Lab on apps development for tablets, smartphones and smartwatches

Week 1: Android presentation & Java for Android

Dr. Marina Zapater, Prof. David Atienza

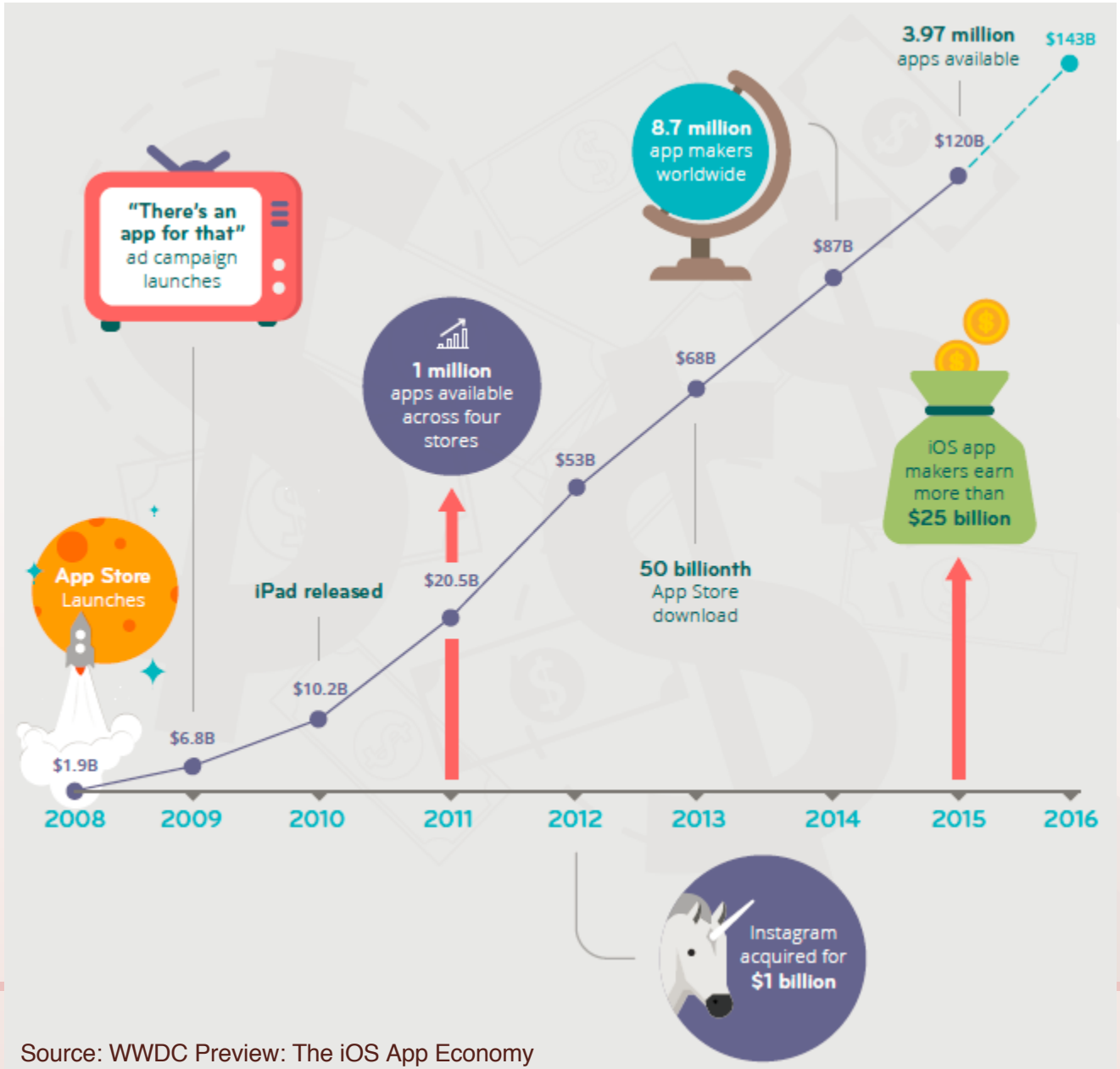
Ms. Elisabetta de Giovanni, Ms. Farnaz Forooghifar, Ms. Halima Najibi,
Mr. Dionisijie Sopic, Mr. Grégoire Surrel

Embedded Systems Laboratory (ESL) – Faculty of Engineering (STI)

Class outline

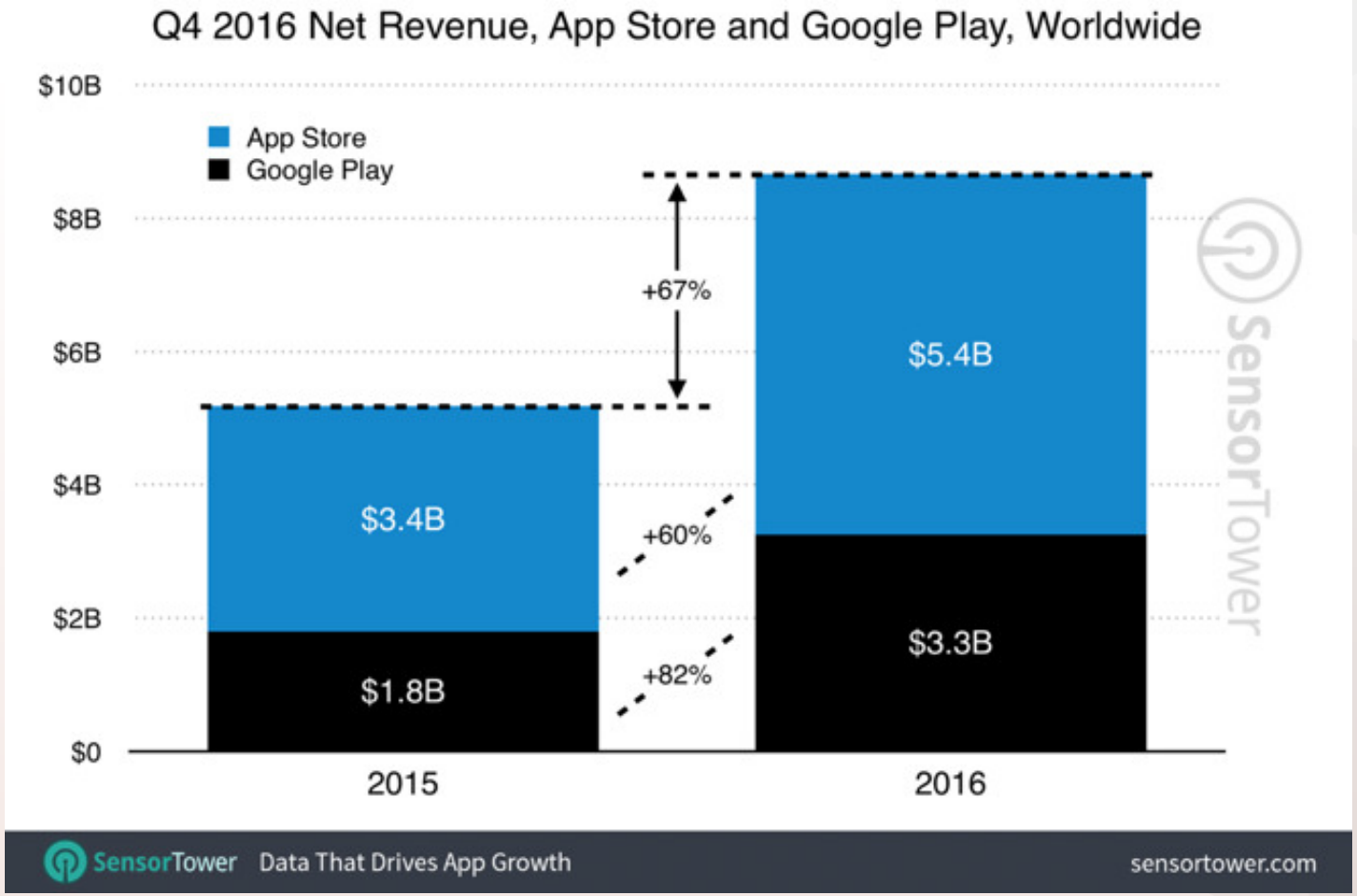
- **Android presentation:**
  - some numbers, motivation
- Java for Android Crash-course
- Gentle reminder: project groups
- Lab of Today:
  a) Learning Java for Android: some *very guided* Java exercises
  b) Create and build your first app for tablet & watch → HelloWorld!

Growth of the mobile app economy (iOS)

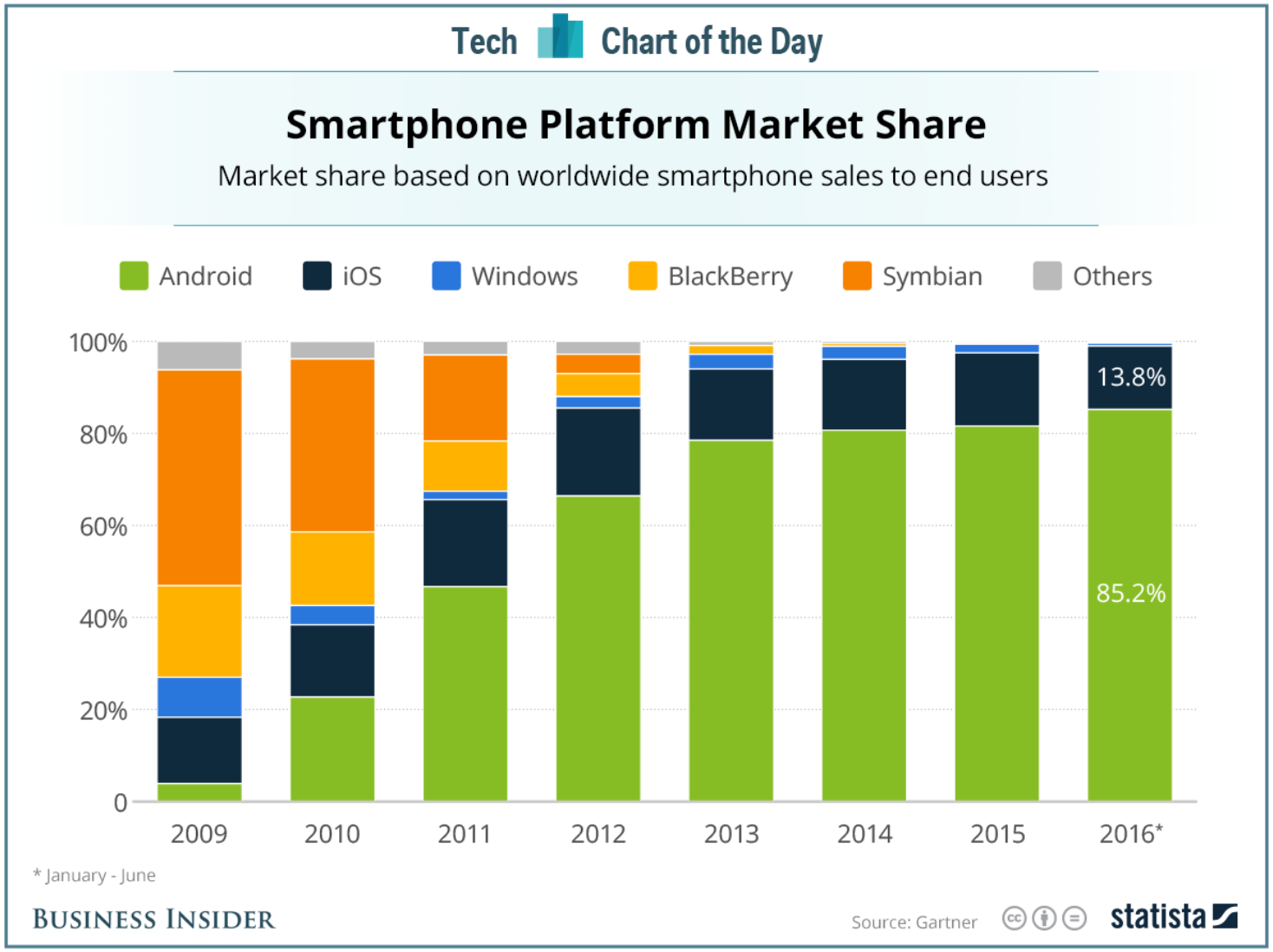


Source: WWDC Preview: The iOS App Economy

App. Store (iOS) vs. Google Play (Android)

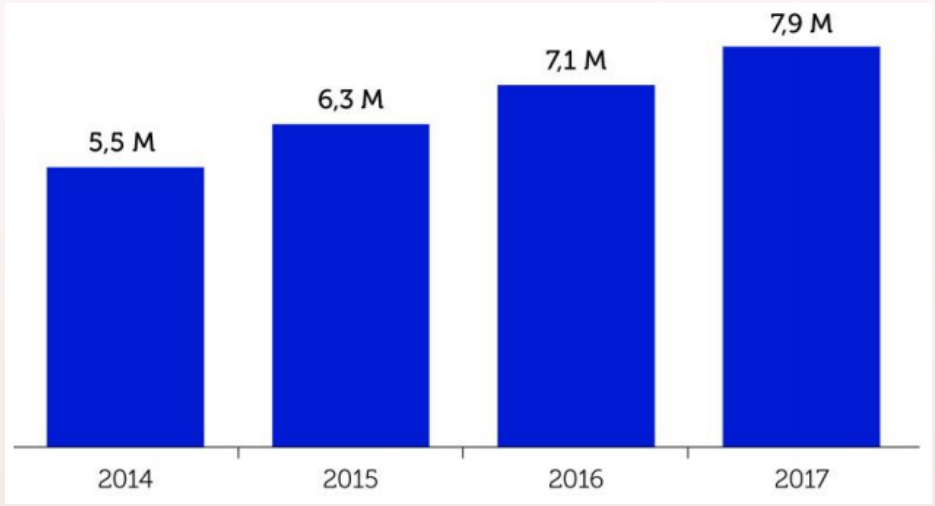


Apps market: Android & iOS are the last two standing

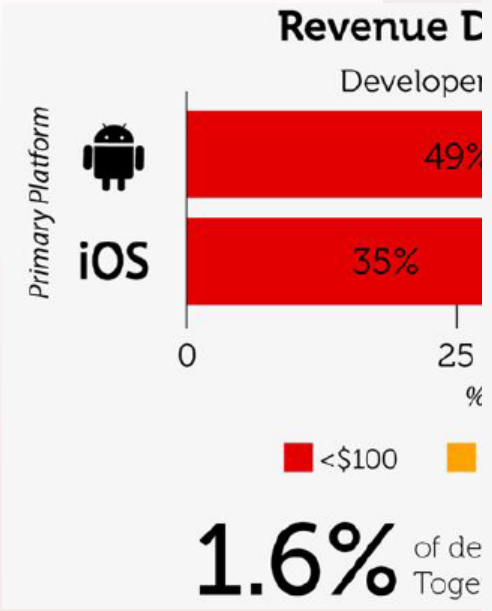


800,000 new developers per year

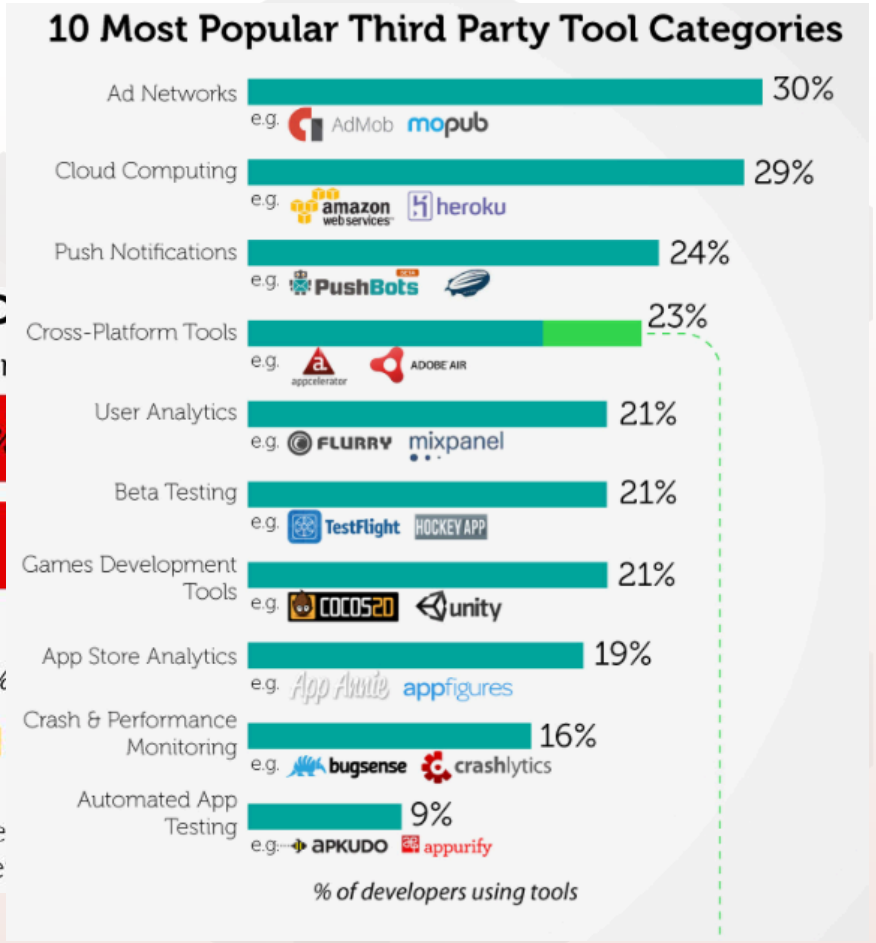
- 2% of developers earn 54% of all app. Revenue
- 64% of Android developers are below the app. poverty line
- Successful developers use more tools



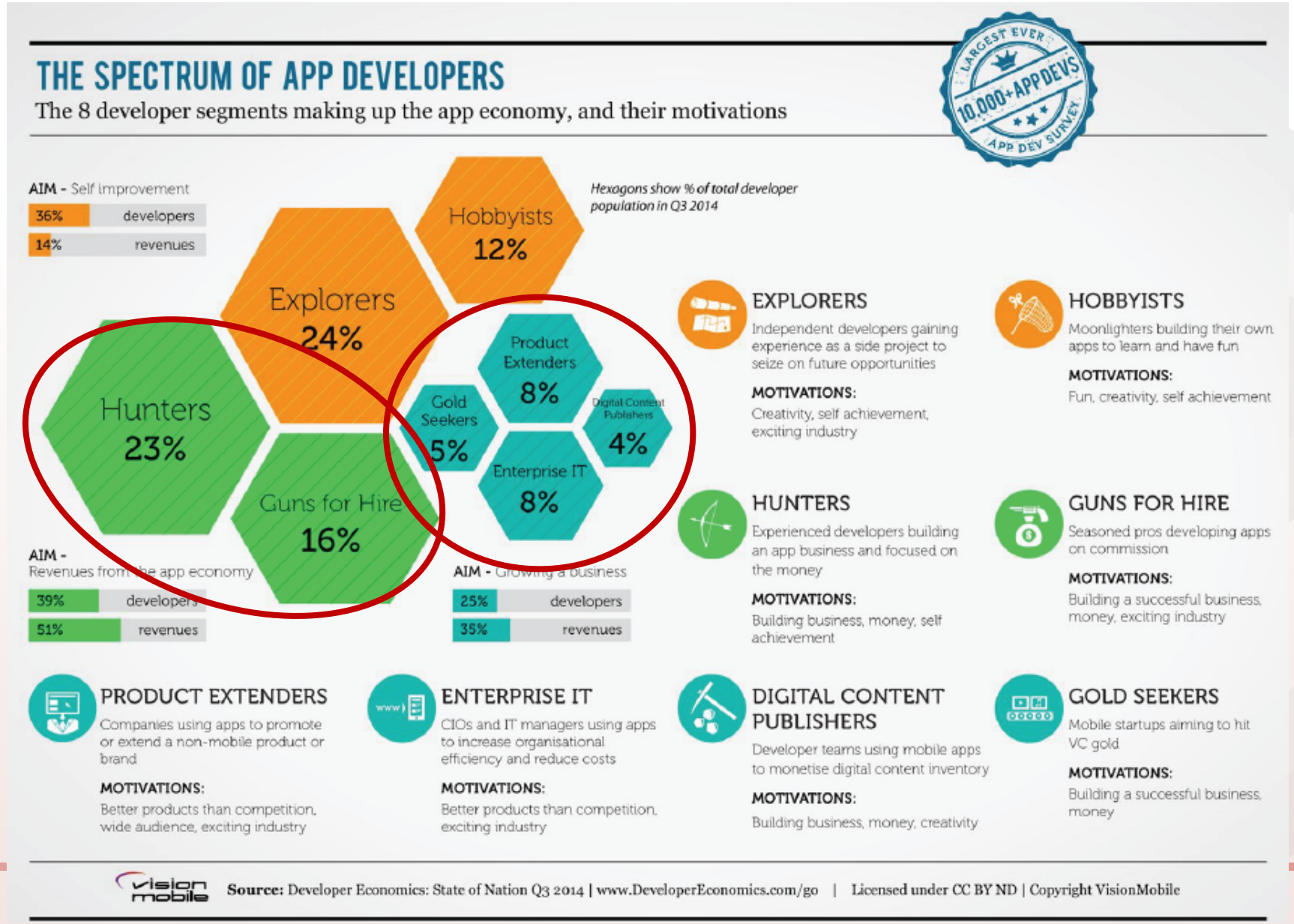
Source: VisionMobile



Source: GlobalVision



The spectrum of app developers



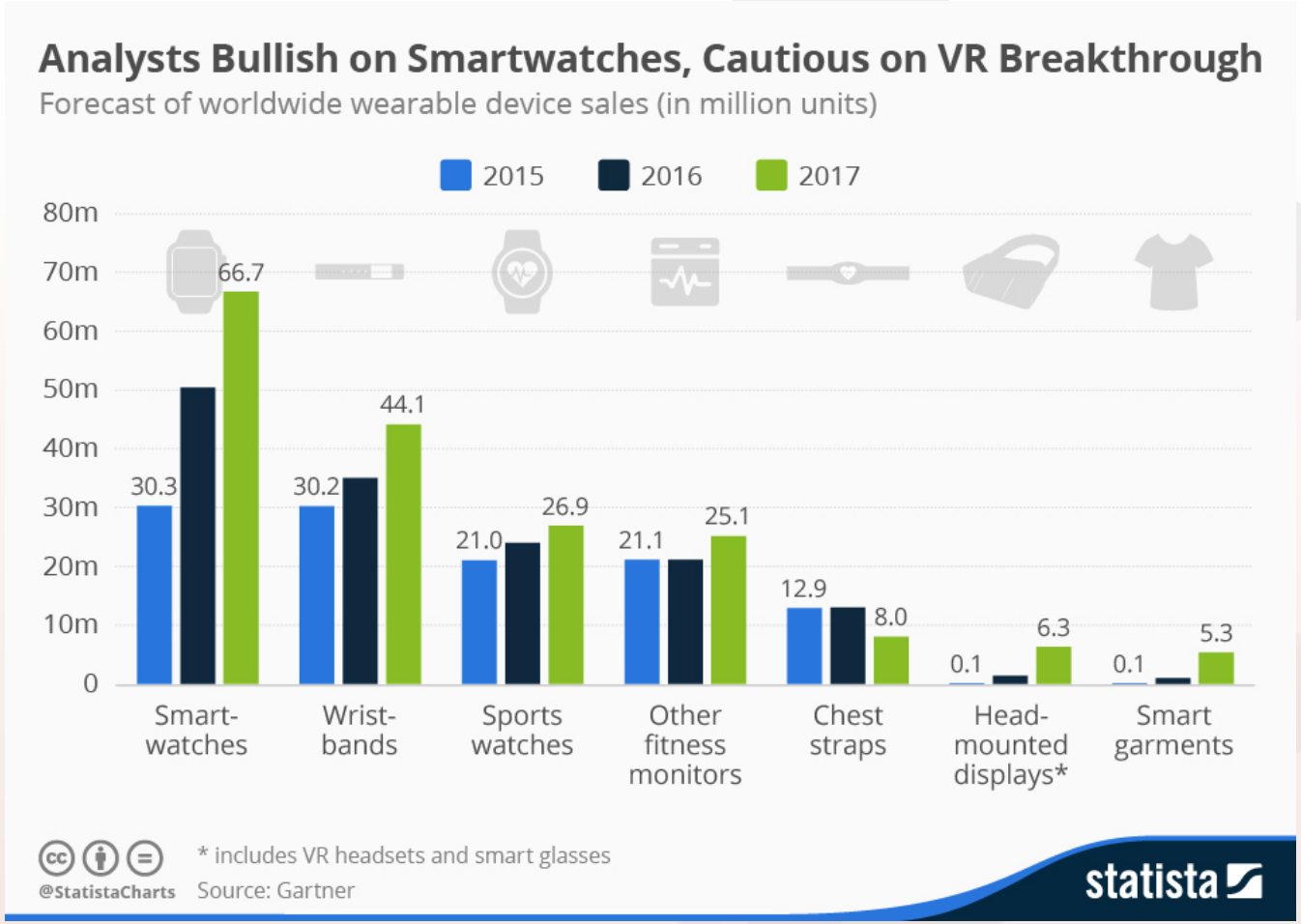
Developers are conquering… the wrist

Apple Watch

Android Wear

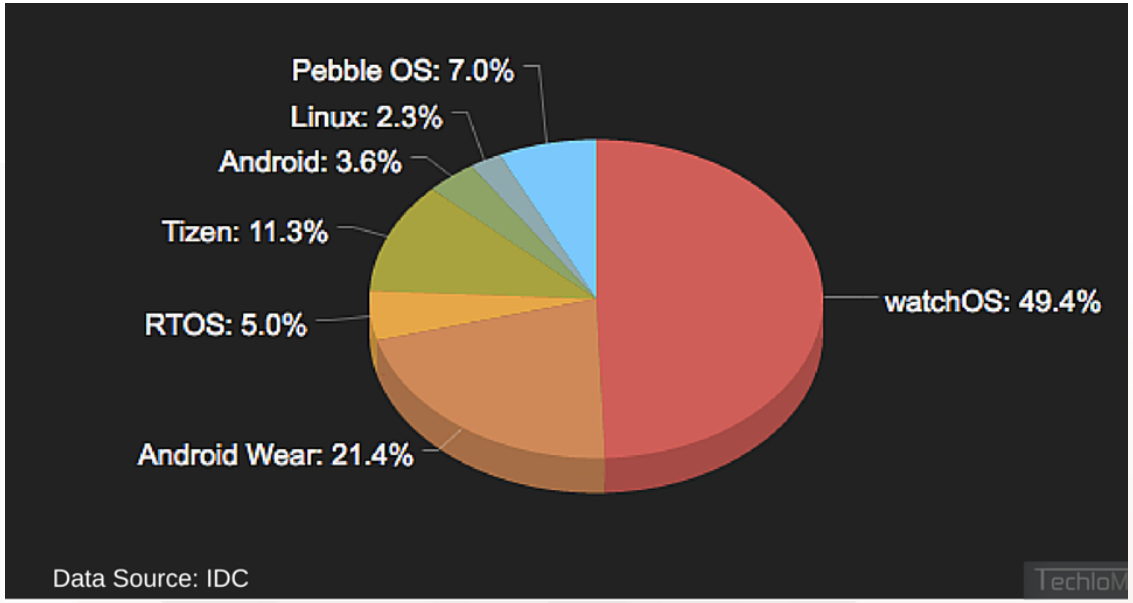
Pebble

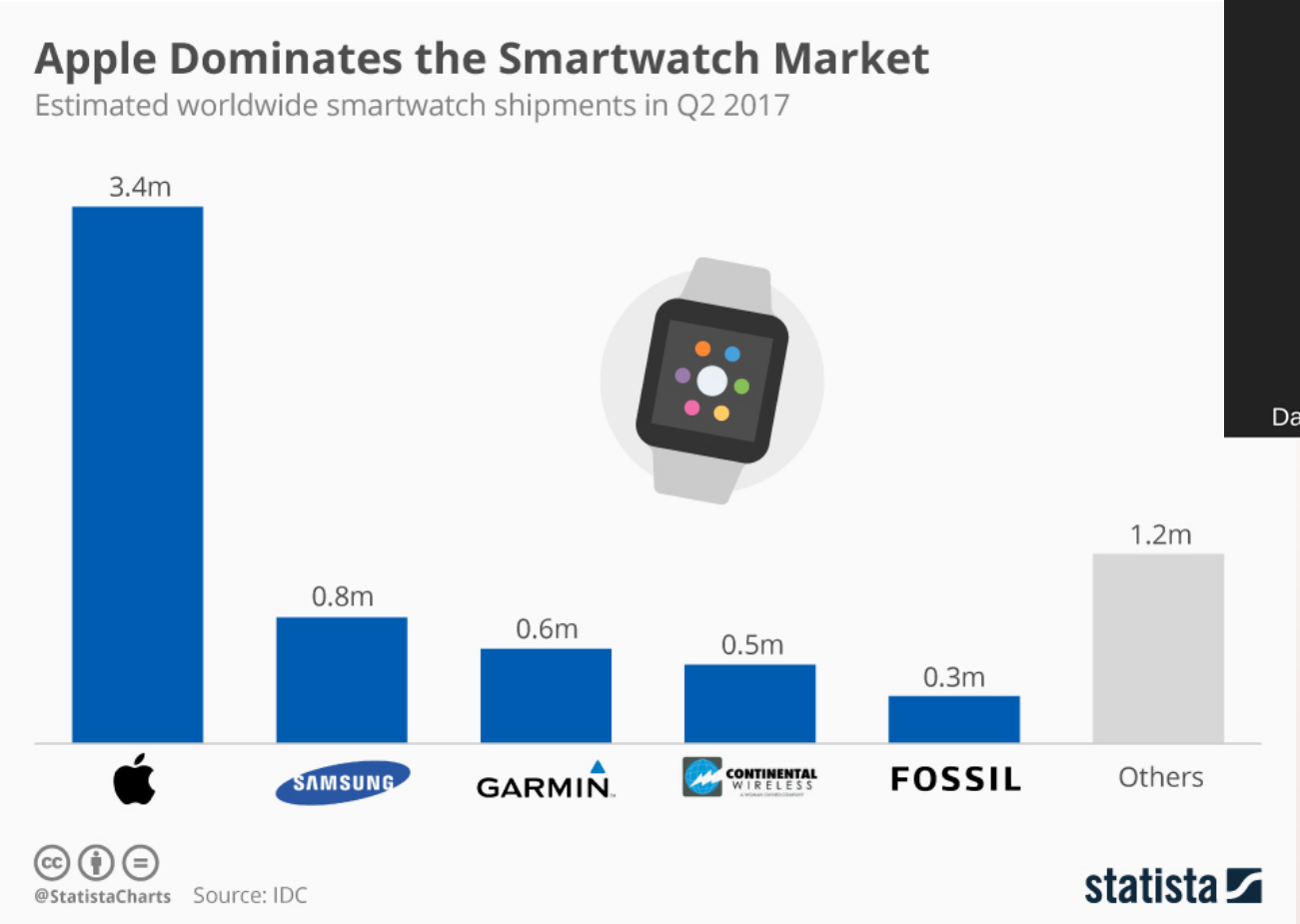
Source: Apple, Pebble, Android Wear Center, VisionMobile estimates



The same old “Android vs. Apple” game

- … but with more players





Developers are conquering… the home

Source: Developer Megatrends

Developers are conquering… the sky

DJI DEVELOPER

Raised $75M to build an industrial UAV developer platform (May 2015)

http://dev.dji.com/

3D Robotics

"Android for drones"

http://dronekit.io/

Airware

An operating system for commercial drones

http://www.airware.com/developers

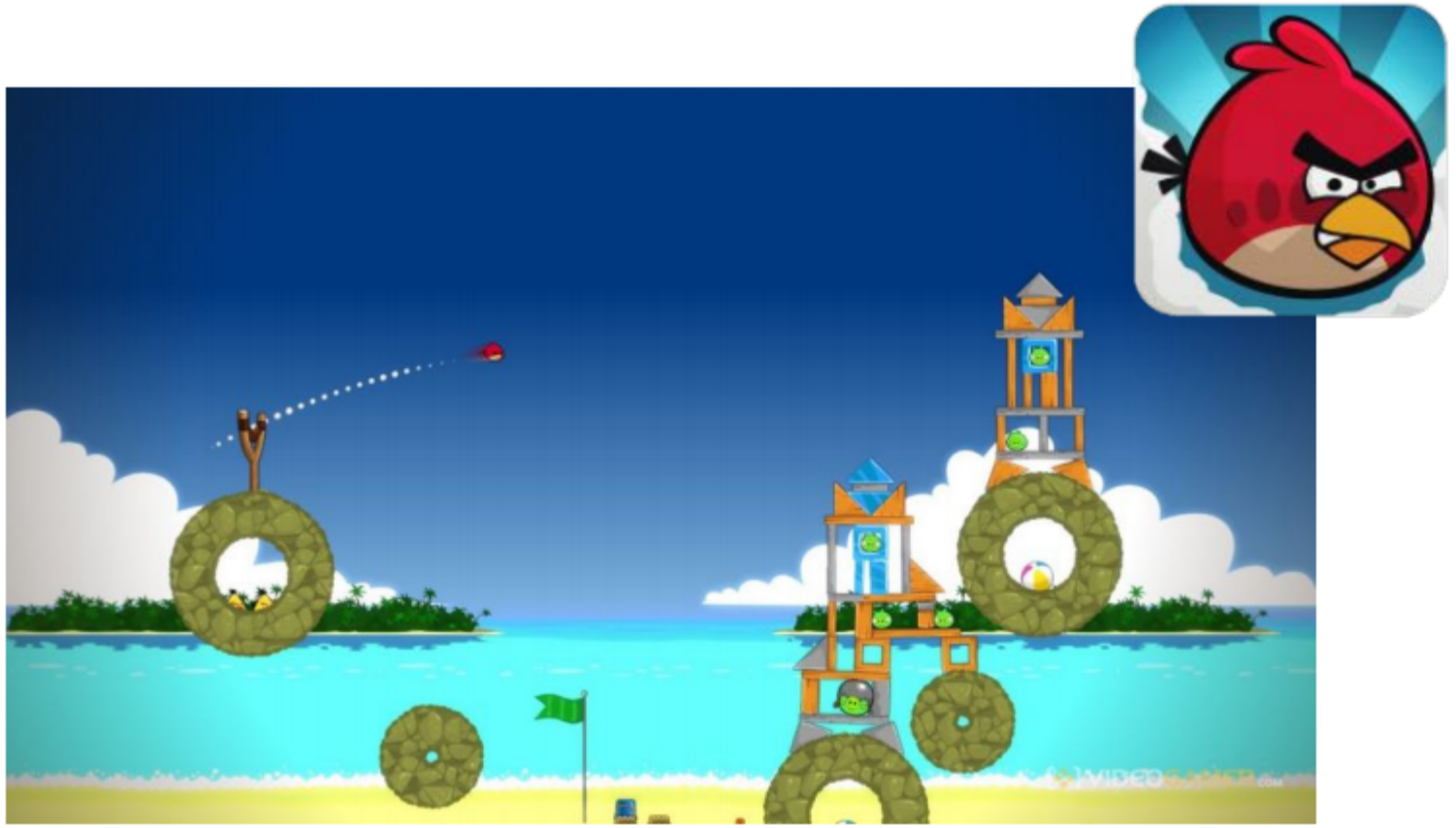
Source: Developer Megatrends

Developers are conquering…

- …cars
- …cities
- …healthcare
- …clothing
- …factories

- …well, everything really!

Yesterday: the traditional app



the value is delivered by the app itself
(even when connected to a backend service)

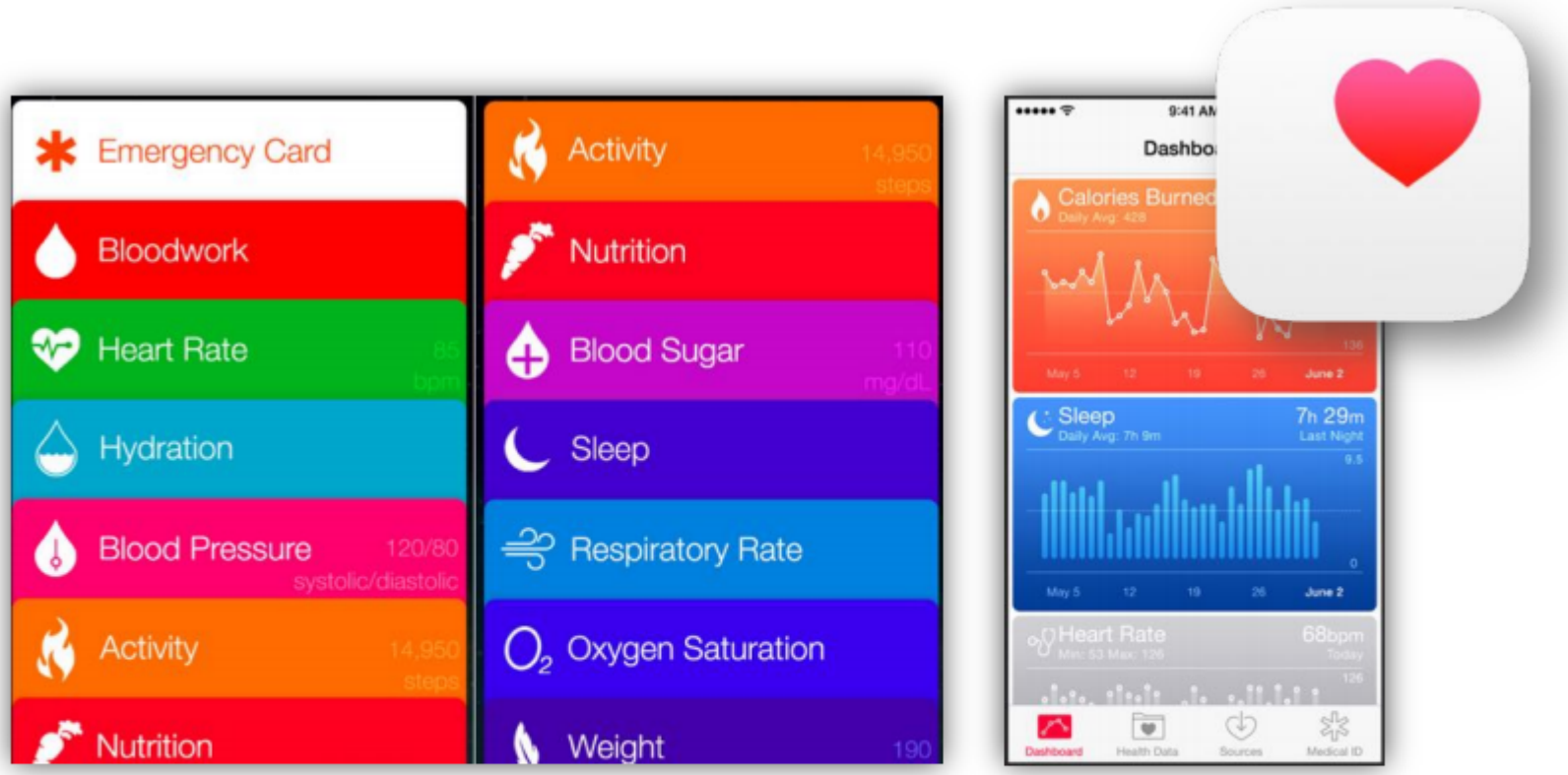
Source: Developer Megatrends

Today: the companion app

the value is delivered by the device
the app is just the remote control

Source: Developer Megatrends

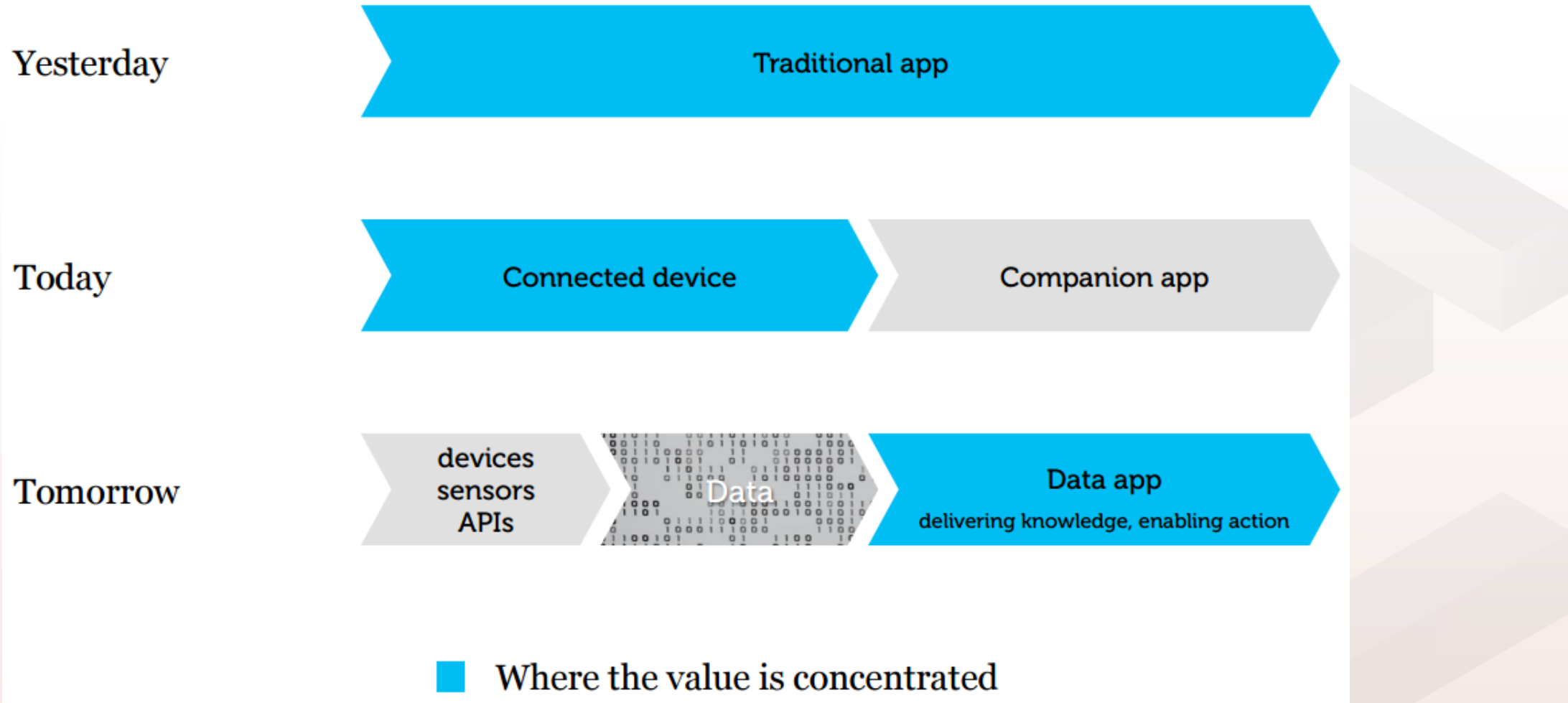
Tomorrow: the data app



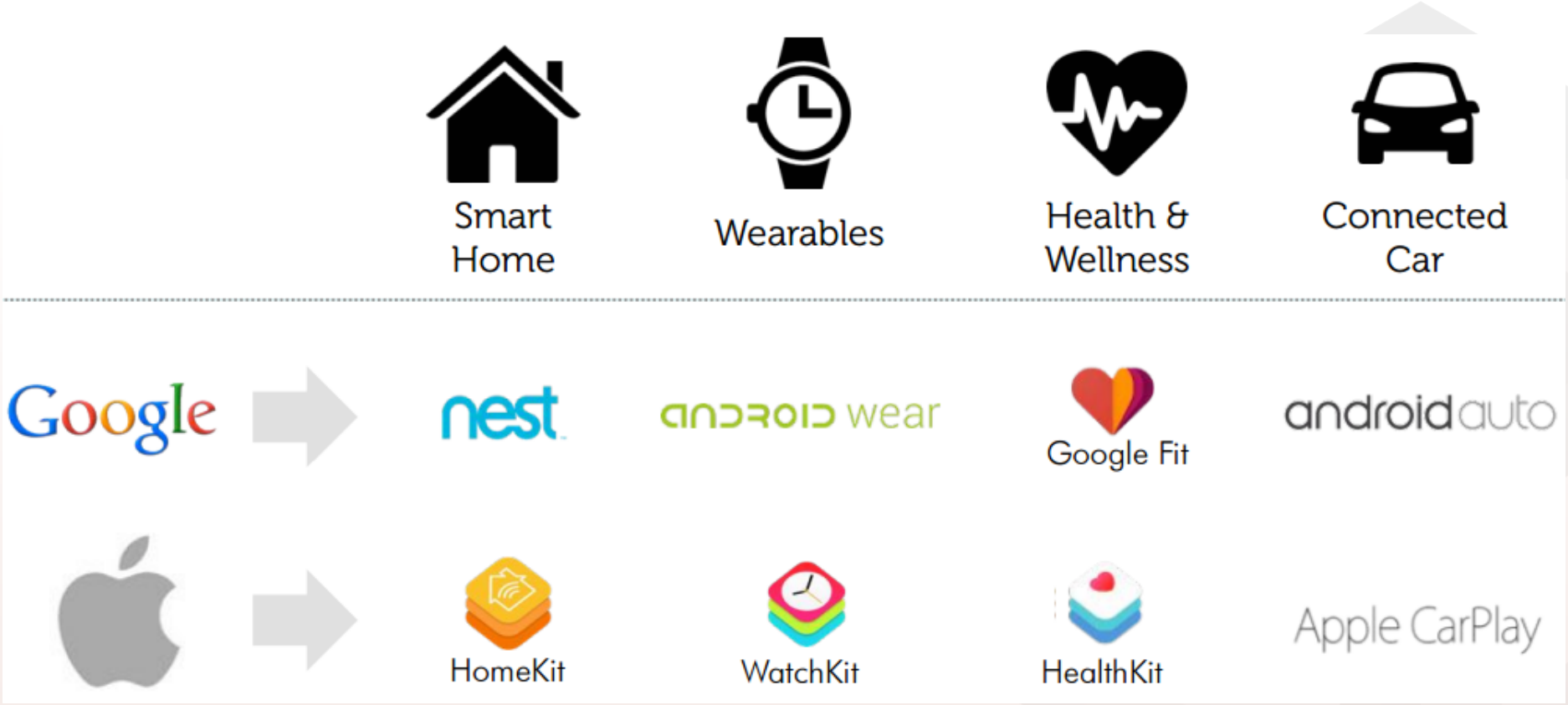
the value comes from combining all kinds of data
the app processes triggers and signals across devices, sensors and APIs

Source: Developer Megatrends

Innovation: making sense of data



The future: Internet of Things



Source: Developer Megatrends

That's why we give you so many devices in this course!

Android development Challenges

- Performance: make your apps **responsive** and **smooth**
  - CPU typically runs slower compared to PCs
  - RAM available to Apps
  - Disk (flash) access is very slow
- Lifecycle
  - Apps must pause/quit often, and restore to give the illusion that they are always running
- User Interface (UI) design
  - Screen sizes
  - Portrait/Landscape
  - Very high DPI - small text may not be readable
  - Touches
- Network access may be slow and (very) intermittent
- Compatibility: run well on older platform versions

Android development Opportunities

- 6.8B mobile phones in use, and very high growth (esp. in Asia)
- Wearables (watches) are an increasing market
- "Wild wild west" of application development
  - Application end-less landscape
  - No dominant 3rd party developers.... yet
  - What will the killer app categories be?
- **You can develop for it today!**

Lectures: Outline of the course

8 weeks of theory/lab sessions (+ 5 weeks for projects)

1. **Android presentation + Java for Android crash-course (today!)**
2. Android basics
3. Activities, intents and application lifecycle
4. User interface
5. Data Management
6. Notifications, Alarms and Sensors
7. Services and background tasks
8. Interacting with external peripherals

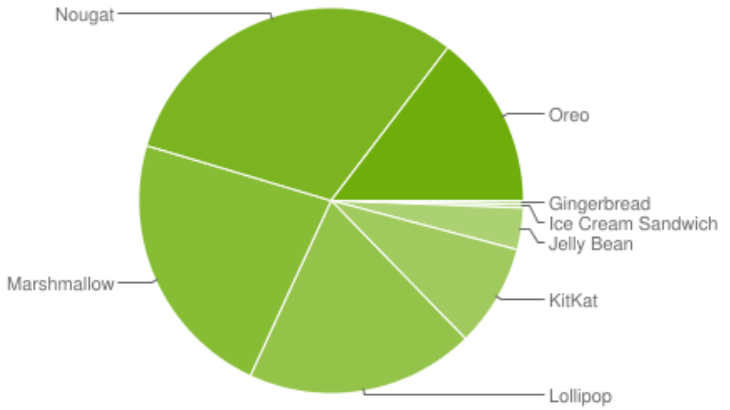
Ref.: https://developers.google.com/training/courses/android-fundamentals

What is Android?

- Android is a **full ecosystem** used on over 80% of all smartphones
- Mobile operating system based on Linux Kernel
- User Interface for touch screens
- Sensors can discover user action and respond
- Multi-platform system:
  - 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 versions

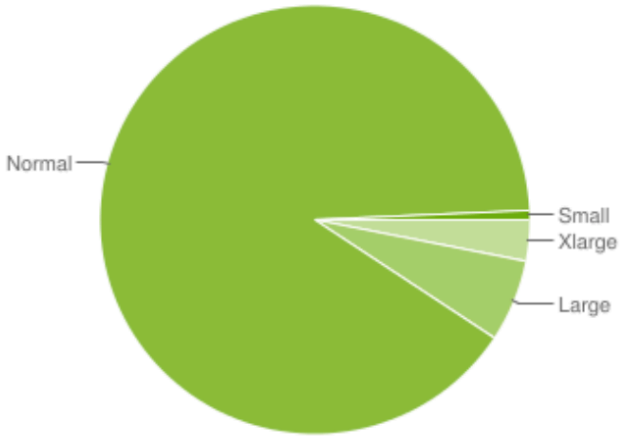
Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.8%
4.3		18	0.5%
4.4	KitKat	19	8.6%
5.0	Lollipop	21	3.8%
5.1		22	15.4%
6.0	Marshmallow	23	22.7%
7.0	Nougat	24	20.3%
7.1		25	10.5%
8.0	Oreo	26	11.4%
8.1		27	3.2%

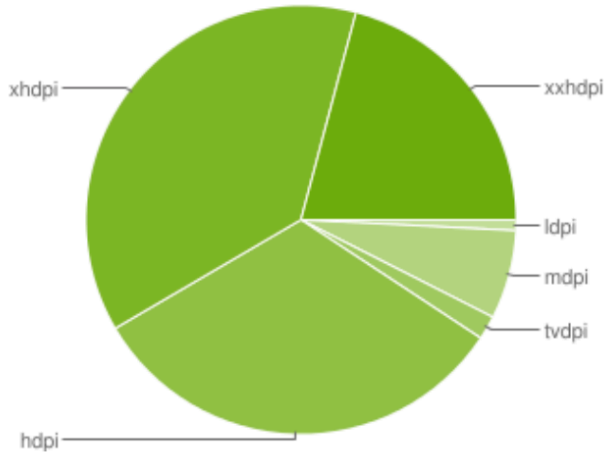


Source: https://developer.android.com/about/dashboards/index.html

Screen sizes and densities

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	0.7%						0.7%
Normal		1.6%	0.2%	31.3%	36.1%	20.8%	90.0%
Large	0.1%	3.0%	1.6%	0.5%	0.9%	0.1%	6.2%
Xlarge		2.1%		0.5%	0.5%		3.1%
Total	0.8%	6.7%	1.8%	32.3%	37.5%	20.9%	





Data collected during a 7-day period ending on September 11, 2017.

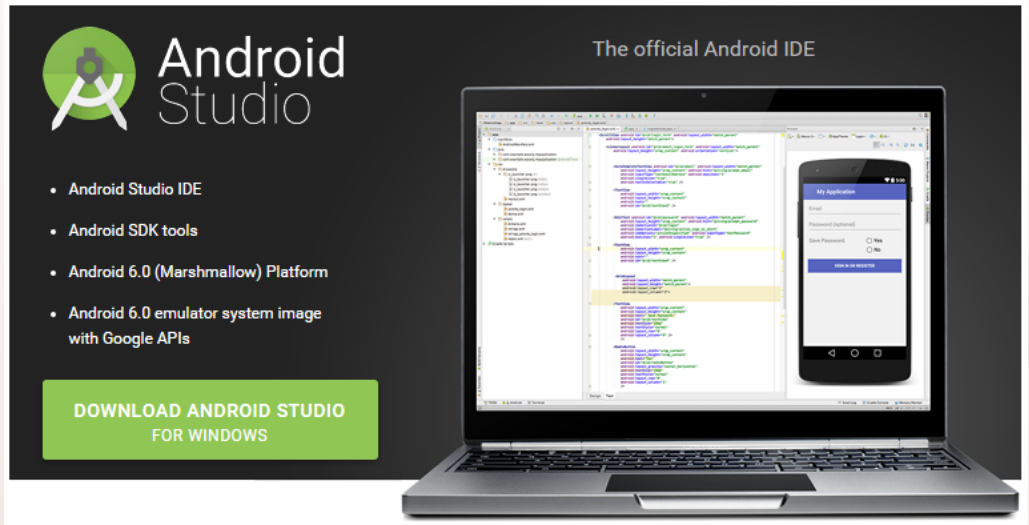
Any screen configurations with less than 0.1% distribution are not shown.

Source: https://developer.android.com/about/dashboards/index.html

Android Software Developer Kit (SDK)

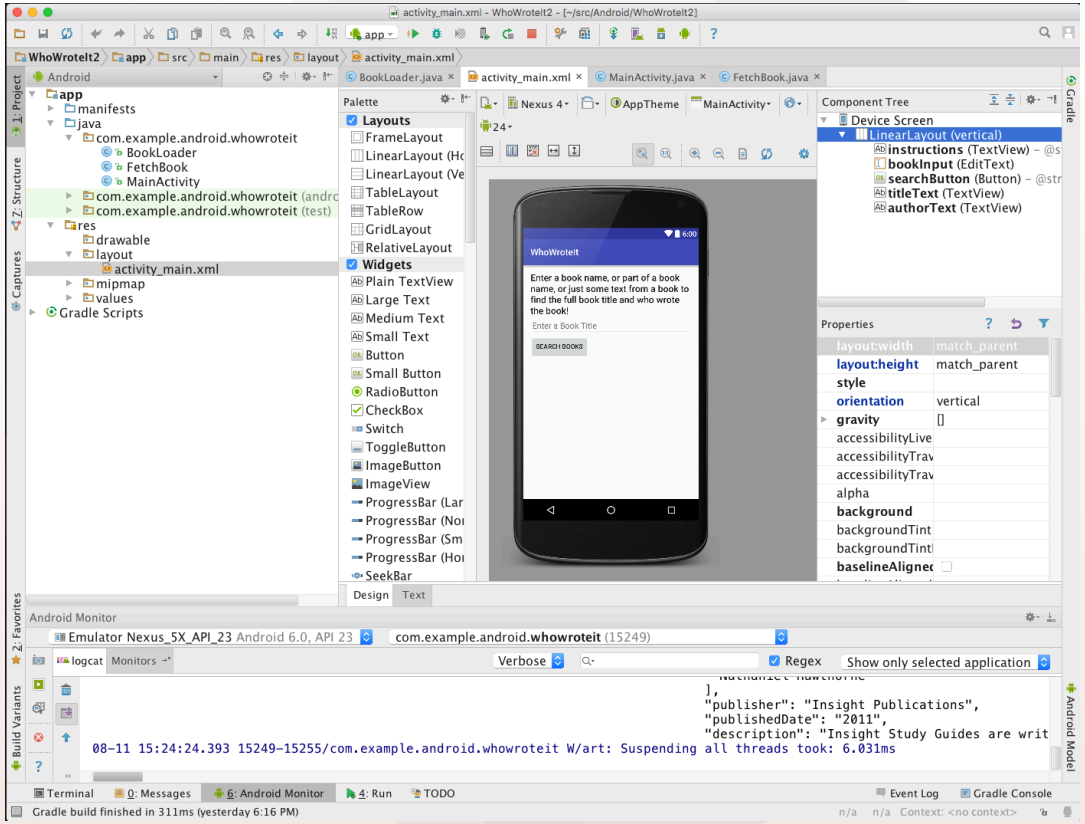
- Developers can download the **Android SDK** for free: http://developer.android.com/sdk/index.html
  - Development tools (debugger, monitors, editors)
  - Libraries (maps, wearables)
  - Virtual devices (emulators)
  - Documentation (http://developers.android.com)
  - Sample code

- **Android itself is an Open Source Project:** http://source.android.com/



Android Studio

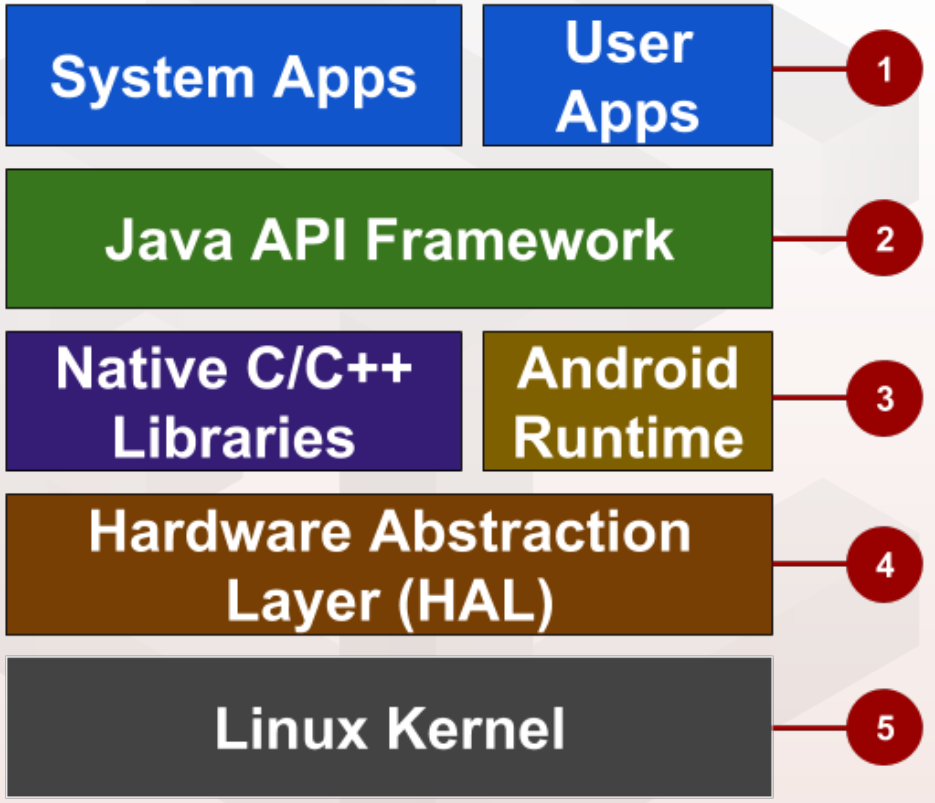
- Official Android IDE
  - Develop, run, debug, test and package apps, monitor tools
  - Virtual devices (emulators)
  - Visual layout editor
- Apps are written in Java
  - ART virtual machine
  - Highly customized and optimized for mobile devices
- Configuration files are in XML
- Apps are packaged into an APK file and installed on the system
- Apps published via the Google Play store
  - Official Google distribution service



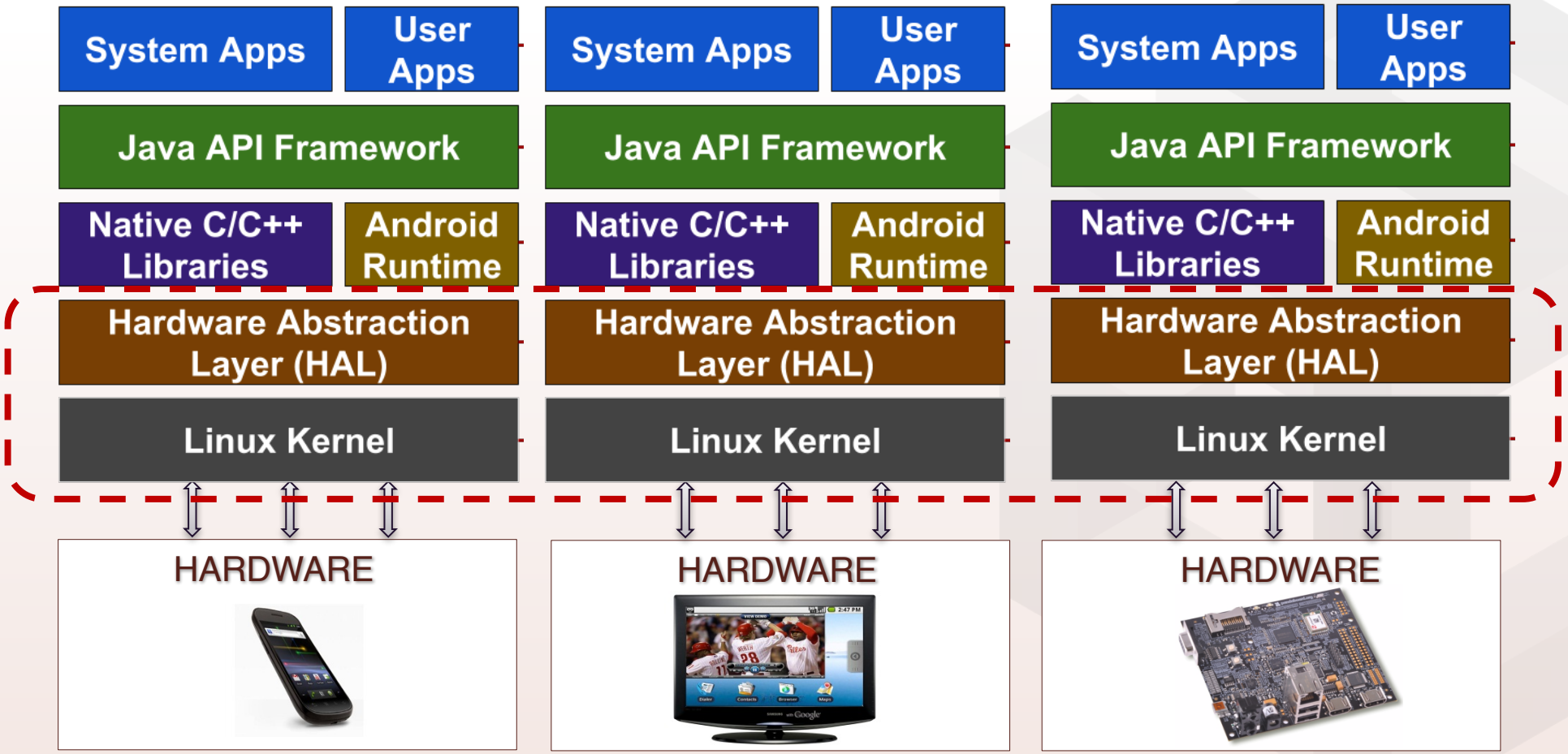
Android platform architecture

The Android stack is composed of layers:

1. System and user apps
2. Android OS API in **Java** framework
   - Feature set of the Android-OS available via Java APIs
3. Android runtime
   - Each app runs its own process with its own instance of the runtime
4. Hardware Abstraction Layer (HAL)
   - Standard interfaces that expose hardware as a library
5. Linux Kernel



Multi Platform Support



Huawei MediaPad T3 10”

- Chipset:
  - Qualcomm Snapdragon 425
    - 4 core Cortex A53 @1.4GHz
  - 3GB RAM
  - Built-in storage: 16GB
- Sensors
  - Ambient light sensor
  - Gyroscope
  - Accelerometer
  - Magnetometer
- Camera: 5MP (frontal 2MP)
- Bluetooth 4.0, A2DP
- WiFi 802.11 b/g/n
- 2G/3G/4G
- LCD Capacitive touchscreen
  - 9.6”, 16million colors, 800x1280px, 157ppi

HTC Nexus 9 (2014)

- Chipset: Nvidia Tegra K1
  - CPU: Dual-core 2.3 GHz Denver, GPU: 192-core Kepler
  - RAM: 2GB RAM
  - Built-in storage: 32 GB
- Sensors:
  - Ambient light sensor
  - Gyroscope
  - Accelerometer
  - Magnetometer
- GPS
- Wi-Fi 802.11 a/b/g/n/ac
- Bluetooth v4.1
- NFC
- Front Camera 8 MP
- LCD capacitive touchscreen
  - 8.9" IPS LCD, QXGA (2048x1536)

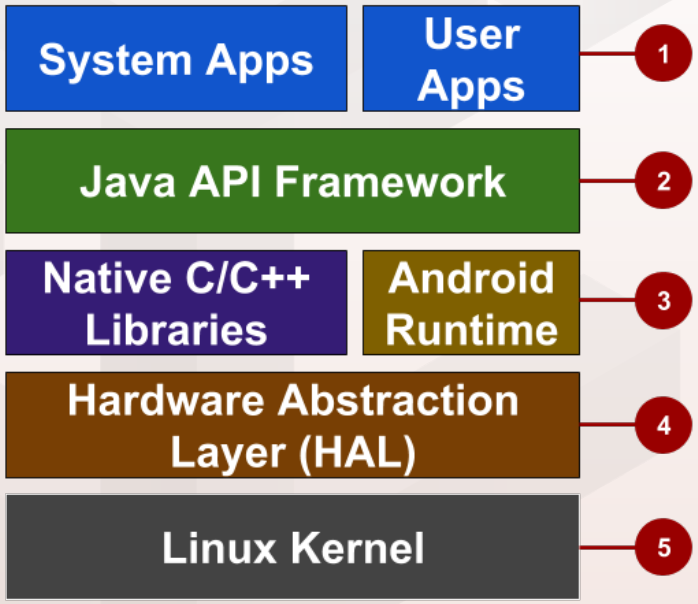
Huawei Watch 2 (2017)

- Chipset:
  - CPU: Qualcomm Snapdragon 2100
  - 768MB RAM and 4GB Flash
- Sensors:
  - 6-axis a+G sensor
  - 3-axis compass
  - Heart Rate Sensor (PPG)
  - Barometer
  - Capacitive sensor
  - Ambient light sensor
- GPS
- WiFi 2.4GHz 802.11b/g/n
- Bluetooth 4.1
- Display:
  - 1.2-inch circular AMOLED display
  - 390x390 pixels w. 326 PPI
- Operating system:
  - Android Wear 2.0

What is an Android app?

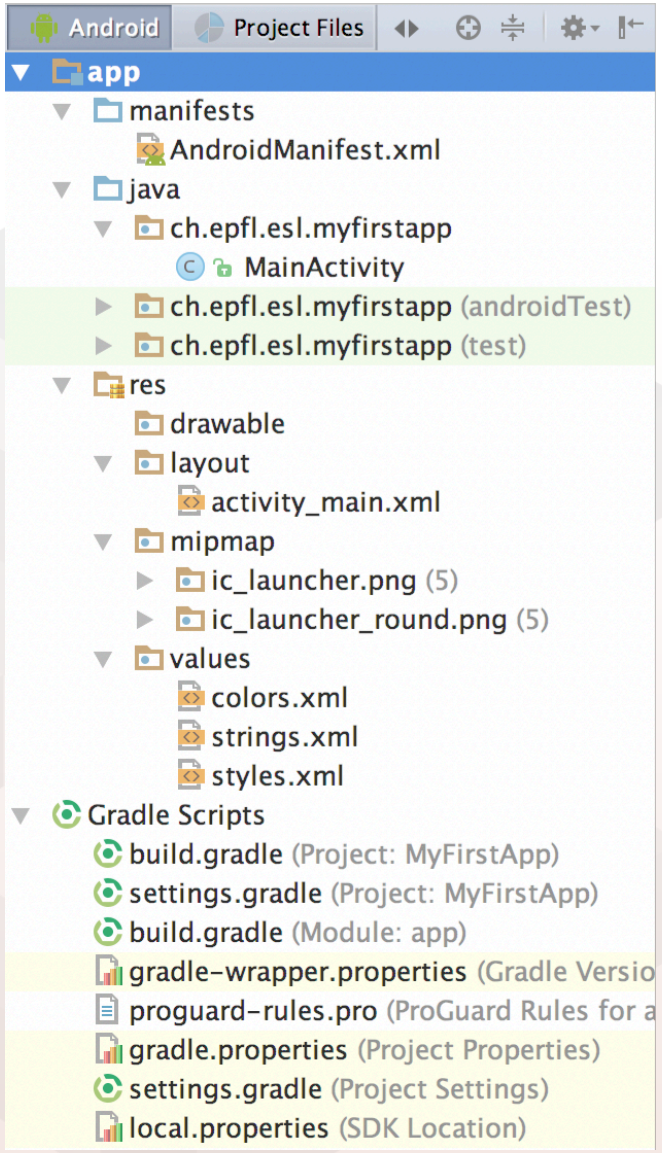
- One or more interactive screens
- Written using Java Programming Language and XML
- Uses the Android Software Development Kit (SDK)
- Uses Android libraries and Android Application Framework (2)
- Executed by Android Runtime Virtual machine (ART) (3)

To program Android devices…
we must learn (a bit of) Java



Android application building blocks

- Manifest:
  - Characteristics of the app and component definition
- Components → ”java” folder
  - Java code (functionality) of your app
  - activities, services, …, and helper classes
- Resources → ”res” folder
  - layouts, images, strings, colors, XML and media files
- Gradle Scripts:
  - Scripts used to build the application
  - APK versions in Gradle config files



We’ll go in depth on Week 2 (next week)

Class outline

- Android presentation:
  - some numbers, motivation
- **Java for Android Crash-course**
- Gentle reminder: project groups
- Lab of Today:
  - Learning Java for Android: some Java exercises
  - Create and build your first app → HelloWorld!

Java and Object-Oriented Programming (OOP)

- Android applications are developed using Java
  - Good news is: Java is easy to learn…! Once you know another language
  - Even better: Android studio will help you a lot!

- What is an OOP language? → One with…
  - **Abstract data types**:
    - Classes (types)
    - Objects (instances of a class)

    } **Objects and classes**
  - **Encapsulation**
    - Information hiding and interfaces

    } **Private & public methods and attributes….**
  - **Inheritance**
  - Polimorfism
  - And others: dynamic binding, dispatching, genericity…

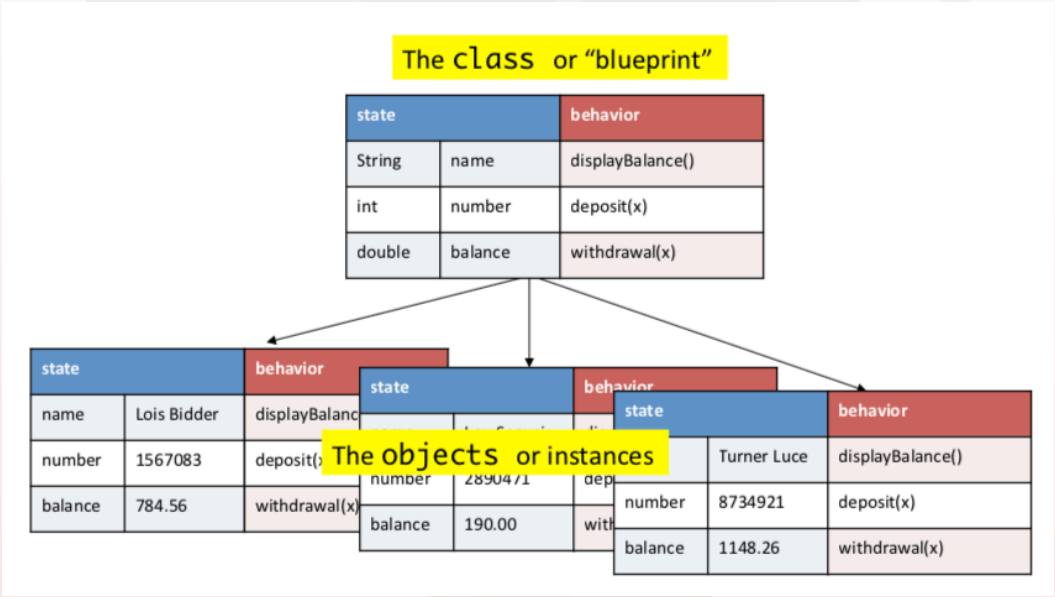
Classes and objects

- A class defines a type:
  - An abstraction represented as a set of features/members
    - Attributes/fields/instance variables
    - Routines/methods/member functions
  - A mold for all its objects
- An object is an instance of that class
  - A chunk of memory shaped as the class dictates
  - Can be attached to a reference of a certain type statically (compile time)
  - … or dynamically (resolved during execution)
- Purpose: abstraction
  - Hide the details about an object in order to reduce complexity and increase efficiency

Examples of classes and objects

- A class … → and its objects…
  - Class: city → Object: Lausanne, Madrid, Boston, Boston…
  - Class: games → Object: tetris, chess
  - Class: vehicles → Object: car, bicycle, motorbike

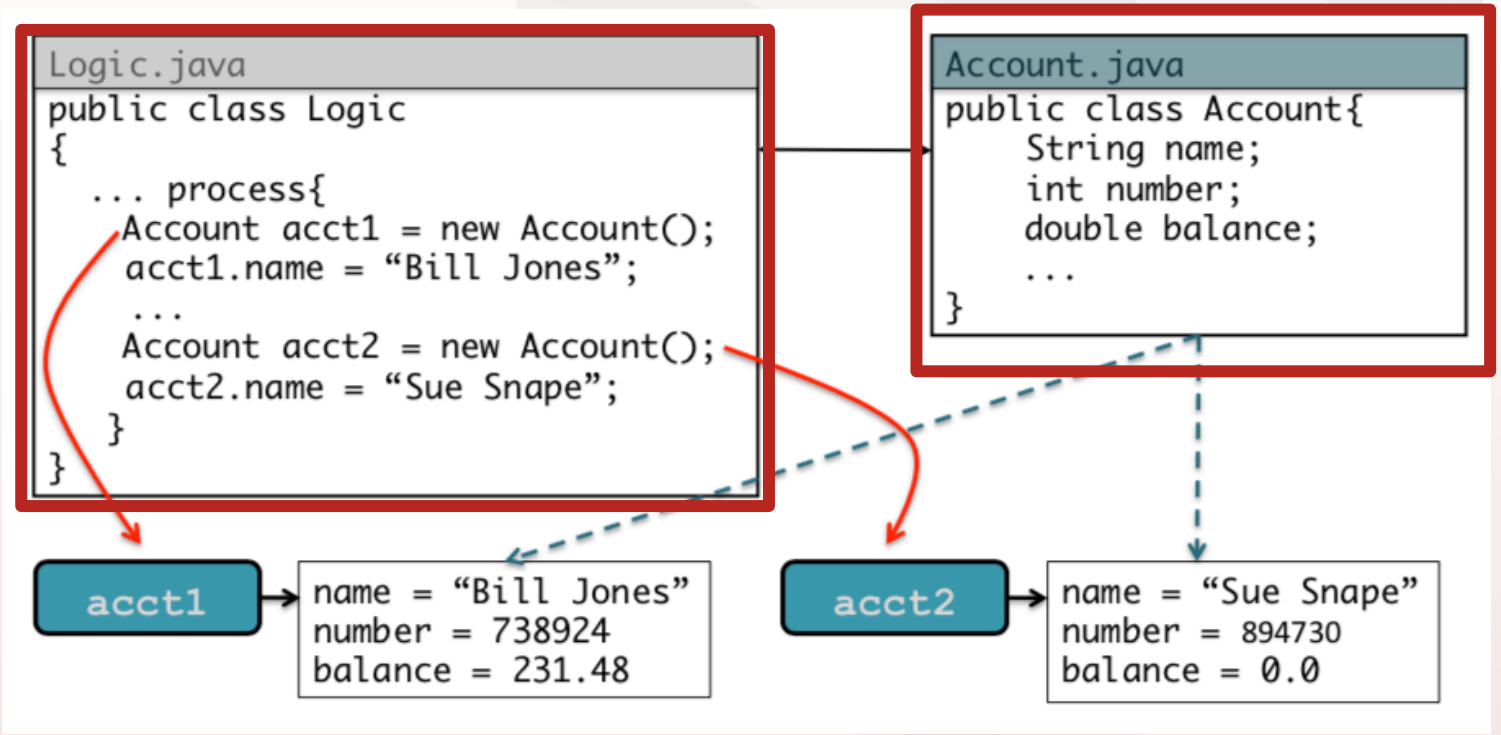
- In today's lab:
  - Class: account
    - Objects: accounts for clients
  - The class contains:
    - Attributes: name of client, account number, balance
    - Methods: to check balance, deposit or withdraw money



A class to create and manage "bank accounts"

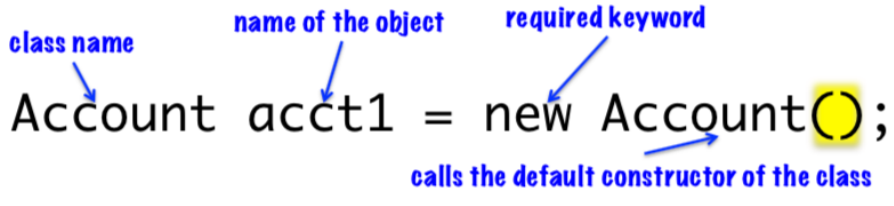
- We create a java file for our "Account.java" class
  - … and fill it with some attributes and methods
  - Typically, one java class per file

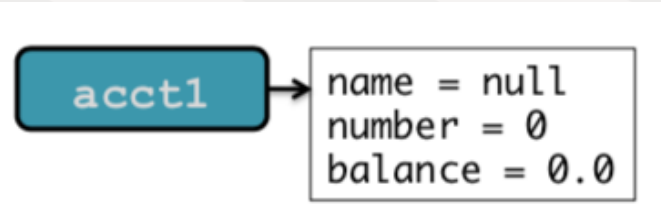
- We create an object "acct1"
  - You can declare a variable to hold that object
    - Account acct1;
  - … but this does not create the object! You have to use "new"
    - Account acct1 = new Account();
  - When doing so, you're calling the constructor



Constructors

- Constructors are used to instantiate objects
- We can do this in two phases:
  - Account acct1;
  - Acct1 = new Account();
- The previous is what we call the "default constructor"
  - Creates an empty instance of an object, initialized to zero/false/null
- Constructors can be "overloaded"
  - We can write several constructors, each with its own (different) parameter list






```
public Account(String newName, int newNumber){
    Account acctA = new Account("Sue Vlaki", 289476);
}

public Account(String newName, int newNumber,
               double initialBalance){
    name = newName;
Account acctB = new Account("Joseph Schmoe", 392784, 187.13);
    balance = initialBalance;
}
```

Encapsulation

- Both attributes and methods can have different visibility/access levels
  - In particular, they can be accessible or inaccesible to other classes
  - The set of visible features also defines the ways to access them!
    - Public: accessible by other objects from the same or different classes
    - Private: accessible only by the object itself

- Two types of methods used to access data:
  - Getters (or accesors)
    - Access the value of a field from outside the object
  - Setters (or mutators)
    - Change the value of a field from outside the object

- A class should always contain getters and setters!

Accessor

```
value = acct1.getName();
```

```
public String getName(){
   return (name);
}
```

Mutator

```
acct1.setName("Rick Rudd");
acct1.deposit(18.91);
```

```
public void setName(String
   nm){
   name = nm;
}
```

```
public void deposit(double
   amt){
   balance = balance + amt;
}
```

The two basic reuse mechanisms

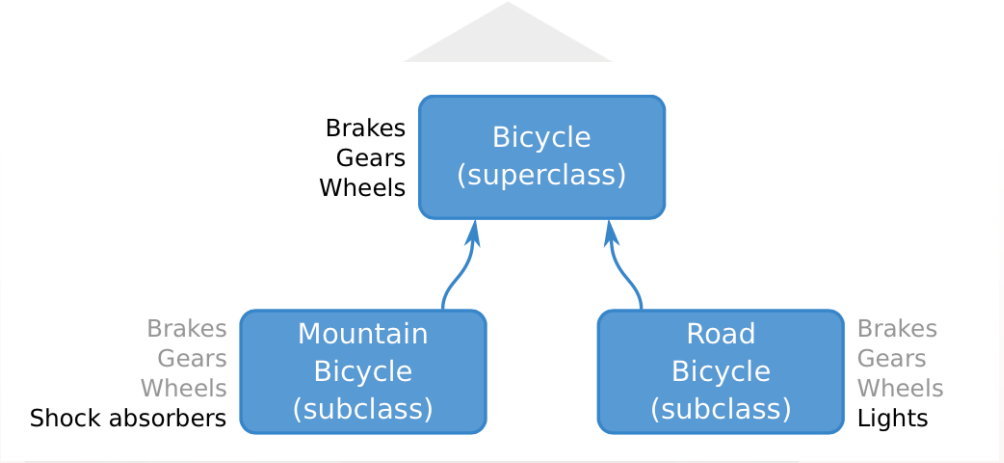
- **Inheritance** relationship
  - A class gets the features of another class by inheriting from it.
    - To create a class that inherits from another we "extend" from the parent class
  - The derived class can also:
    - Introduce new features
    - Redefine features of the parent class
  - *Note for advanced OOP users:*
    - *In Java we only hace single inheritance! (unlike C++!)*

```
public class AccountClient extends Account {

    private boolean blockCard; // we cannot cash out more

    public AccountClient(String newName, int newNumber,
                         double initialBalance, boolean limit) {
        super(newName, newNumber, initialBalance);
        this.blockCard = limit;
    }
```

- Client relationship:
  - A class gets to use the features of another class by declaring an attribute of that other class' type → We create an object of a class from the "Main"

An example of inheritance

- In today's lab, we will see an example of inheritance
  - Some synomyms and naming issues
    - Superclass = Parent class
    - Subclass = Child class
- Careful! Constructors are not inherited!
  - We call the constructor of the parent class from the children class using "**super**"
- We can also override (rewrite) a method from a parent class!
  - We use "**@Override**" before the method



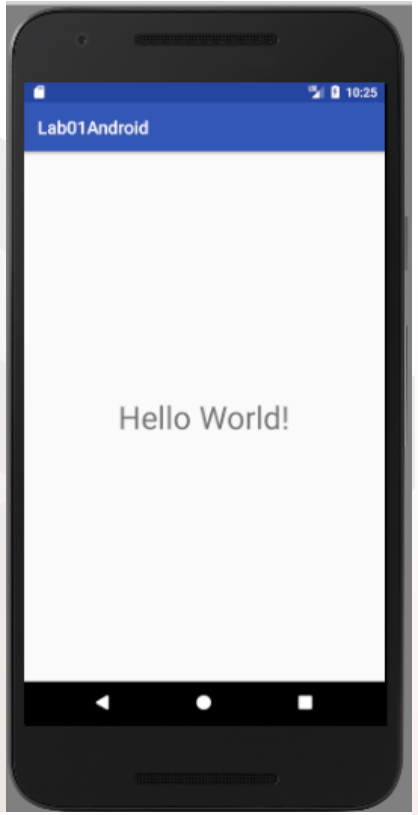

```
public class AccountClient extends Account {

    private boolean blockCard; // we cannot cash out more

    public AccountClient(String newName, int newNumber,
                         double initialBalance, boolean limit) {
        super(newName, newNumber, initialBalance);
        this.blockCard = limit;
    }
```

Java for Android

- Java for Android, extensively uses the concepts of class, encapsulation and inheritance.
- All classes eventually extend from the *Java Object* class
  - As we will see on the next lecture

- Today, in Lab01b, we will create for our HelloWorld app our first “Activity”

- **Activity** is a single screen with a user interface
  - For the HelloWorld program, we will create a “MainActivity” class that **inherits** from an “Activity”
  - We will **@Override** the onCreate function
  - We will call the parent constructor using “**super**”




```
@Override
    protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    TextView view = (TextView) findViewById(R.id.my_text);
    view.setText(R.string.my_custom_message);
}
```

Class outline

- Android presentation:
  - some numbers, motivation
- Java for Android Crash-course
- **Gentle reminder: project groups**
- Lab of Today:
  - Learning Java for Android: some Java exercises
  - Create and build your first app → HelloWorld!

Projects: reminder

- For the labs and projects, please create **groups of 3 students.**
- Request a project before **November 21st**
  - 1 member of the group sends e-mail to Dr. Zapater with:
    - Names of group members
    - A **prioritized list (at least 3 choices)** of the projects you want to do.
  - Assignment will be based on order of reception of e-mail
  - If you don't pick a project, you'll be assigned one

- 18 students already have requested group and project
- New projects on the slides on Moodle!
- By next week → first assignment of students to projects
  - Drone project 1.A/1.B/1.C already **full.**
  - List will be shared via Moodle with further details

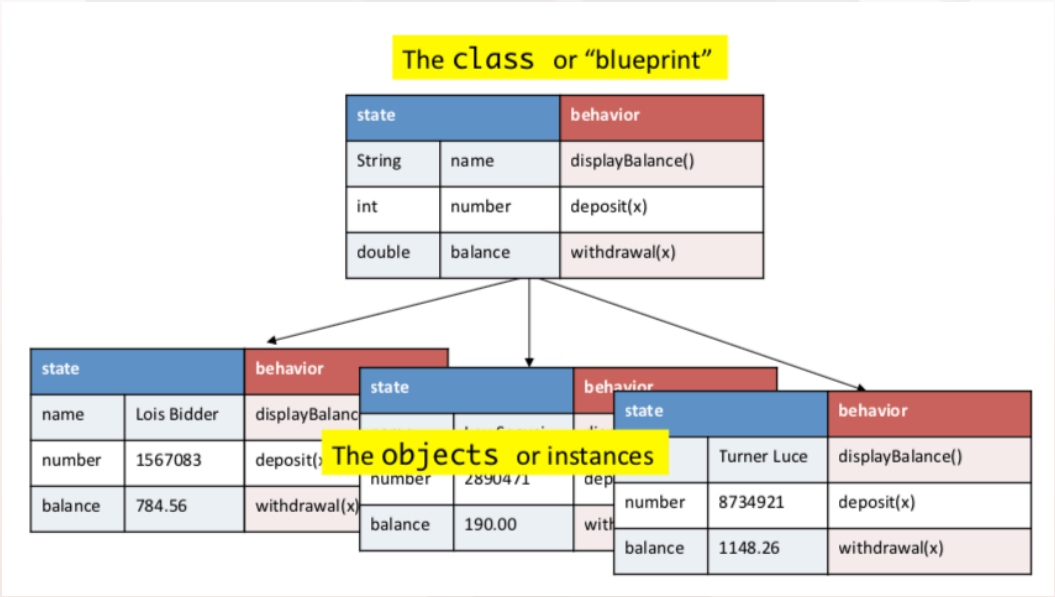
Today's Lab – Part A "Java basics"

1. Importing an Android project
2. Creating your first class and object with the "Accounts" example
3. Basics of encapsulation
4. Constructor basics
5. Inheriting from classes

Download the lab guide from Moodle.

Already know Java?

Go directly to Part B!



Today's Lab – Part B "Hello world!"

1. Creating your first Android project
2. Fundamental notions
3. Running your app (on real device and emulator)
4. Basics of Android Studio
5. "Hello World!" project on the Phone/Tablet
6. "Hello World!" project on the Watch

Download the lab guide from Moodle.