Mobile Device Application Principles

- Programming Android Applications using Java!

Installing the development environment

- We'll be using the Android Studio which can be downloaded here:

  https://developer.android.com/sdk/index.html

- Start the download now, but please wait until after the course intro for installing it.

Topics

- Introduction to Programming for Android
  - Android Studio
  - Basic UI elements
  - Intro to the framework
  - Hooking up the UI to code

- More GUI
  - Spinners
  - CheckBoxes, RadioButtons, etc.
  - ListViews and Adapters

Topics

- More about the framework
  - Activity lifecycle
  - Multiple Activities
  - Intents, Bundles
  - Services, IntentServices
  - Notifications
  - Location Based Services
- Persistence in Android
  - Shared Preferences
  - File I/O
  - Introduction to SQLite databases

Topics

- Internet/Cloud connectivity
  - ConnectionManager
  - HTTP/s connectivity
  - JSON parsing
  - RESTful method calls (time permitting)

Course Evaluation (also on class plan)

Midterm Exam (Week 7)	25%
Final Exam (Week 14)	30%
Assignments	30%
Project	15%

Note:
You must average at least 50% on the exams to pass the course!

My Expectations

You are expected to be prepared, to be an active learner, to be respectful and to be vigilant regarding breaches of academic honesty.

- Being prepared
  - Taking responsibility for setting up your laptop and keeping it in good working order
  - Striving to arrive to class on time
  - Submitting assignments by their due date

My Expectations

- Active Learner
  - Pay attention during class time (you are paying for this!)
  - Completing all in-class activities
  - Asking questions whenever you don't understand.
    If you are having doubts, odds are others are as well
  - Reviewing material prior to class

- Respectful
  - Avoiding disruptions such as games, videos, texting, chatting, etc during class (breaks OK)
  - Drinks and light snacks are OK, but please no eating meals during class time (subway sandwich, burgers & fries, etc.)

Academic Honesty (we all want to help friends!)

- Good ways to help
  - Talking things over to help each other understand concepts
  - Showing someone where to find the info they need
  - Testing another student's program to look for bugs
  - Sitting with someone to advise them while they write or debug a program (but not dictating code line-by-line for them to write)
- Bad ways to help (may result in serious penalties)
  - Writing part of somebody's code for them
  - Giving someone your program (or parts of it) so they can use it as guide. Includes posting your assignment in a public forum
  - For group assignments, letting someone join your group and then doing all the work for them.

Academic Honesty (We all need a bit more help!)

- Good use of resources
  - Using all posted material on SLATE
  - Going to online forums (i.e. StackOverflow) to ask questions or look at answers for some coding issues you may have
  - Taking snippets of code and including it in your work as long as attribution is given back to the site (url) and it does not represent a significant portion of the complete submission

- Bad use of resources (may result in serious penalties)
  - Requesting/using a complete solution to an assignment question in an online forum
  - Using snippets of code without proper attribution
  - Using online resources when explicitly told not to (i.e. quiz, exam)
  - Using online code that represents a significant portion of the completed work (even if it is properly attributed)

What you can expect from me

- To arrive to class on time and well prepared
- To teach you the technology and underlying concepts as best I can
- To post slides before class starts so that you can easily follow during class
- To be sensitive to the needs of the class (I need help from you on this; I try, but I can’t always read minds!)
- To treat everyone in a professional, courteous and respectful manner
- To get back with any information I do not readily have the answers to (android has new releases almost weekly!)

- I’ll try to make it fun ☺

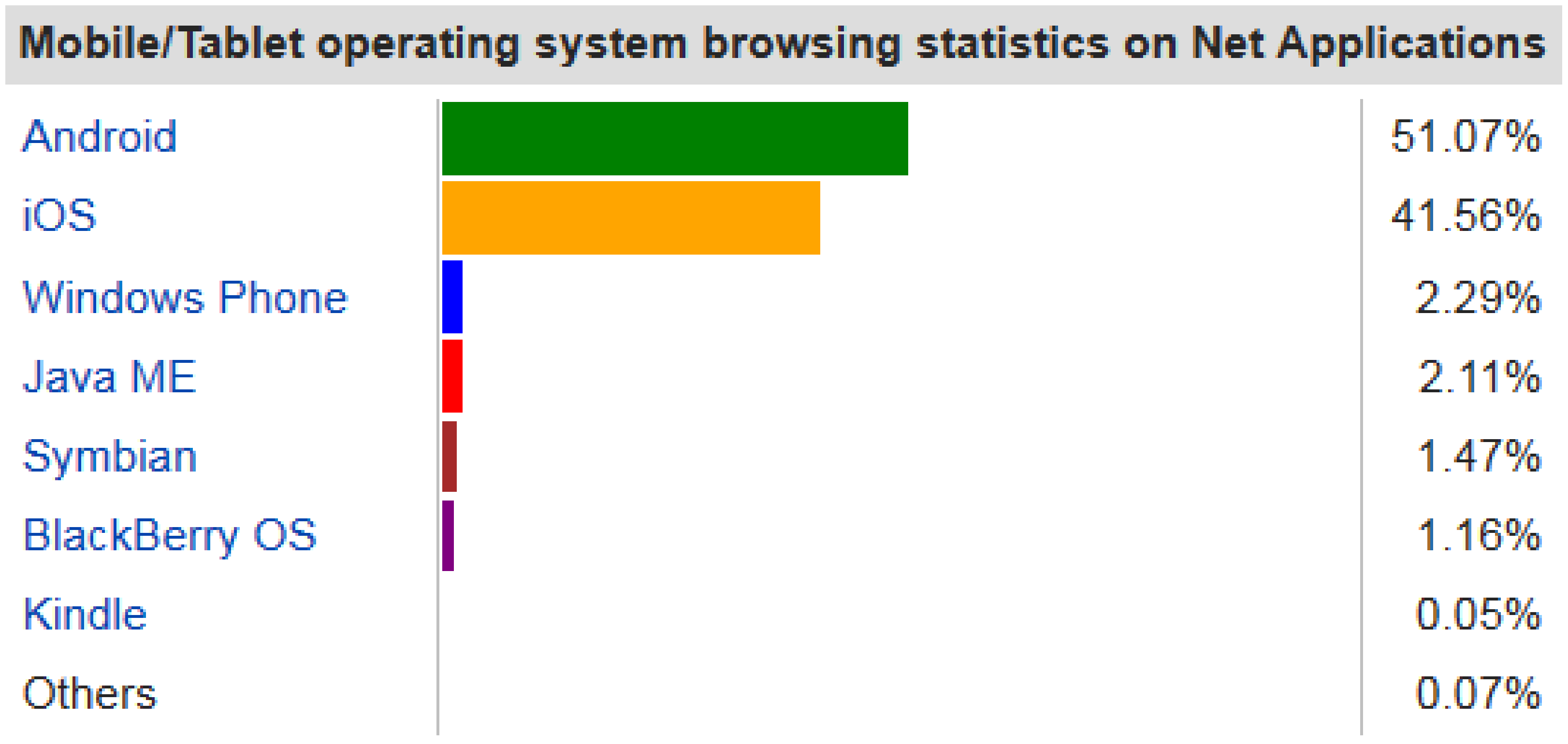
Let's get to know each other

- Groups of 2 or 3
- Interview each other and find out things such as
  - Studying and working
  - Your experiences with programming
  - The "loves" and "hates" of programming
  - Your expectations for this course
- Come up front of the class and do your first presentation
- Each person presents the other

Who am I? Paul Bonenfant

- BA, MSc in mathematics
- Employment roles (partial list)
  - Pneumatic/hydraulic tool repairman
  - Defence Scientist (shooting at T72 tanks ☺)
  - Programmer for robotic applications
  - Lead Programmer for business applications
  - Technical Project Manager and Lead Technical Architect (projects over $1M)
  - Manager, Software Development (10+ employees)
  - Freelance consultant
  - Professor ☺

Why learn how to program for Android?


Mobile/Tablet operating system browsing statistics on Net Applications

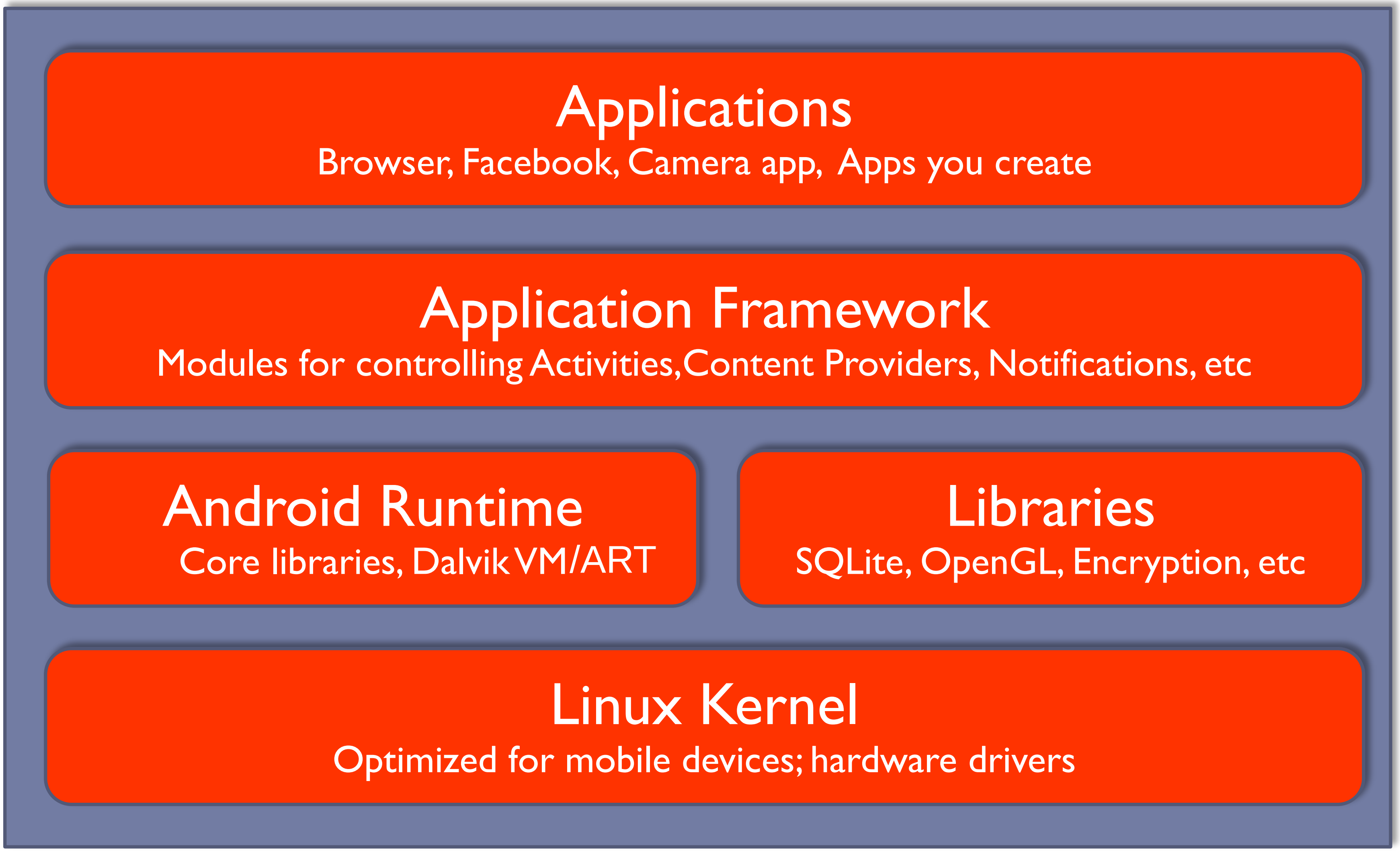
Mobile OS market share as of July 2015 Net Applications[43]

* Taken from Wikipedia as of Fall 2015

Why learn how to program for Android?

- We can use Java to create very powerful and compelling apps.

Android's Software Stack



Applications

Browser, Facebook, Camera app, Apps you create

Application Framework

Modules for controlling Activities,Content Providers, Notifications, etc

Android Runtime

Core libraries, Dalvik VM/ART

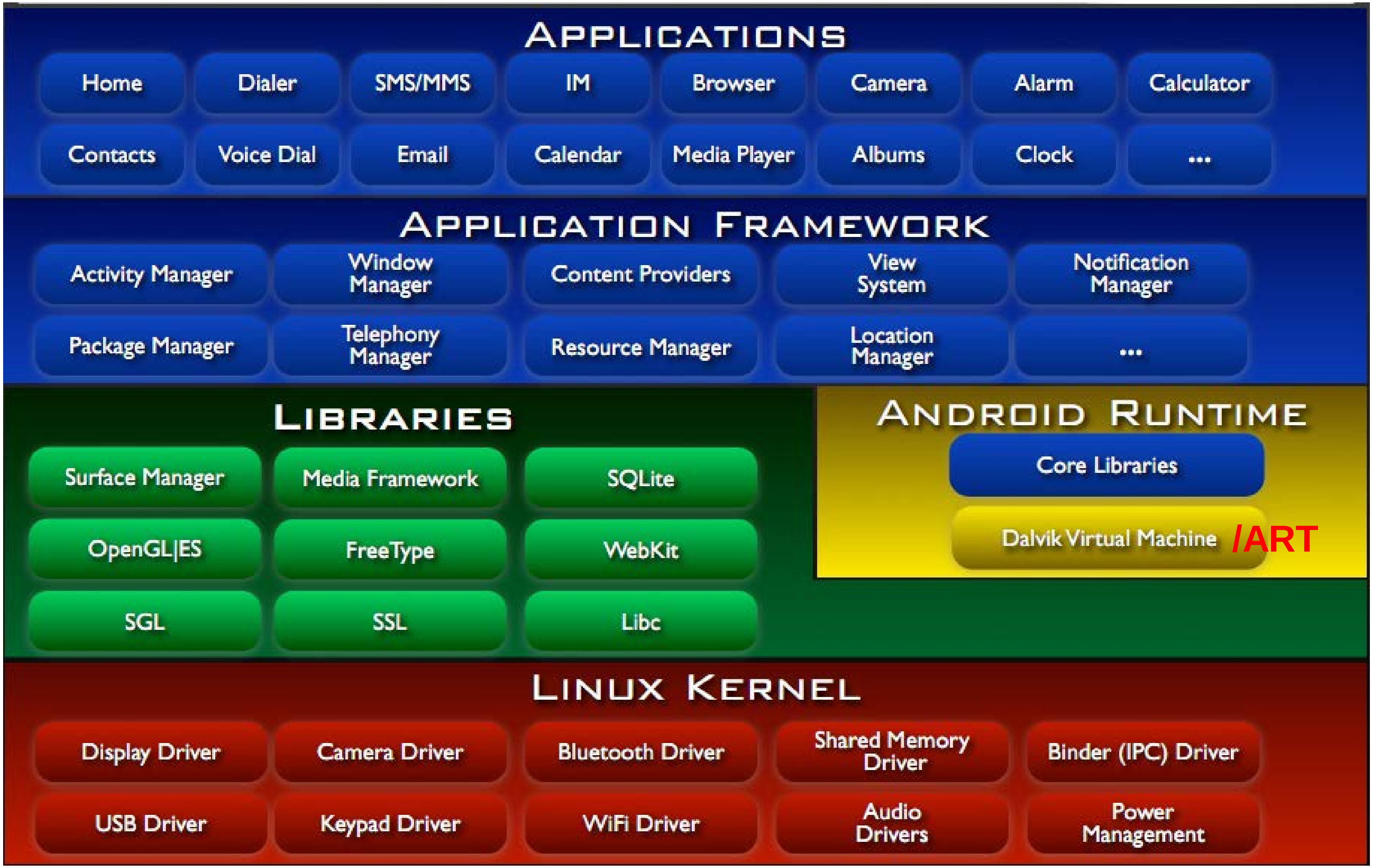
Libraries

SQLite, OpenGL, Encryption, etc

Linux Kernel

Optimized for mobile devices; hardware drivers

Android’s Software Stack



A word on Dalvik VM and ART

- Dalvik VM
  - The Android Runtime since the first release up to Android 4.4 (Kit Kat)
  - Uses just-in-time compilation (faster installs)
- Android Runtime (ART)
  - An optional runtime introduced in Android 4.4 (Kit Kat); on all devices from 5.0 (Lollipop) onwards
  - Uses ahead-of-time compilation (faster execution)
  - Improvements in garbage collection performance and scheduling
  - Better debugging support and diagnostic information
- See here and here for more information

A word on Dalvik VM and ART

- What does this mean for me??
  - All of the applications we will be creating in this class will run equally well in both runtimes.
  - Only have to worry about issues in more advanced settings
    - Explicitly calling garbage collection to manage potential memory issues
    - Using JNI to call C/C++ code
  - You may not get the same debug diagnostic information in the IDE if you are using a device/emulator pre-4.4
  - You should know the basics (these last two slides) for any quiz/test!

Hybrid Android Apps

- Hybrid application development using HTML 5 and CSS with 3rd party tools
  - Titanium, PhoneGap, etc.
- Benefits
  - Can create compelling apps quickly
  - Can have the same code-base for different devices
- Drawbacks
  - Cannot access all underlying hardware capabilities.
  - Typically not the best solution where performance could be an issue.

Android NDK

- Using the Android NDK (Native Development Kit)
  - Program mobile apps using C, C++
- Benefits
  - Lowest level access to device hardware/memory
  - Can result in the most efficient implementation of time-consuming algorithms
- Drawbacks
  - More difficult to program; must handle memory management
  - Development time is typically much longer but you write only intensive parts of your application using it

Android SDK

- This what we'll be using
  - Programming mobile apps using Java
- Benefits
  - Programming in a high-level language
  - Have access to all hardware and threading capabilities
  - Sweet spot between hybrid and NDK apps
- Drawbacks
  - The code runs only on Android devices

Developing for Mobile Devices

- Exercise 1
  - to be done in groups now, with discussion following:

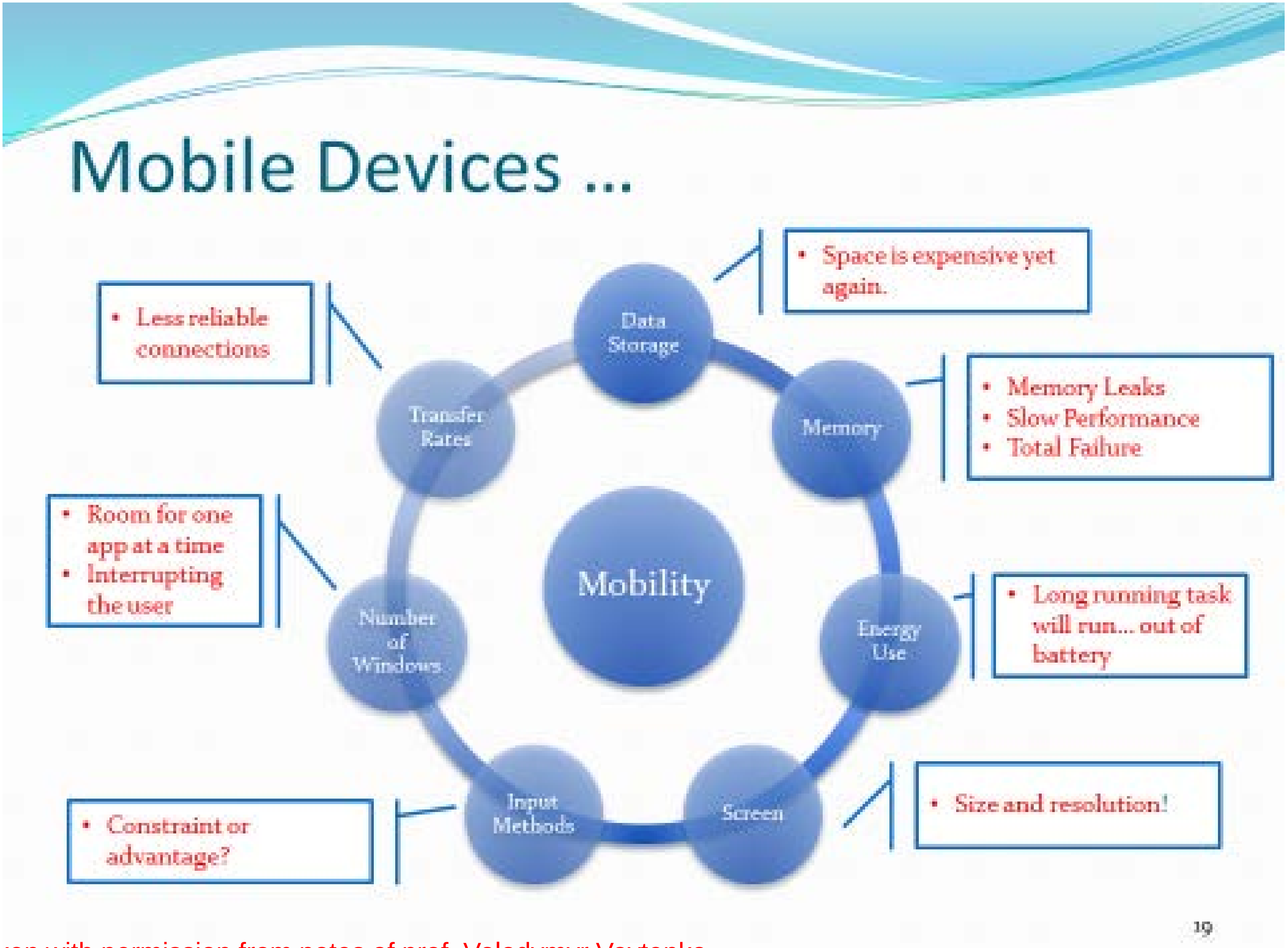
1. Identify and describe **at least three** differences between mobile applications and typical desktop applications.

2. Identify and describe **at least three user's needs and expectations** of mobile applications.

* Taken with permission from notes of prof. Volodymyr Voytenko

Some possible answers to Exercise 1.1:

1. Mobile devices are "MOBILE"! Desktops are not!
2. Mobile devices have smaller storage capacity than desktops.
3. Mobile devices have lower memory capacity than desktops.
4. Limited power on mobile devices; desktops can run continuously without worrying about power and can run power-intensive apps.
5. Smaller and varying screen size on mobile apps.
6. Mobile apps – different input methods
7. Desktops can display multiple app screens at the same time; mobile apps display only a single app screen at a time. (although this is changing)
8. Mobile devices use wireless - slower data transfer
9. Others…



Mobile Devices …

Mobility

Data Storage

- Space is expensive yet again.

Memory

- Memory Leaks
- Slow Performance
- Total Failure

Energy Use

- Long running task will run… out of battery

Screen

- Size and resolution!

Input Methods

- Constraint or advantage?

Number of Windows

- Room for one app at a time
- Interrupting the user

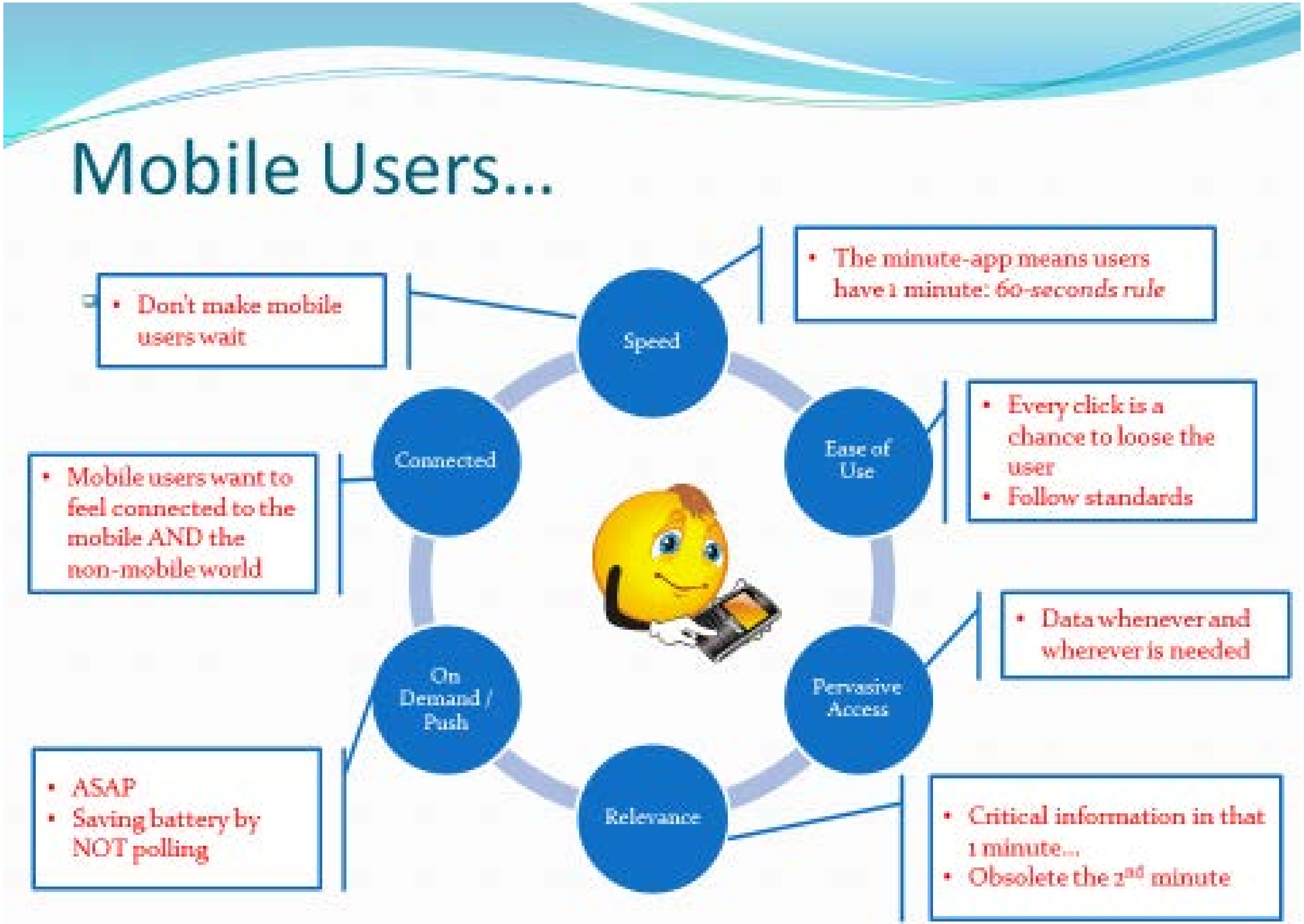
Transfer Rates

- Less reliable connections

* Taken with permission from notes of prof. Volodymyr Voytenko

Some possible answers to Exercise 1.2:

1. Users want time-sensitive information quickly (10 second rule). Do not make user wait!!!!
2. Users want only the most relevant information – only the options/info most relevant to the current task.
3. User wants data whenever and wherever is needed
4. Simple and easy-to-use, consistent.
5. Others…



Mobile Users…

- Don't make mobile users wait

Speed

- The minute-app means users have 1 minute: 60-seconds rule

Ease of Use

- Every click is a chance to loose the user
- Follow standards

Pervasive Access

- Data whenever and wherever is needed

Relevance

- Critical information in that 1 minute…
- Obsolete the 2nd minute

On Demand / Push

- ASAP
- Saving battery by NOT polling

Connected

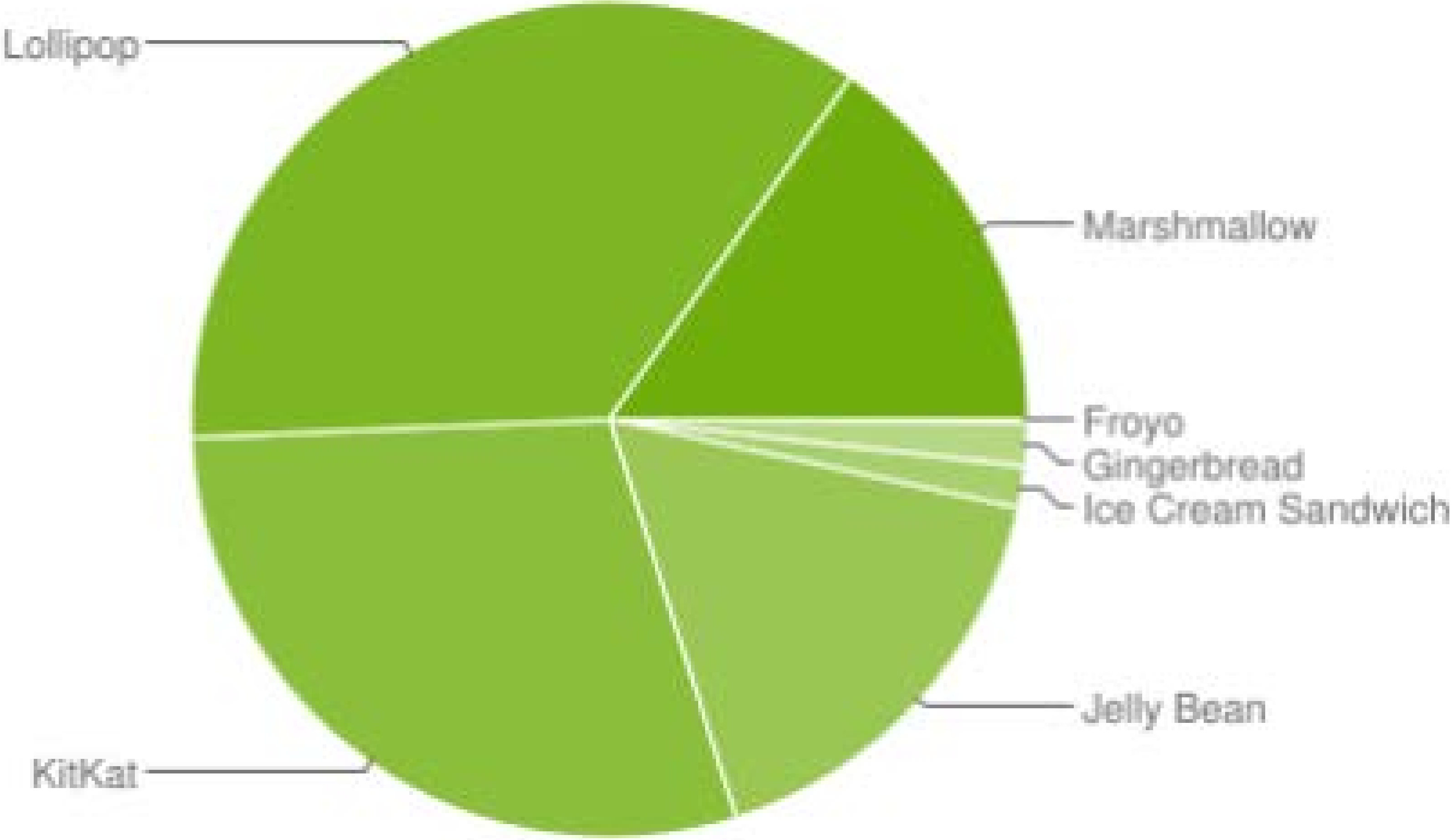
- Mobile users want to feel connected to the mobile AND the non-mobile world

* Taken with permission from notes of prof. Volodymyr Voytenko

Different Flavors of Android

Name	Android Version	API
Cupcake (April 2009)	1.5	Level 3
Donut	1.6	Level 4
Éclair	2.0, 2.01, 2.1	Levels 5, 6, 7
Froyo	2.2 – 2.2.3	Level 8
Gingerbread	2.3 – 2.3.2, 2.3.3 – 2.3.7	Levels 9, 10
Honeycomb	3.0, 3.1, 3.2	Levels 11, 12, 13
Ice Cream Sandwich	4.0 – 4.0.2, 4.0.3 – 4.0.4	Levels 14, 15
Jelly Bean	4.1, 4.2, 4.3	Levels 16, 17, 18
KitKat	4.4	Level 19
Lollipop	5.0, 5.1-5.1.1	Level 21, 22
Marshmallow	6.0 – 6.01	Level 23
Nougat	7.0	Level 24

Distribution of Android Versions



* Taken from developer.android.com (As of Aug. 1, 2016)