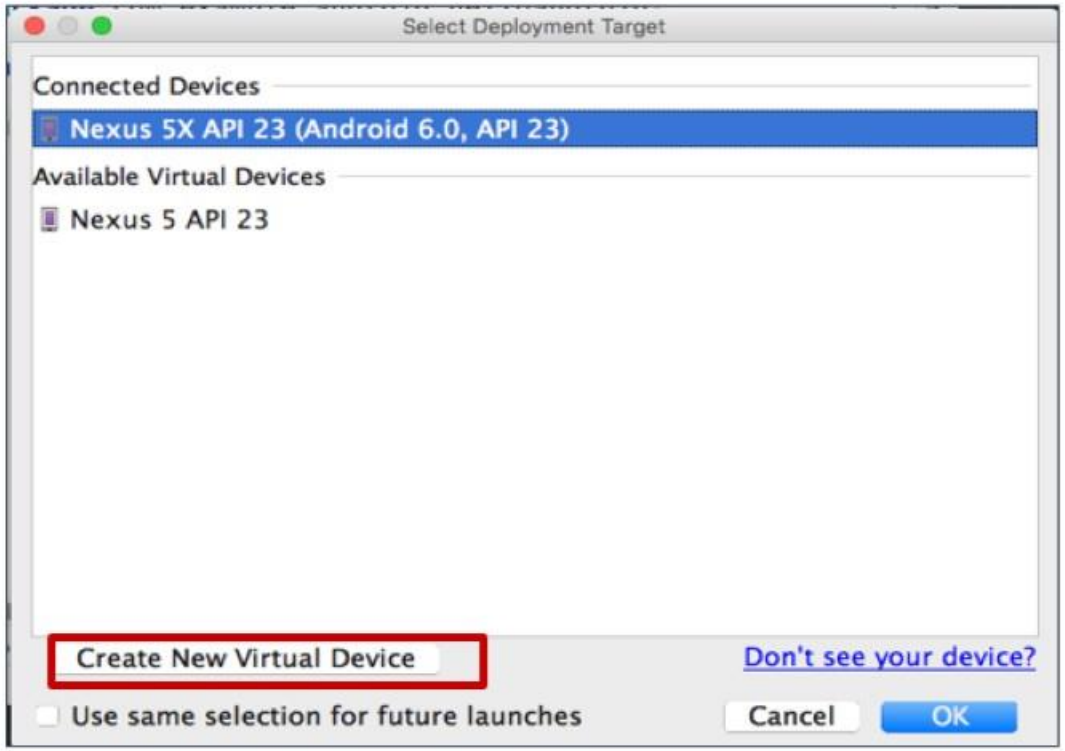
1. Run
2. Select virtual or physical device
3. OK

Create a virtual device

Use emulators to test app on different versions of Android and form factors.

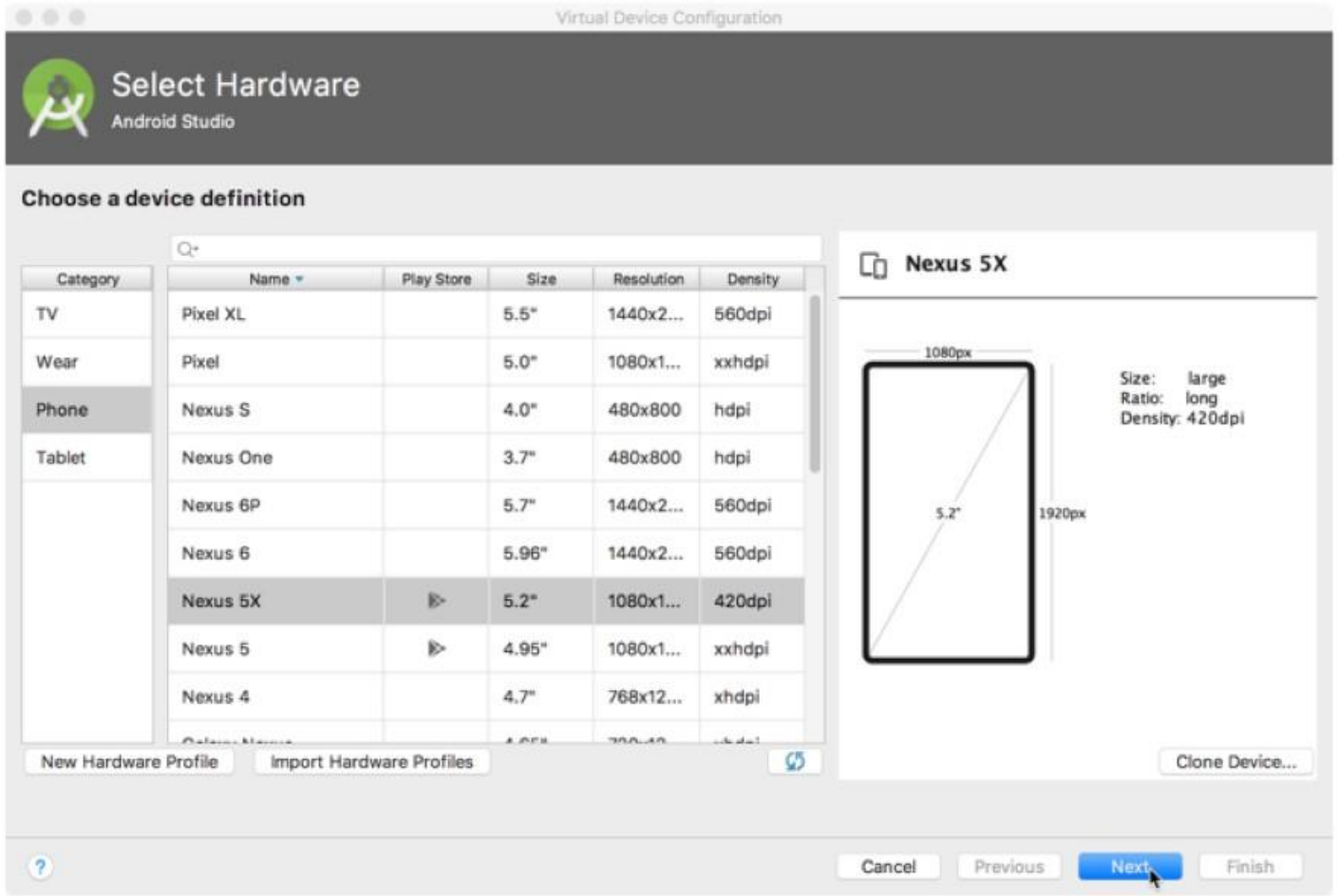
Tools > Android > AVD Manager or:



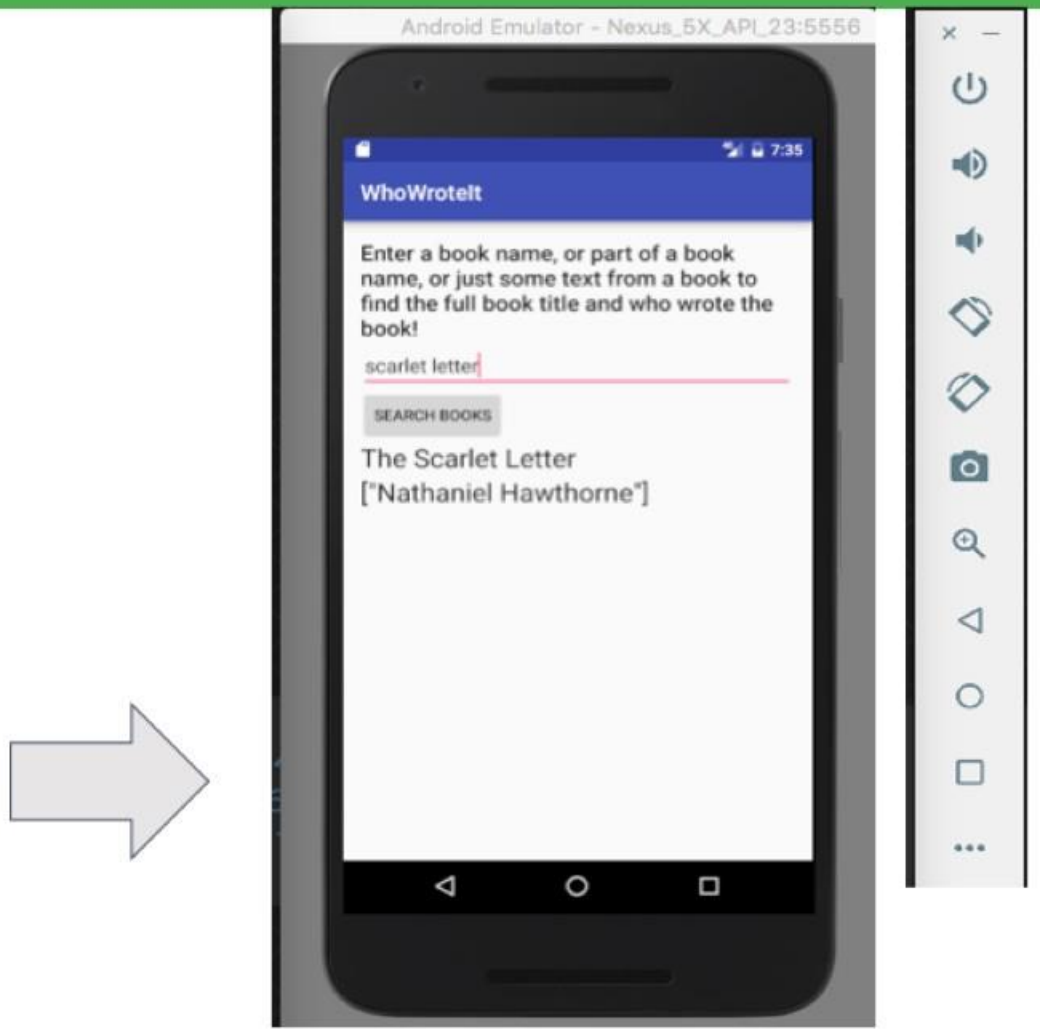


Configure virtual device

1. Choose hardware
2. Select Android version
3. Finalize

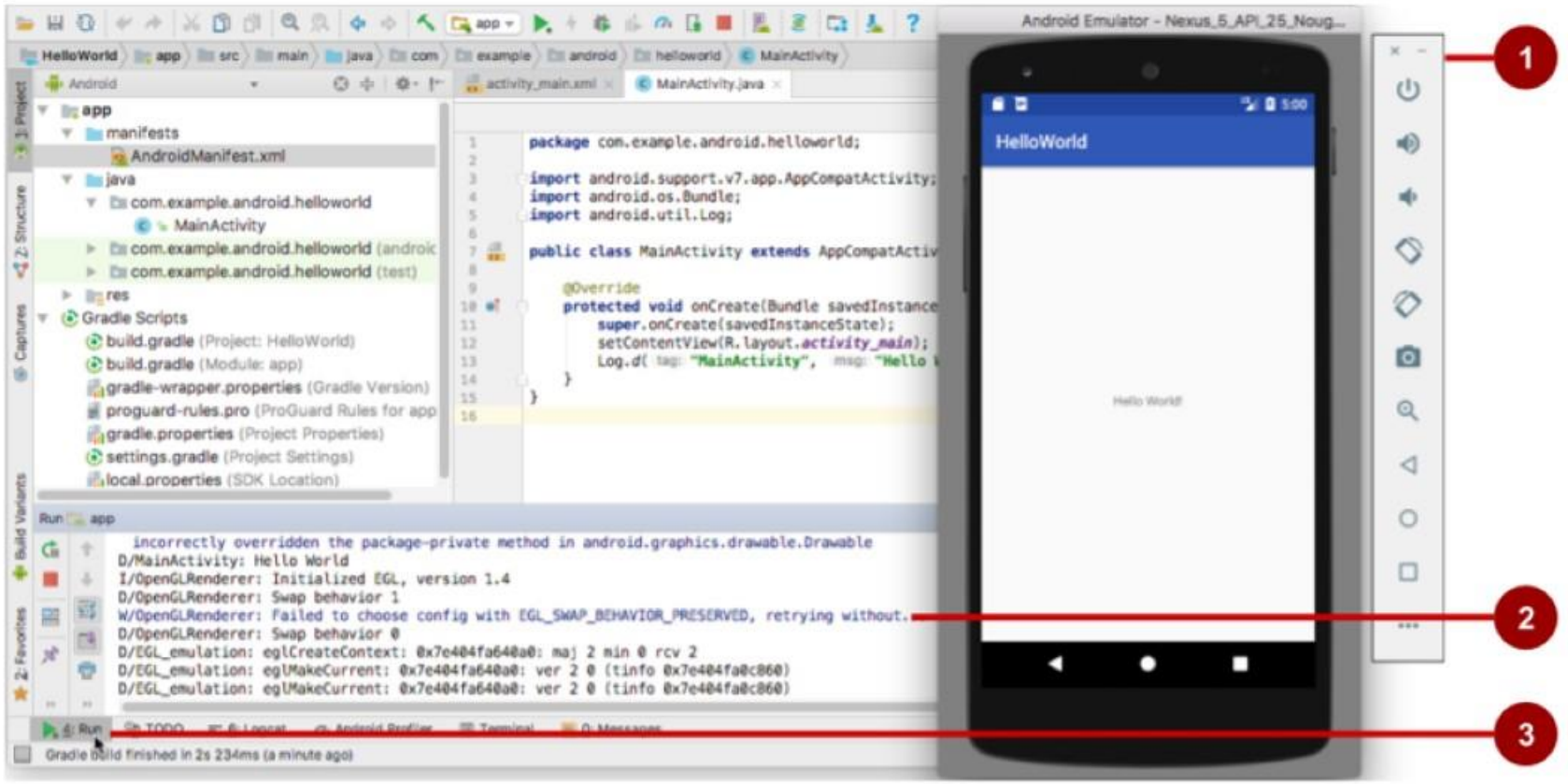


Run on a virtual device



Get feedback as your app runs

1. Emulator running the app
2. Run pane
3. **Run** tab to open or close the Run pane

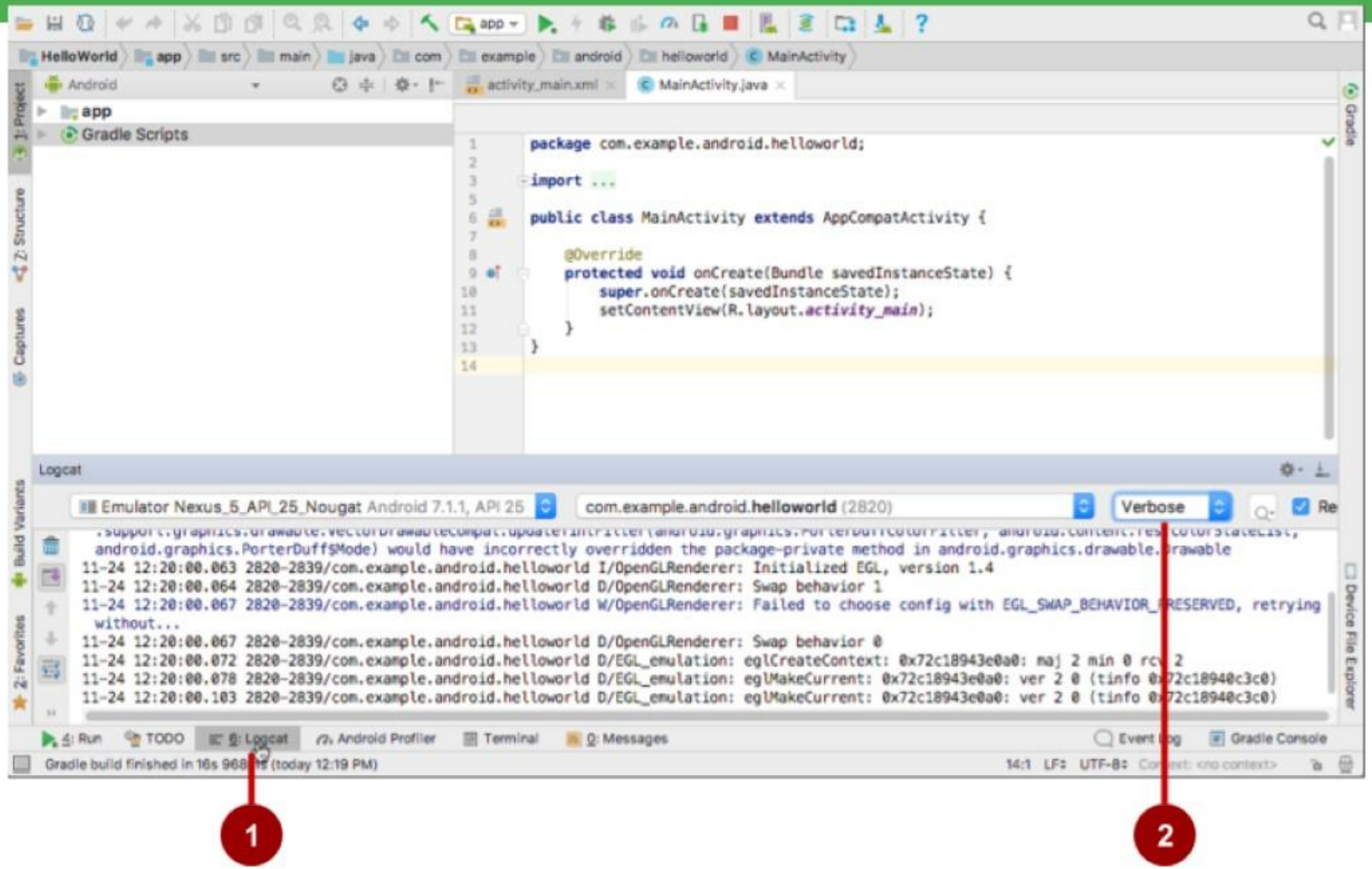


Adding logging to your app

- As the app runs, the **Logcat** pane shows information
- Add logging statements to your app that will show up in the Logcat pane
- Set filters in **Logcat** pane to see what's important to you
- Search using tags

The Logcat pane

1. **Logcat** tab to show Logcat pane
2. Log level menu



Learn more

- System Vs User App
- Introduction to Android
- Platform Architecture
- UI Overview
- Platform Versions
- Supporting Different Platform Versions
- Android Studio User's Guide

Thank you

sherinmoussa@cis.asu.edu.eg