

MOBILE DEVELOPMENT

SETUP

CONTENTS

Android app's anatomy

Tools for constructing Android apps

AVD - Emulator: disk images

Android studio: Hello world app

Android emulator - Looking under the hood

Android emulator

ANDROID APP'S ANATOMY

Android applications are usually created using the Java programming language

Apps must import various Android Libraries (such as android.jar, maps.jar, etc) to gain the functionality needed to work inside Android OS

Android apps are made of multiple elements such as: user-defined classes, android jars, third-party libraries, XML files defining the UIs or views, multimedia resources, data assets such as disk files, external arrays and strings, databases, and finally a Manifest summarizing the 'anatomy' and permissions requested by the app.

The various app components are given to the compiler to obtain a single signed and deployable Android Package (an .apk file)

Like “.class” files in Java, “.apk” files are the byte-code version of the app that finally will be ‘executed’ by interpretation inside either a Dalvik Virtual Machine (DVM) or an Android-Runtime Engine (ART).

ANDROID APP'S ANATOMY

(Dalvik virtual machine vs. Android runtime)

The Dalvik Virtual Machine is a Just-in-Time (JIT) runtime environment (similar to the Oracle's Java Virtual Machine JVM) that interprets Android byte-code only when it's needed

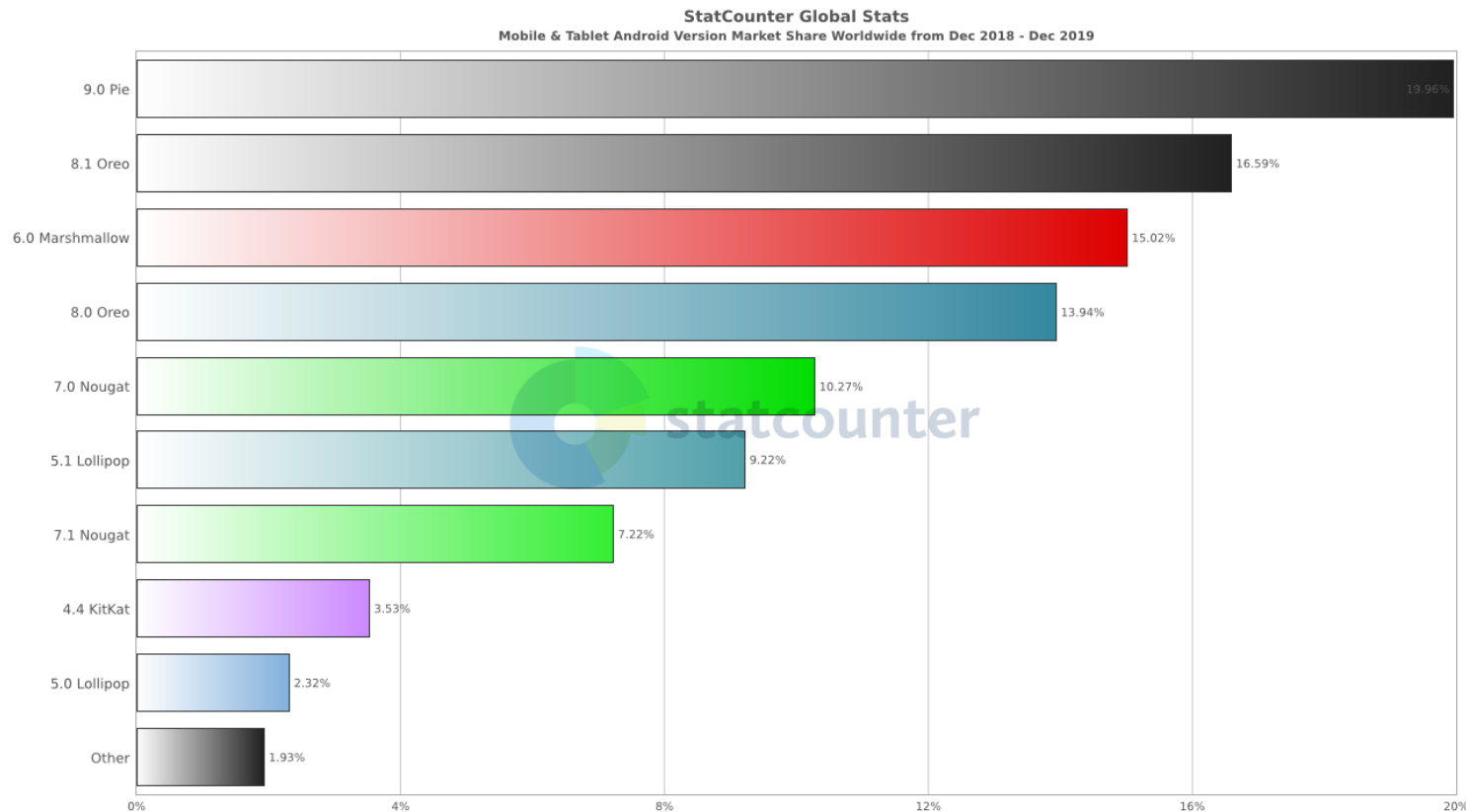
The newer ART (introduced as an option in Android 4.4 KitKat) is an anticipatory or Ahead-of-Time (AOT) environment that compiles code before it is actually needed

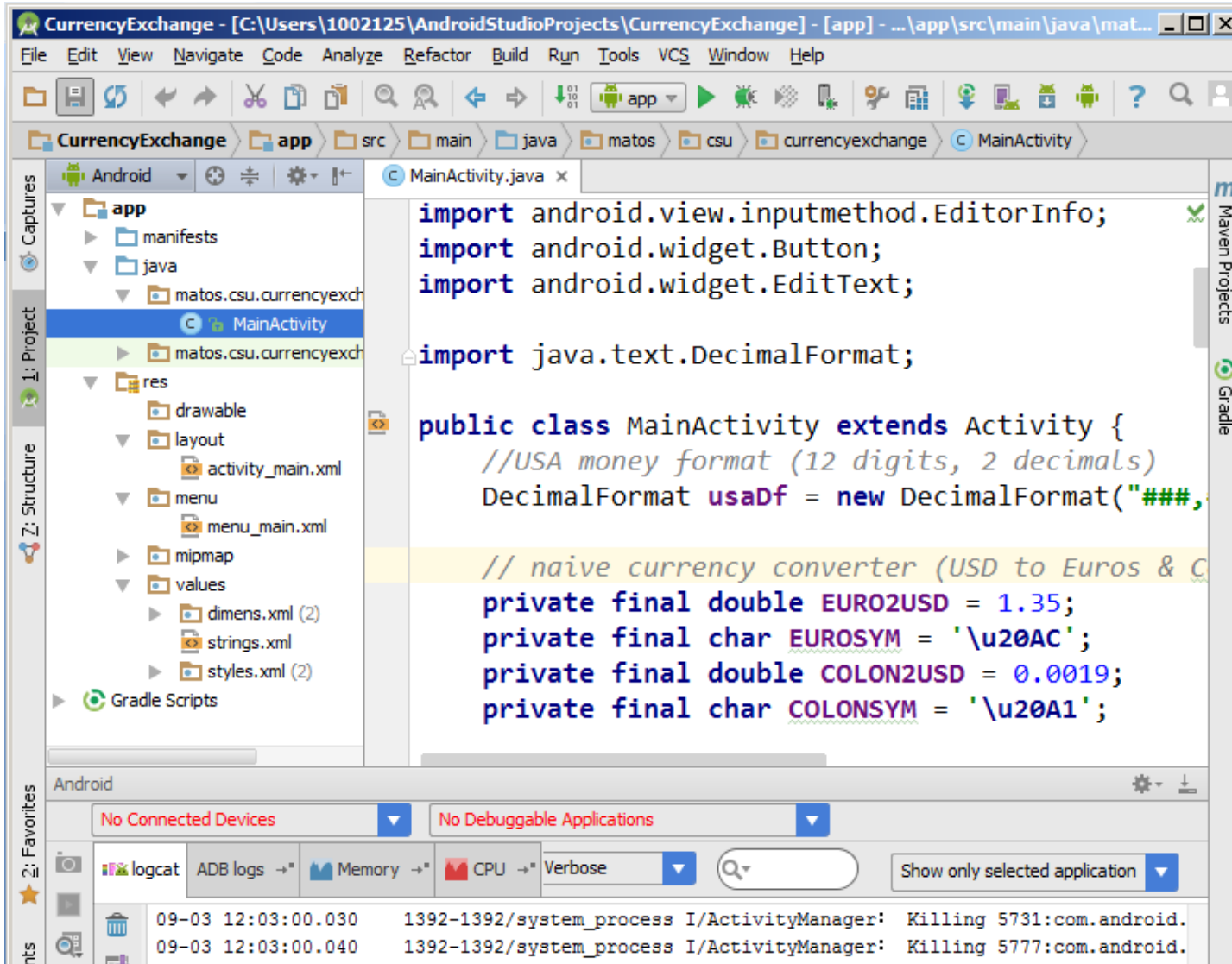
ART promises:

- enhanced performance and battery efficiency,
- improved garbage collection,
- better debugging facilities,
- Improved diagnostic detail in exceptions and crash reports.

ANDROID APP'S ANATOMY (Android SDK)

SDKs are named after types of desserts. Available versions at the time of writing are





TOOLS FOR CONSTRUCTING ANDROID APPS

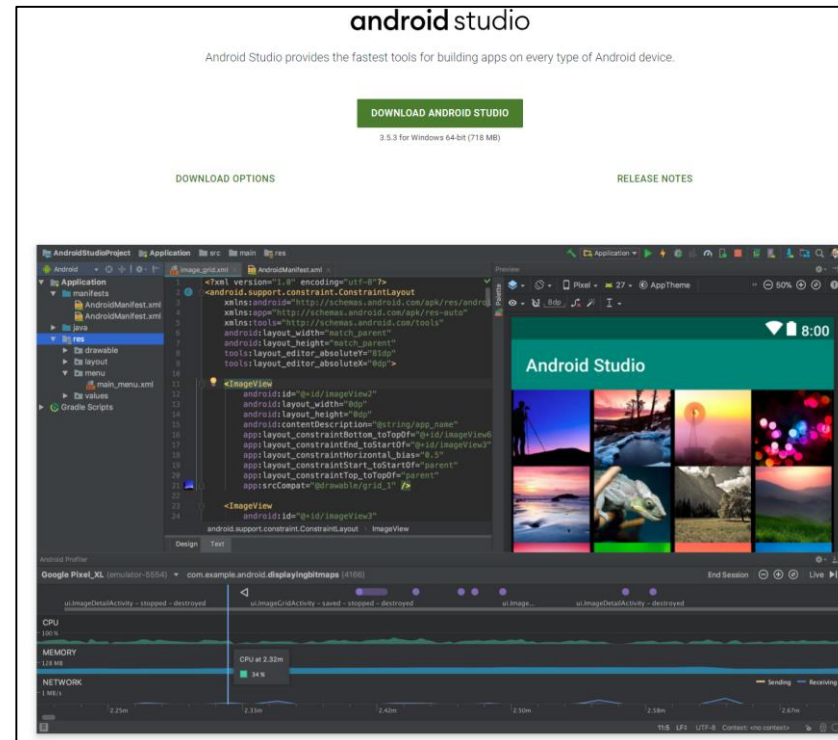
Android apps are made out of many components. The use of an IDE is strongly suggested to assist the developer in creating an Android solution

Android Studio is a new Android-only development environment based on IntelliJ IDEA. It is the 'preferred' IDE platform for Android development

TOOLS FOR CONSTRUCTING ANDROID APPS (SETTING UP)

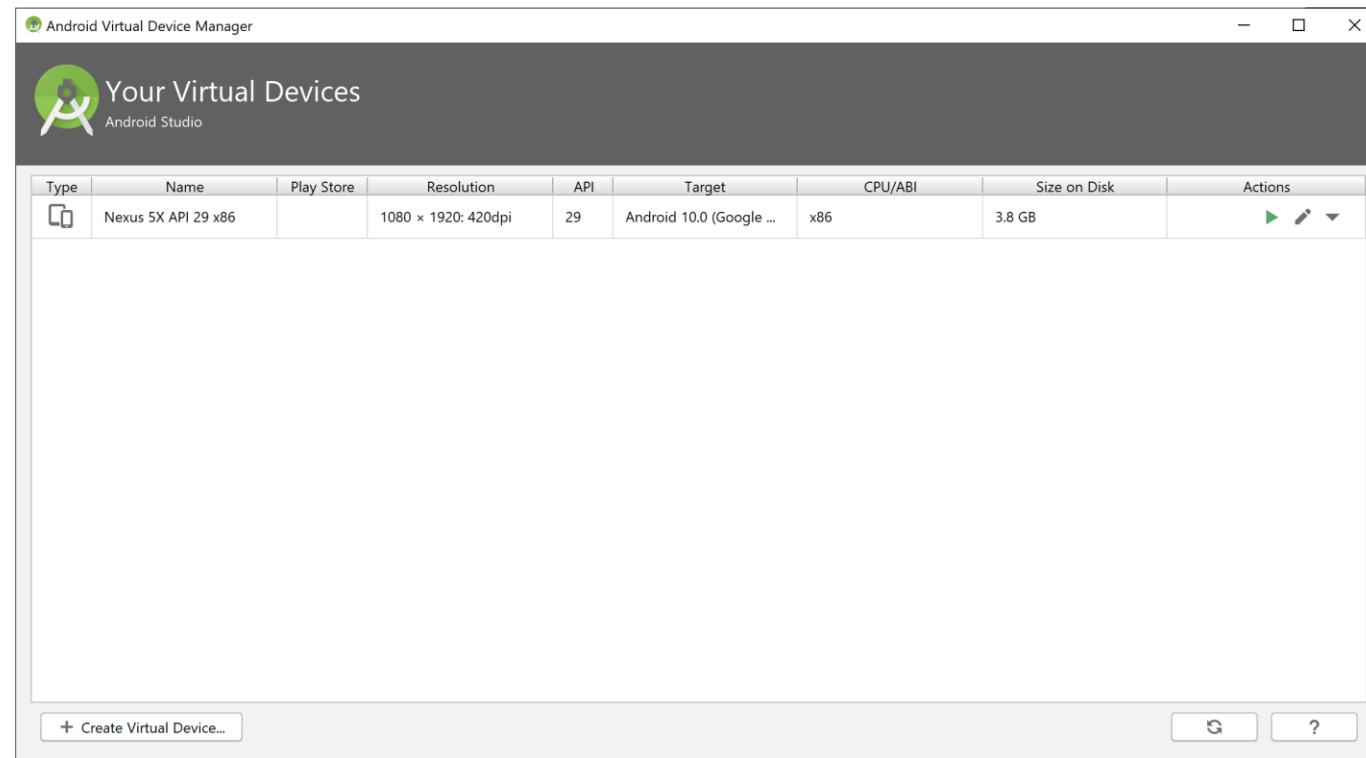
Download IDE from <https://developer.android.com/studio>

Run the executable, you are (almost) done!



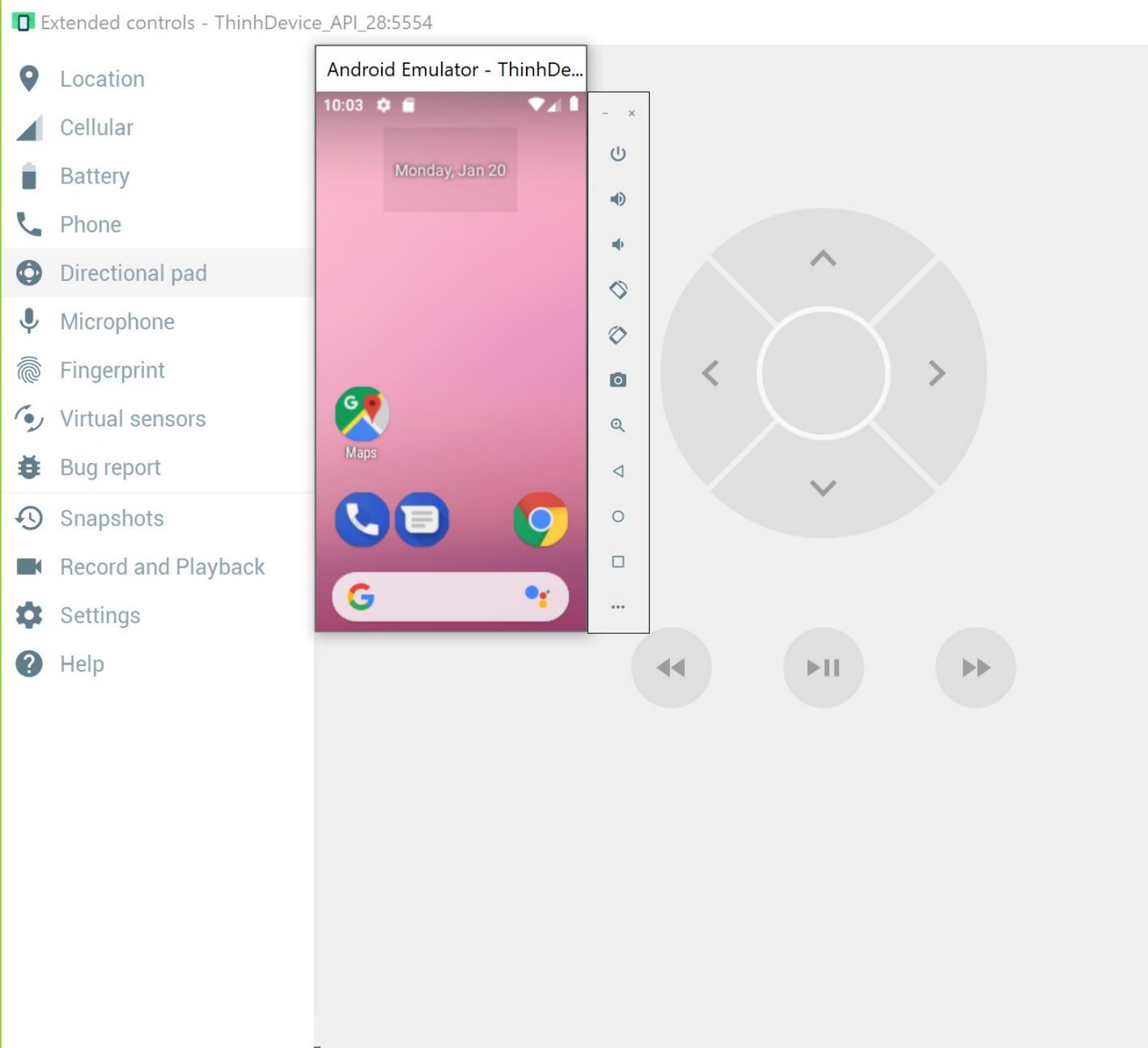
TOOLS FOR CONSTRUCTING ANDROID APPS (Working with virtual devices)

The Android Studio process to create, edit, remove, and execute AVDs is like the strategy already discussed for Eclipse-ADT (only cosmetic differences on the GUI)



TOOLS FOR CONSTRUCTING ANDROID APPS (Working with virtual devices)

EXAMPLE OF AN AVD EMULATOR
WEARING A SKIN



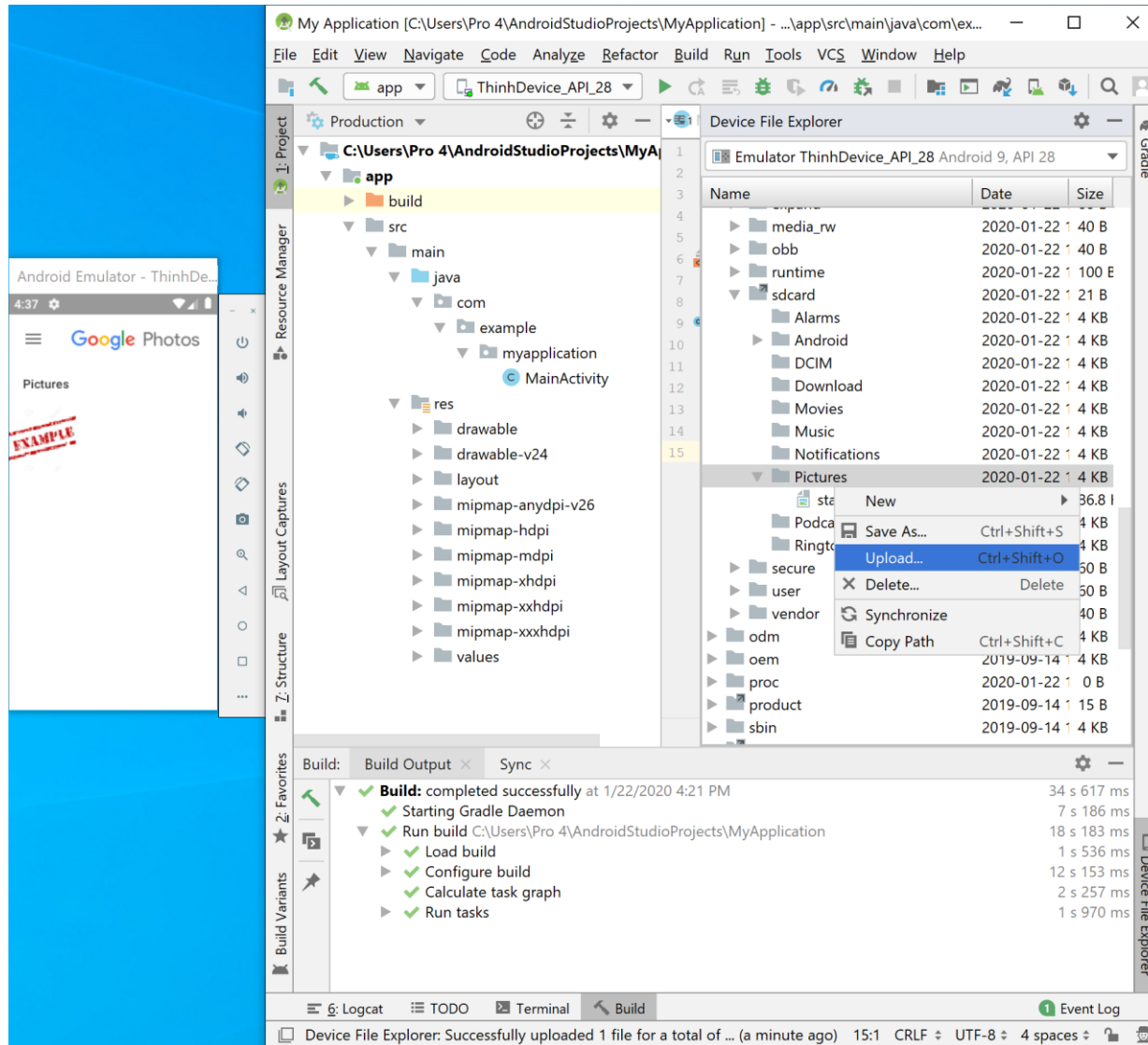
<input type="checkbox"/> Name	Date modified	Type	Size
data	1/20/2020 10:05 PM	File folder	
snapshots	1/20/2020 9:59 PM	File folder	
tmpAdbCmds	1/20/2020 10:15 PM	File folder	
AVD.conf	1/20/2020 10:16 PM	CONF File	1 KB
cache.img	1/20/2020 9:59 PM	Disc Image File	67,584 KB
cache.img.qcow2	1/20/2020 9:59 PM	QCOW2 File	193 KB
config.ini	1/20/2020 9:59 PM	Configuration settings	2 KB
emulator-user.ini	1/20/2020 10:21 PM	Configuration settings	1 KB
emu-launch-params.txt	1/20/2020 10:15 PM	Text Document	1 KB
encryptionkey.img	1/20/2020 9:55 PM	Disc Image File	1,024 KB
encryptionkey.img.qcow2	1/20/2020 9:59 PM	QCOW2 File	384 KB
hardware-qemu.ini	1/20/2020 10:15 PM	Configuration settings	4 KB
multiinstance.lock	1/20/2020 10:15 PM	LOCK File	0 KB
read-snapshot.txt	1/20/2020 10:15 PM	Text Document	0 KB
sdcard.img	1/20/2020 9:59 PM	Disc Image File	524,288 KB
sdcard.img.qcow2	1/20/2020 10:21 PM	QCOW2 File	512 KB
userdata.img	1/20/2020 9:56 PM	Disc Image File	1,024 KB
userdata-qemu.img	1/20/2020 9:59 PM	Disc Image File	819,200 KB
userdata-qemu.img.qcow2	1/20/2020 10:21 PM	QCOW2 File	224,384 KB
version_num.cache	1/20/2020 9:59 PM	CACHE File	1 KB

AVD - EMULATOR: DISK IMAGE

When you create a Virtual Device, the SDK makes several disk images containing among others:

- OS kernel,
- the Android system,
- user data (userdata-qemu.img)
- simulated SD card (sdcard.img)

By default, the Emulator searches for the disk images in the private storage area of the AVD in use, for instance the “ThinhDevive_API_28” AVD is at: C:\Users\Pro 4\.android\avd\ThinhDevice_API_28.avd



AVD - EMULATOR: DISK IMAGE (Transferring files to/from emulator's SD card)

Android-Studio uses the 'Device
File Explorer' submenu

Expand the mnt (mounted devices)
folder

Expand the sdcard folder

Right-mouse at the subfolder you
want to upload to

Click upload and choose a file
stored in your PC

AVD - EMULATOR: DISK IMAGE (Locate your 'android-sdk' & AVD folder)

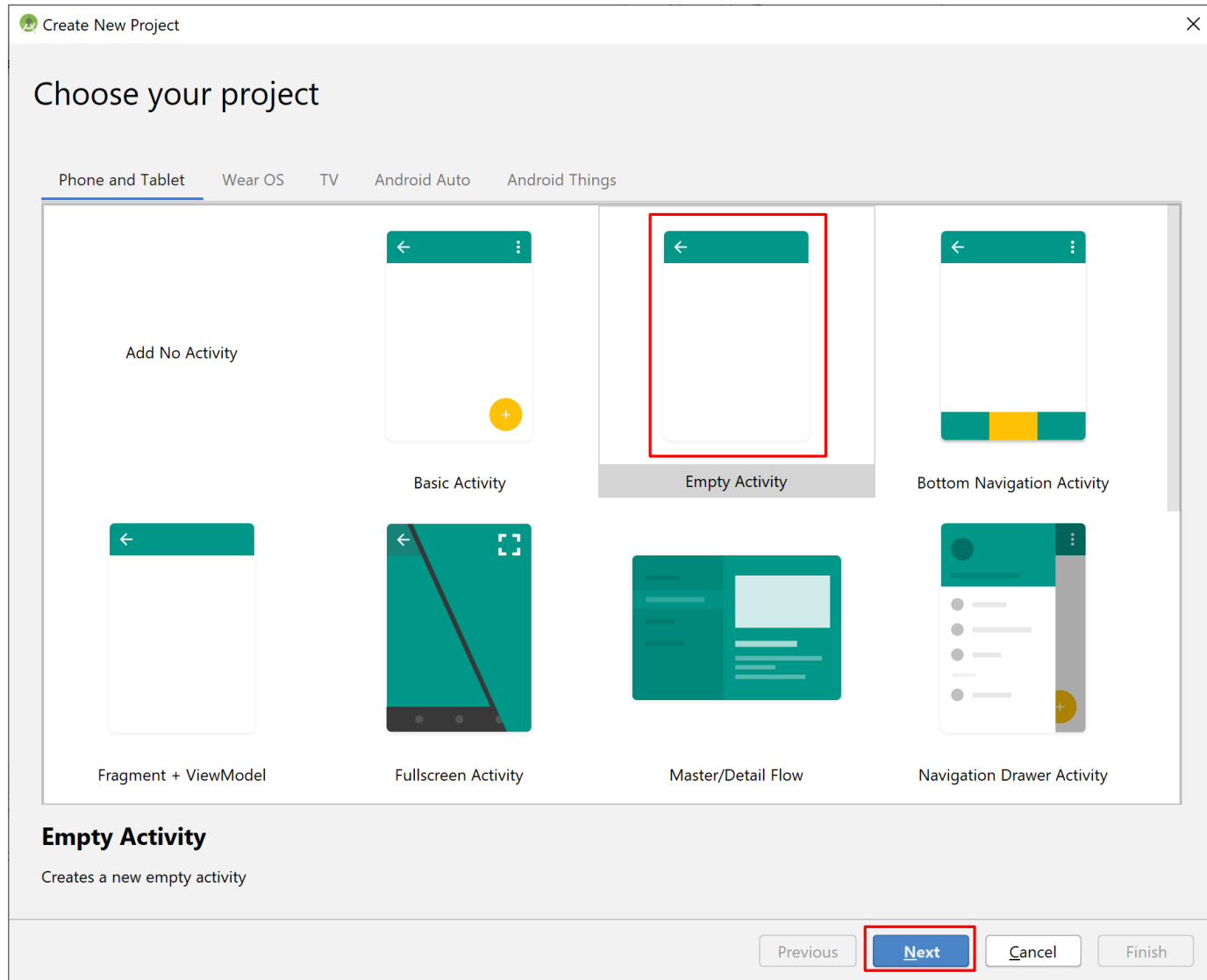
After you complete your setup look for the following two subdirectories in your PC's file system

C:\Users\Pro 4\.android\avd

Name	Date modified	Type	Size
Nexus_5X_API_29_x86.avd	1/20/2020 9:45 PM	File folder	
ThinhdDevice_API_28.avd	1/22/2020 4:24 PM	File folder	
Nexus_5X_API_29_x86.ini	1/20/2020 9:24 PM	Configuration settings	1 KB
ThinhdDevice_API_28.ini	1/20/2020 9:59 PM	Configuration settings	1 KB

C:\Users\Pro 4\AppData\Local\Android\Sdk

Name	Date modified	Type	Size
.temp	1/20/2020 9:56 PM	File folder	
build-tools	1/20/2020 9:23 PM	File folder	
emulator	1/20/2020 9:17 PM	File folder	
extras	1/20/2020 9:17 PM	File folder	
fonts	1/20/2020 9:32 PM	File folder	
licenses	1/20/2020 9:58 PM	File folder	
patcher	1/20/2020 9:16 PM	File folder	
platforms	1/20/2020 9:18 PM	File folder	
platform-tools	1/20/2020 9:17 PM	File folder	
skins	1/20/2020 9:49 PM	File folder	
sources	1/20/2020 9:24 PM	File folder	
system-images	1/20/2020 9:52 PM	File folder	
tools	1/20/2020 9:16 PM	File folder	
.knownPackages	1/22/2020 4:20 PM	KNOWNPACKAGES Fi...	1 KB



ANDROID STUDIO: HELLO WORLD APP

We will use Android Studio IDE to create a bare bone app.


Click on the entry: 'File -> new Project'

A wizard will guide you providing a sequence of menu driven selections.

The final product is the skeleton of your Android app.

Create New Project

Configure your project



Empty Activity

Creates a new empty activity

Name
My Application

Package name
com.example.myapplication

Save location
C:\Users\Pro 4\AndroidStudioProjects\MyApplication2

Language
Java

Minimum API level
API 28: Android 9.0 (Pie)

Information: Your app will run on < 1% of devices.
[Help me choose](#)

☐ This project will support instant apps

☒ Use androidx.* artifacts

Warning: project location should not contain whitespace, as this can cause problems with the NDK tools.

[Previous](#) [Next](#) [Cancel](#) [Finish](#)

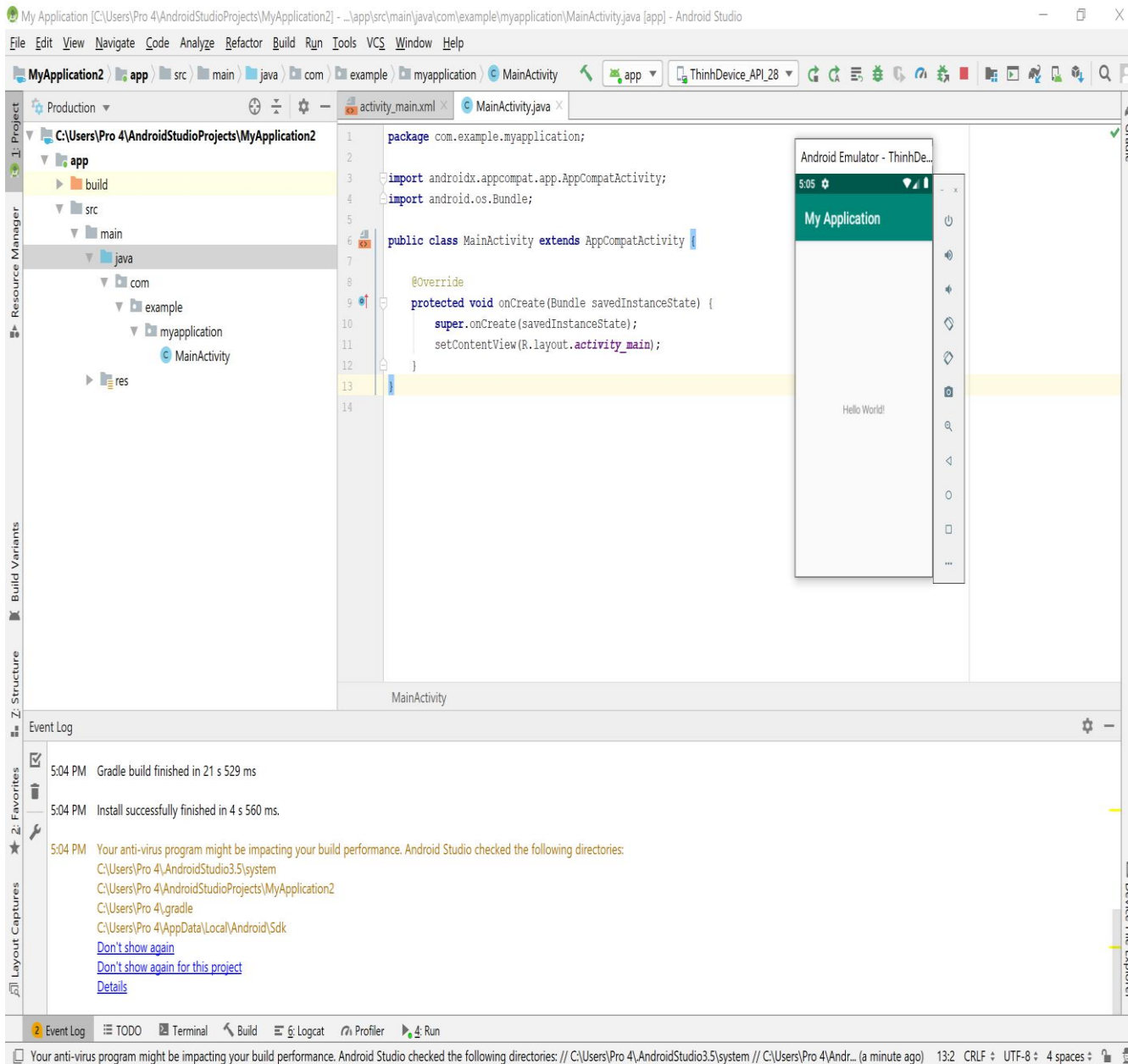
ANDROID STUDIO: HELLO WORLD APP

Enter in the Application Name box

Enter Package name

Choose other options

Click Finish



ANDROID STUDIO: HELLO WORLD APP

java/: holds your Main-Activity Java code. All other Java files for your application go here.

res/: this folder stores application resources such as drawable files, UI layout files, string values, menus, multimedia, etc.

Manifests: the Android Manifest for your project (switch to from “Production” to “Project Files”)

```
Administrator: Command Prompt - adb shell
C:\Users\Pro 4\AppData\Local\Android\Sdk\platform-tools>adb shell
generic_x86_arm:/ $ ls -l
ls: ./init.usb.rc: Permission denied
ls: ./init: Permission denied
ls: ./init.zygote32.rc: Permission denied
ls: ./init.rc: Permission denied
ls: ./ueventd.rc: Permission denied
ls: ./init.usb.configfs.rc: Permission denied
ls: ./adb_keys: Permission denied
ls: ./init.envirion.rc: Permission denied
ls: ./metadata: Permission denied
total 52
dr-xr-xr-x 60 root root 0 2020-01-22 20:50 acct
lrw-r--r-- 1 root root 11 2019-09-14 12:22 bin -> /system/bin
lrw-r--r-- 1 root root 50 2019-09-14 12:22 bugreports -> /data/user_de/0/com.android.shell/files/bugreports
drwxrwx--- 2 system cache 4096 2019-09-14 11:52 cache
lrw-r--r-- 1 root root 13 2019-09-14 12:22 charger -> /sbin/charger
drwxr-xr-x 4 root root 0 2020-01-22 20:50 config
lrw-r--r-- 1 root root 17 2019-09-14 12:22 d -> /sys/kernel/debug
drwxrwx--x 38 system system 4096 2020-01-20 21:59 data
lrw----- 1 root root 23 2019-09-14 12:22 default.prop -> system/etc/prop.default
drwxr-xr-x 15 root root 2600 2020-01-22 20:50 dev
lrw-r--r-- 1 root root 11 2019-09-14 12:22 etc -> /system/etc
drwx----- 2 root root 16384 2019-09-14 12:22 lost+found
drwxr-xr-x 11 root system 240 2020-01-22 20:50 mnt
drwxr-xr-x 2 root root 4096 2019-09-14 11:52 odm
drwxr-xr-x 2 root root 4096 2019-09-14 11:52 oem
dr-xr-xr-x 217 root root 0 2020-01-22 20:50 proc
lrw-r--r-- 1 root root 15 2019-09-14 12:22 product -> /system/product
drwxr-x--- 2 root shell 4096 2019-09-14 12:14 sbin
lrw-r--r-- 1 root root 21 2019-09-14 12:22 sdcard -> /storage/self/primary
drwxr-xr-x 4 root root 80 2020-01-22 20:50 storage
dr-xr-xr-x 12 root root 0 2020-01-22 20:50 sys
drwxr-xr-x 14 root root 4096 2019-09-14 12:22 system
drwxr-xr-x 8 root root 4096 2019-09-14 12:12 vendor
1|generic_x86_arm:/ $
```

ANDROID EMULATOR – LOOKING UNDER THE HOOD

Although it is not necessary, a developer may gain access to some of innermost parts of Android OS

Use the AVD Manager to start one of your AVDs (say ThinkDevice_API_28)

At DOS command (cmd in Windows) prompt level run Android Debug Bridge (adb) application: adb shell

“adb” is a tool located in directory C:\Your-SDK-Folder\Android\android-sdk\platform-tools\

Administrator: Command Prompt

```
C:\Users\Pro 4\AppData\Local\Android\Sdk\platform-tools>adb devices
List of devices attached
emulator-5554    device

C:\Users\Pro 4\AppData\Local\Android\Sdk\platform-tools>
```

ANDROID EMULATOR — LOOKING UNDER THE HOOD

If more than one emulator is running (or your phone is physically connected to the computer using the USB cable) you need to identify the target.

1. Get a list of attached devices `adb devices`

- List of devices attached
 - emulator-5554 device
 - emulator-5556 device
 - HT845GZ45737 device

2. Run the adb as follows: `adb -s emulator-5554 shell`

ANDROID EMULATOR – LOOKING UNDER THE HOOD

Android accepts a number of Linux shell commands including the useful set below

- ls show directory (alphabetical order)
- mkdir make a directory
- rmdir remove directory
- rm -r to delete folders with files
- rm remove files
- mv moving and renaming files
- cat displaying short files
- cd change current directory
- pwd find out what directory you are in
- df shows available disk space
- chmod changes permissions on a file
- date display date
- exit terminate session

There is no copy (cp) command in Android, but you could use cat instead. For instance:

- # cat data/app/theInstalledApp.apk > cache/theInstalledApp.apk

ANDROID EMULATOR – LOOKING UNDER THE HOOD

If you want to transfer an app that is currently installed in your rooted developer's phone to the emulator, follow the next steps:

- Run command shell: > adb devices (find out your hardware's id, say HT096P800176)
- Pull the file from the device to your computer's file system. Enter the command
 - adb -s HT096P800176 pull data/app/theInstalledApp.apk c:\theInstalledApp.apk
- Disconnect your Android phone
- Run an instance of the Emulator
- Now install the app on the emulator using the command
 - adb -s emulator-5554 install c:\theInstalledApp.apk
 - adb -s emulator-5554 uninstall data/app/theInstalledApp.apk

Should see a message indicating the size of the installed package, and finally: Success.

A file manager app allows you to easily administer the folders and files in the system's flash memory and SD card of your Android device (or emulator)

ANDROID EMULATOR

(Sending text messages from PC to the emulator)

Start the emulator.

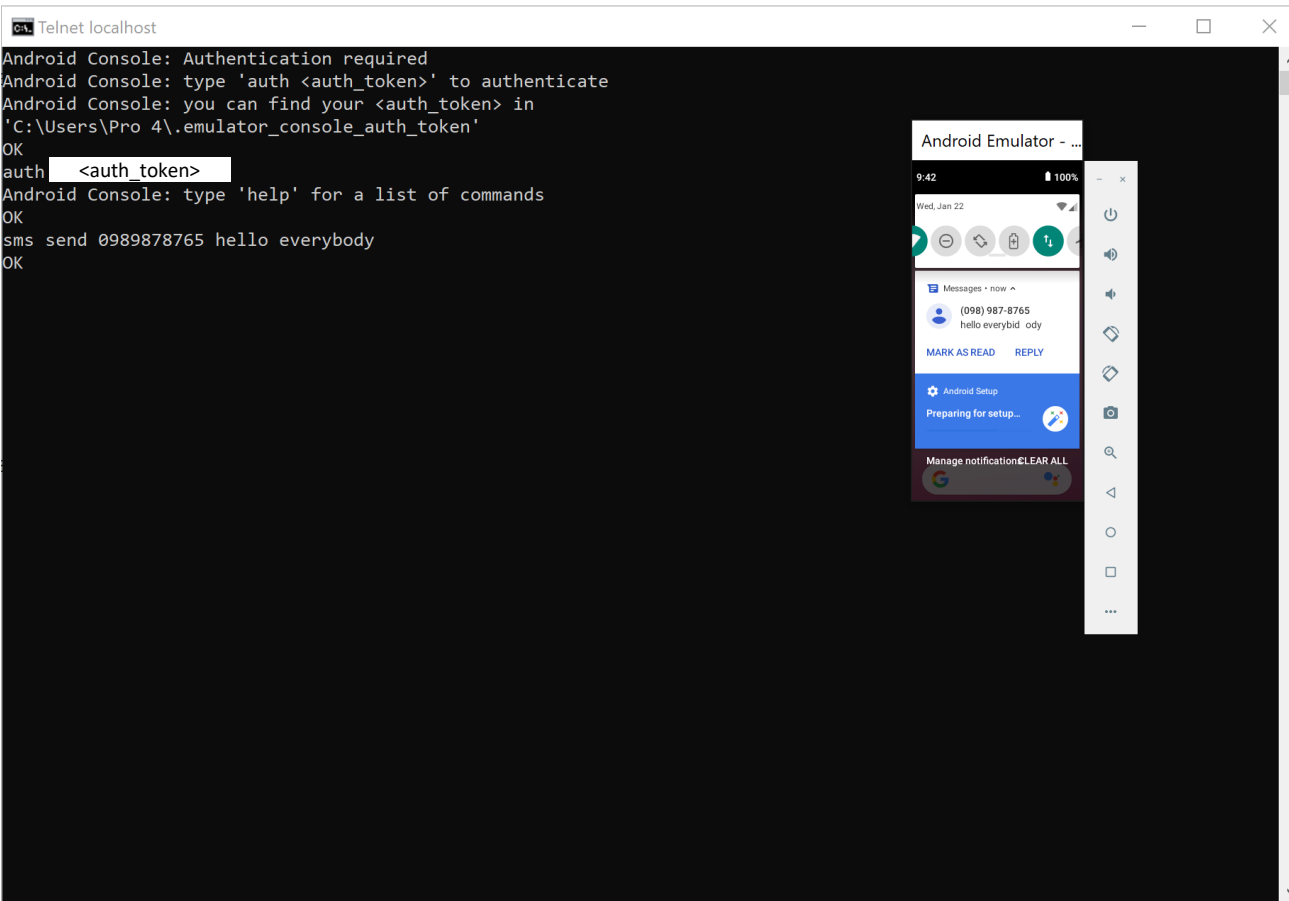
Open a new DOS command shell and type `c:> adb devices`

This way you get to know the emulator's numeric port id (usually 5554, 5556, and so on)

Initiate a Telnet session with the sender at localhost, port 5556 identifies an active (receiving) Android emulator. Type the command: `c:> telnet localhost 5554` (**MUST** turn on telnet client)

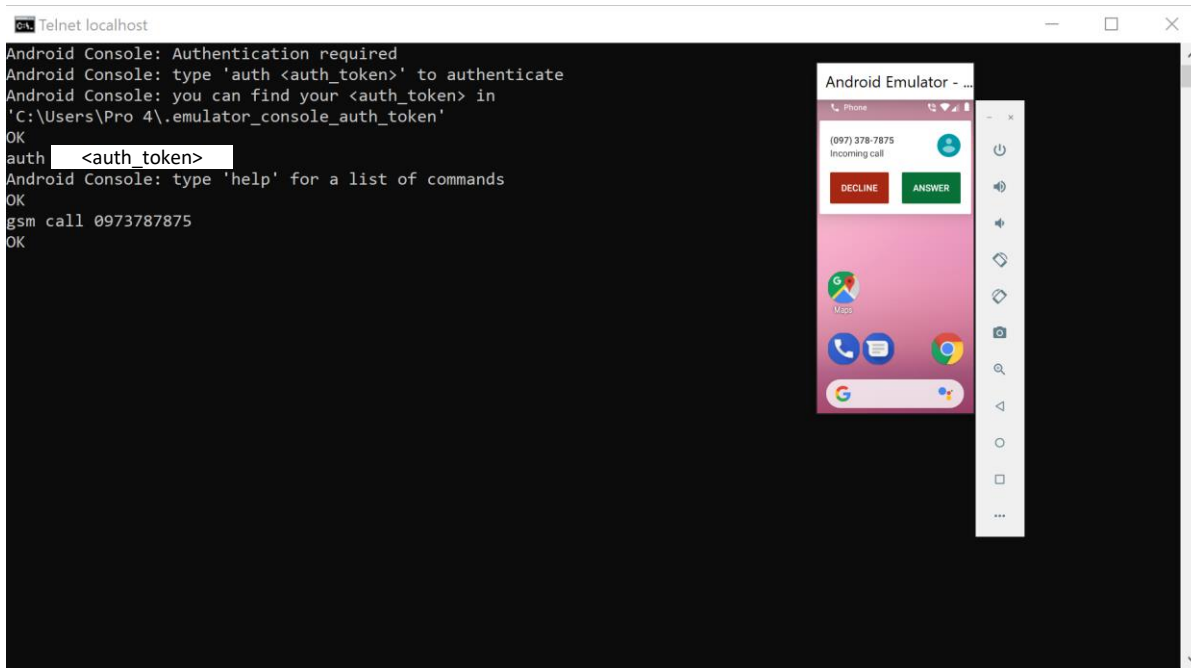
You type "auth <auth_token>" (<auth_token> in C:/Users/.../.emulator_console_auth_token)

After authentication, you can send text message to emulator on port 5554 (no quotes needed for the message): `sms send <Sender's phone number> <text message>`



ANDROID EMULATOR

(Making a phone call from PC to emulator)



```
Telnet localhost
Android Console: Authentication required
Android Console: type 'auth <auth_token>' to authenticate
Android Console: you can find your <auth_token> in
'C:\Users\Pro 4\.emulator_console_auth_token'
OK
auth <auth_token>
OK
Android Console: type 'help' for a list of commands
OK
gsm call 0973787875
OK
```

Start the emulator.

Open a new DOS command shell and type `c:> adb devices`

This way you get to know the emulator's numeric port id (usually 5554, 5556, and so on)

Initiate a Telnet session with the sender at localhost, port 5556 identifies an active (receiving) Android emulator. Type the command: `c:> telnet localhost 5554` (**MUST** turn on telnet client)

You type "auth <auth_token>" (<auth_token> in C:/Users/.../.emulator_console_auth_token)

After authentication, you can send text message to emulator on port 5554 (no quotes needed for the message): `gsm call <Sender's phone number>`

ANDROID EMULATOR

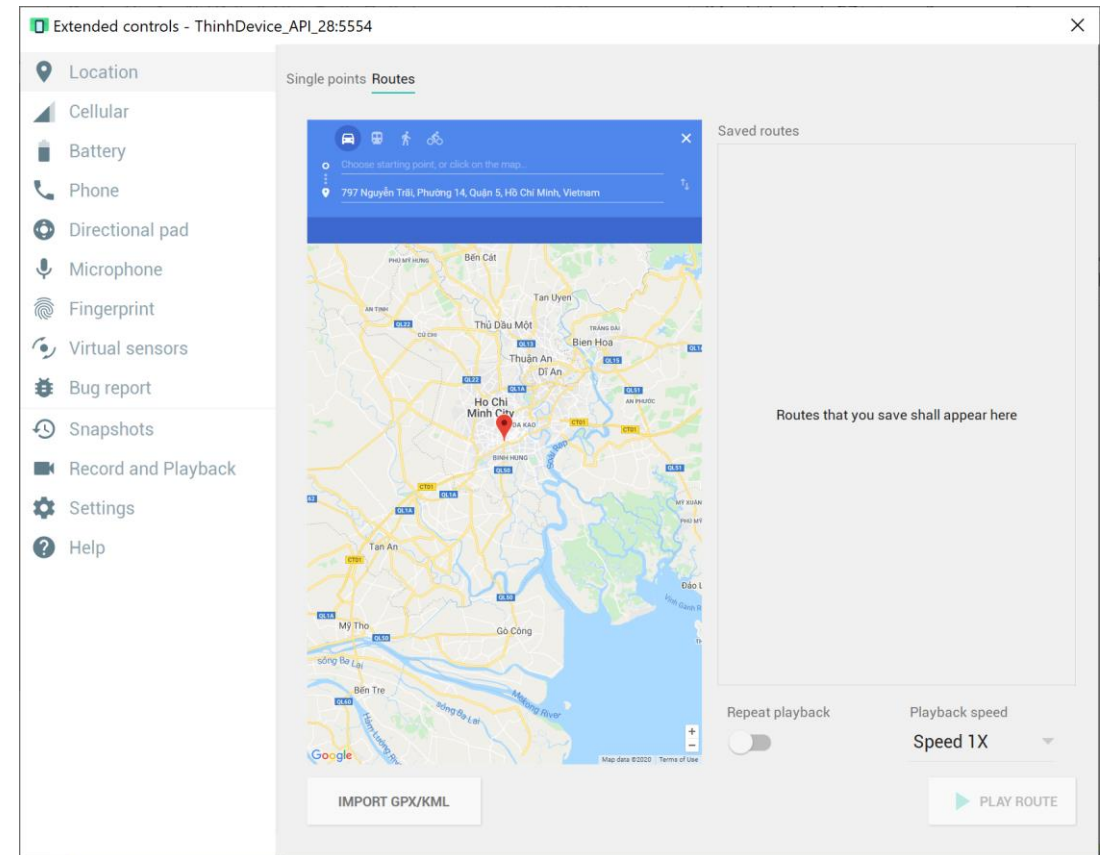
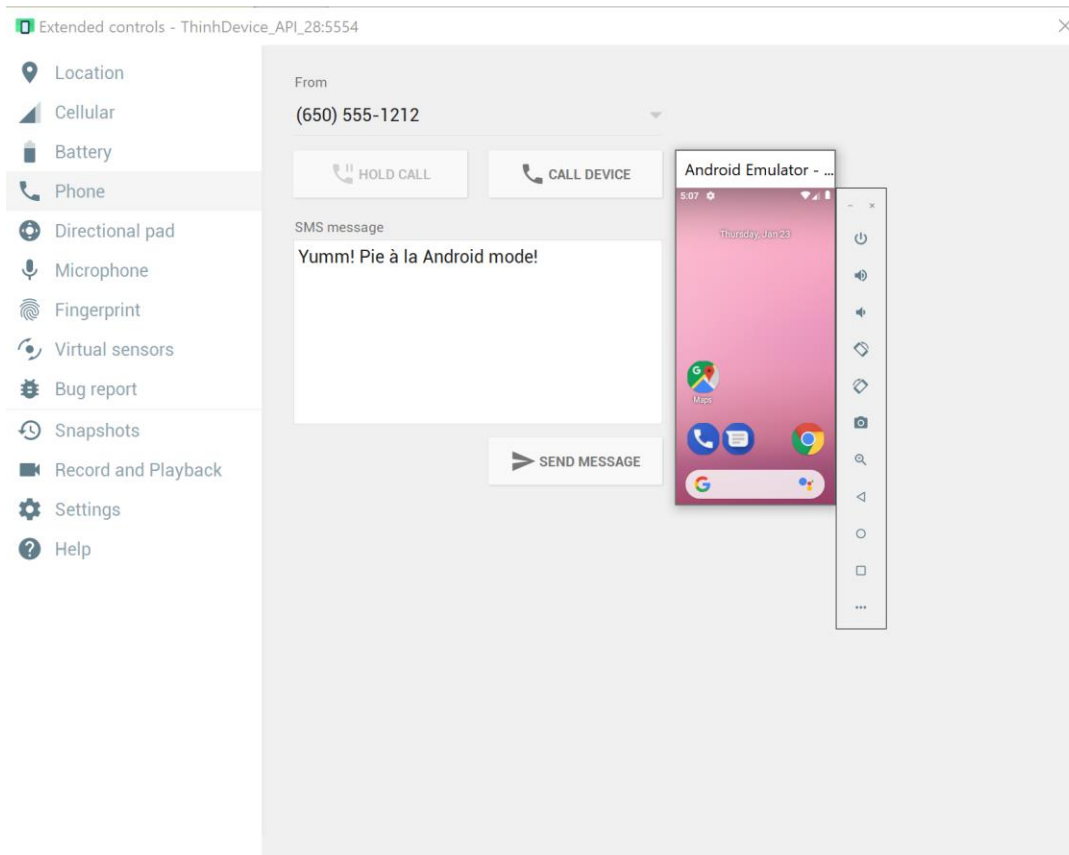
(using extended control in emulator)

It is much simpler to test telephony operations (SMS/Voice) as well as GPS services using the controls included in the IDE (both AS and Eclipse)

1. Telephony Status - change the state of the phone's Voice and Data plans (home, roaming, searching, etc.), and simulate different kinds of network Speed and Latency (GPRS, EDGE, UTMS, etc.)
2. Telephony Actions - perform simulated phone calls and SMS messages to the emulator.
3. Location Controls - send mock location data to the emulator so that you can perform location-aware operations requiring GPS assistance.
 - Manually send individual longitude/latitude coordinates to the device. Click Manual, select the coordinate format, fill in the fields and click Send.
 - Use a GPX file describing a route for playback to the device.
 - Use a KML file to place multiple placemaker points on a map

ANDROID EMULATOR (using extended control in emulator)

Images demonstration



EXERCISES

Inquire how to:

- Connecting your Physical Device to the Computer
- Run two instances of the emulator
- Shortcut to organize all imports

