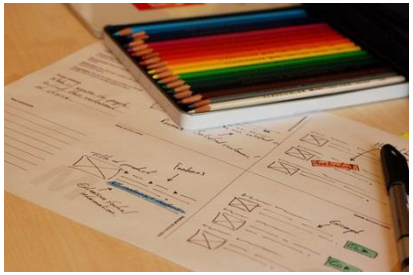
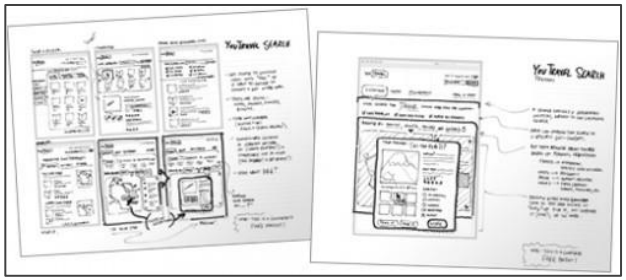
CMSC434

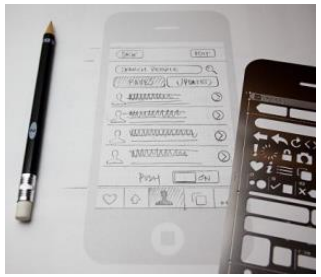
Introduction to Human-Computer Interaction

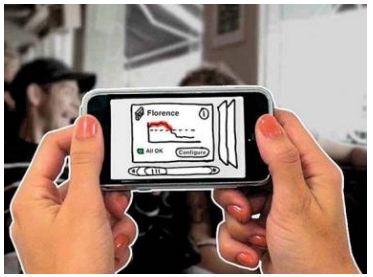
Week 08 | Lecture 15 | Oct 21, 2014

Building Android UIs







HCIL Human Computer Interaction Laboratory

@jonfroehlich

Assistant Professor Computer Science

UNIVERSITY OF MARYLAND

Due Thurs 10/28
Start immediately!

TA05 Paper Prototypes & Testing

Edit | 5 | ...

Paper Prototypes & User Testing

Posted: Friday, October 10
Deadline: Tuesday, October 28 (before classtime)
Point Total: This assignment is worth approximately 11% of your project grade (4.5% overall).

Assignment Overview

Your assignment is to convert a subset of your more promising sketches from the last assignment into paper prototypes and to test them with four users. You will create and **test two different versions** of your prototypes. As I've noted before in class, comparing two designs is far easier/better than evaluating one single design in isolation. For example, it's easier for a user to express preference between one design over another rather than articulating, exactly, why they don't like a single design.

Following from the IRB protocols we discussed in class, your participants must sign an informed consent document before taking part in your test. With the participant's permission, you should **video record all sessions**. This can be done with any type of recording device from a DSLR to a mobile phone camera.

Your report should include figures of your paper prototypes, a description of your study method, and a breakdown of your primary findings from the user testing. The video recordings will be used later in the semester for TA10 (so do not lose them).

What to Do

1. **To start, watch the Nielsen Norman** Paper Prototyping: A How-To Video. This will help prepare you for this project assignment and the next one as well. In addition, prototyping--and, specifically, rapid prototyping--is perhaps one of the most valuable techniques in HCI/design. So, it's worth spending some time on the readings and, of course, applying the prototyping concepts in your projects.

2. **Iterate and refine the three primary tasks** that users should be able to accomplish with your application (based on learnings/reflections from the last assignment).

3. **Then, transition to the first core part of the assignment: riff, iterate, and create two different paper prototypes for the three primary tasks.** That is, you must create "Paper Prototype #1" that allows your users to accomplish the three tasks one way and "Paper Prototype #2" that allows your users to accomplish the three primary tasks another way. The paper prototypes should be functionally different so that your users can compare and contrast their experiences with both of them. So, for example, the two prototypes should represent the tasks in fundamentally different ways. Remember, the focus here is not on aesthetics/beauty but rather on understandability, usability, approachability, and, to some degree, layout, widget type, etc.

4. **Once, you've created the two paper prototype designs, beta test them with members of your team and make requisite changes.** This "eating your own dog food" is a good way to catch errors before investing time in testing with actual users.

5. **Now you're ready for real user testing (the second core part of the assignment). Recruit four independent users to test out your paper prototypes.** These users cannot be members of the class; however, they can be members of Dr. Vibha Sazawal's section. Each user testing session must be done in isolation (that is, you cannot have more than one user testing at a time) with at least two experimenters present. For the testing session, you should follow this protocol:

    1. First, download and modify this IRB "informed consent" template to fit your project [link]. At the beginning of the user testing session, read the "Purpose of this Study" section of the consent form out loud to your participants. This should be done consistently for each participant. Then, give your participants a chance to read the entire consent form, ask questions, and, if they agree to participate, have them sign the form. If they do not agree to participate, simply wish them a nice day and recruit another participant (it can be slightly awkward but this happens!). If they do agree to participate, provide a copy of the form and take the signed copy for yourself (please scan in the signed consent form and include it in your report appendix).

    2. After the informed consent process, you can begin user testing the prototypes. Follow the method described in the Nielsen Norman "Paper Prototyping: A How-To Video." Because you have two different prototype designs for your tasks, you should fully test one

Home
Projects
Recent Changes
Pages and Files
Members
Settings

Course Pages
Home
Syllabus
Lectures
Readings

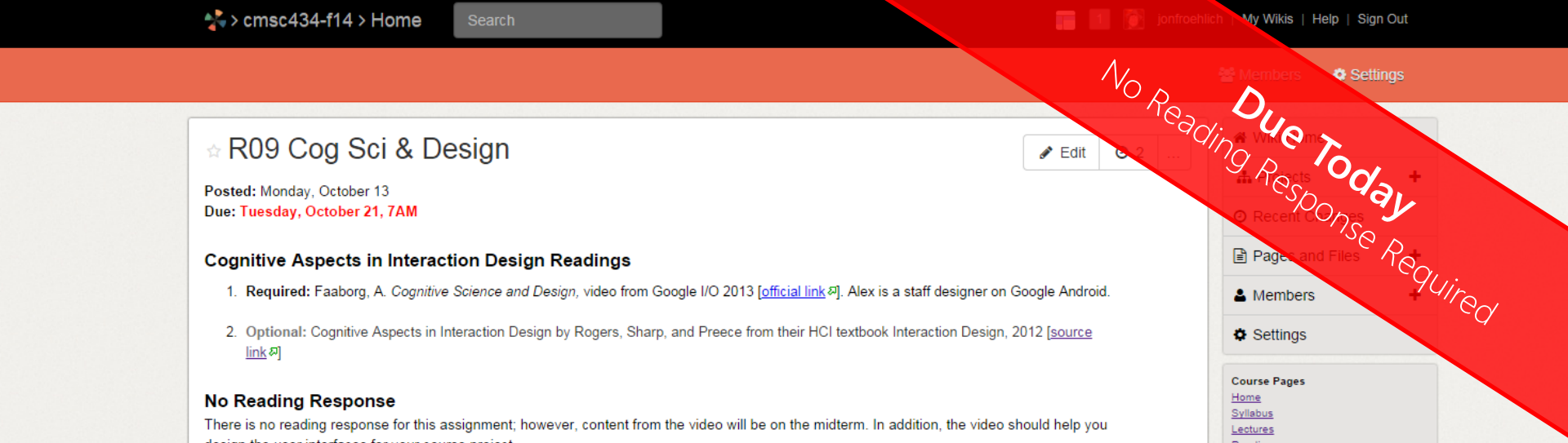
Individual Assignments
Hall of Fame/Shame
IA01 Background Survey - 9/3
IA02 Project Pitch - 9/7
IA03 Pitch Vote - 9/9
IA04 Understanding Metro Users - 10/2
IA05 Building an Android UI

Team Project Assignments
TA01 Group Collaboration Plan - 9/16
TA02 Formative Research - 9/28
TA03 Proposal Presentation - 9/30
TA04 Sketches & Storyboards - 10/14
TA05 Paper Prototypes - 10/28

Project Peer Assessment
Peer Assessment (TA01-TA03) - 10/7

Reading Assignments
R01 Brainstorming - 9/4
R02 Task-Centered Design - 9/9
R03 IDEO Shopping Cart - 9/11
R04 Ethnographic Approach - 9/16
R05 Prototyping - 9/23
R06 Presentation Tips - 9/30
R07 Sketching & Storyboards - 10/7
R08 Principles of Interaction - 10/14
R09 Cog Sci & Design - 10/21

edit navigation



R09 Cog Sci & Design

Edit

Posted: Monday, October 13
Due: Tuesday, October 21, 7AM

Cognitive Aspects in Interaction Design Readings

1. **Required:** Faaborg, A. *Cognitive Science and Design,* video from Google I/O 2013 [official link]. Alex is a staff designer on Google Android.

2. **Optional:** Cognitive Aspects in Interaction Design by Rogers, Sharp, and Preece from their HCI textbook Interaction Design, 2012 [source link]

No Reading Response

There is no reading response for this assignment; however, content from the video will be on the midterm. In addition, the video should help you design the user interfaces for your course project.

Pages and Files
Members
Settings

Course Pages
Home
Syllabus
Lectures
Readings

Individual Assignments
Hall of Fame/Shame
IA01 Background Survey - 9/3
IA02 Project Pitch - 9/7
IA03 Pitch Vote - 9/9
IA04 Understanding Metro Users - 10/2
IA05 Building an Android UI

Team Project Assignments
TA01 Group Collaboration Plan - 9/16
TA02 Formative Research - 9/28
TA03 Proposal Presentation - 9/30
TA04 Sketches & Storyboards - 10/14
TA05 Paper Prototypes - 10/28

Project Peer Assessment
Peer Assessment (TA01-TA03) - 10/7

Reading Assignments
R01 Brainstorming - 9/4
R02 Task-Centered Design - 9/9
R03 IDEO Shopping Cart - 9/11
R04 Ethnographic Approach - 9/16
R05 Prototyping - 9/23
R06 Presentation Tips - 9/30
R07 Sketching & Storyboards - 10/7
R08 Principles of Interaction - 10/14
R09 Cog Sci & Design - 10/21

edit navigation

Due Tues 11/11
Work with Partner / Start immediately!

IA05 Building an Android UI

Edit | 5 | ...

Building an Android UI

Posted: Monday, October 20
Due: Tuesday, November 11, 7AM
Point Total: This assignment is worth approximately 5% of your overall grade.
This assignment is to be completed with one CMSC434 partner of your choice.

Assignment Overview

In this assignment, you will design and implement a custom clock application for Android using two common programmatic approaches: first, using the Android Studio built-in design toolkit (Figure 1) to select, drag-and-drop, and layout widgets visually (along with XML); second, using a custom view that overrides the onDraw() method (Figure 2). Both clock implementations must show and update the current time (in real-time). While the WYSIWYG visual editor is convenient, relatively easy-to-understand, and useful for a wide range of tasks, the built-in widgets are rather limited. Most applications combine off-the-shelf widgets (*e.g.*, textboxes, dropdown menus, buttons) with their own views.

Android Graphical UI Builder

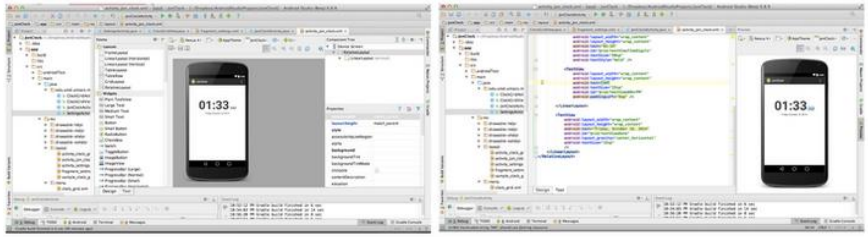


Figure 1. The Android graphical UI builder (WYSIWYG editor) allows you to build your interfaces visually—via a drag-and-drop interface—paired with an XML-based declarative syntax.

Android Custom Views

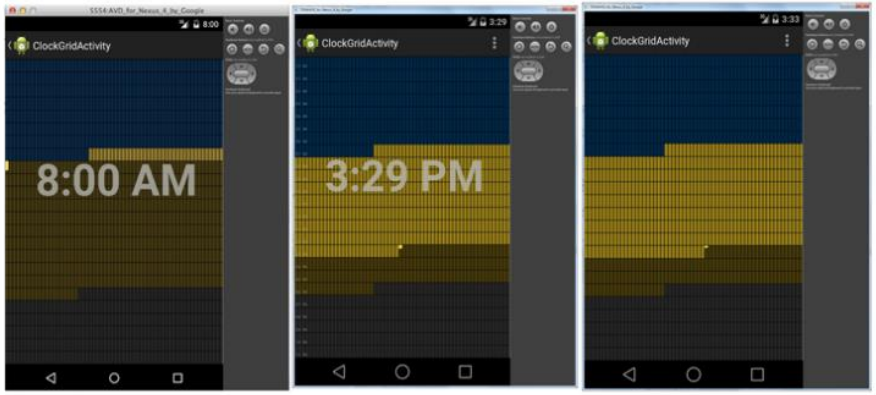


Figure 2. Many applications on your smartphone are built using custom views where the

Projects
Recent Changes
Pages and Files
Members
Settings

Course Pages
Home
Syllabus
Lectures
Readings

Individual Assignments
Hall of Fame/Shame
IA01 Background Survey - 9/3
IA02 Project Pitch - 9/7
IA03 Pitch Vote - 9/9
IA04 Understanding Metro Users - 10/2
IA05 Building an Android UI

Team Project Assignments
TA01 Group Collaboration Plan - 9/16
TA02 Formative Research - 9/28
TA03 Proposal Presentation - 9/30
TA04 Sketches & Storyboards - 10/14
TA05 Paper Prototypes - 10/28

Project Peer Assessment
Peer Assessment (TA01-TA03) - 10/7

Reading Assignments
R01 Brainstorming - 9/4
R02 Task-Centered Design - 9/9
R03 IDEO Shopping Cart - 9/11
R04 Ethnographic Approach - 9/16
R05 Prototyping - 9/23
R06 Presentation Tips - 9/30
R07 Sketching & Storyboards - 10/7
R08 Principles of Interaction - 10/14
R09 Cog Sci & Design - 10/21

edit navigation

☆ IA05 Building an Android UI

Edit 5 ...

Building an Android UI

Posted: Monday, October 20
Due: **Tuesday, November 11, 7AM**
Point Total: This assignment is worth approximately 5% of your overall grade.
This assignment is to be completed with one CMSC434 partner of your choice.

Assignment Overview

In this assignment, you will design and implement a custom clock application for Android using two common programmatic approaches: first, using the Android Studio built-in design toolkit (Figure 1) to select, drag-and-drop, and layout widgets visually (along with XML); second, using a custom view that overrides the onDraw() method (Figure 2). Both clock implementations must show and update the current time (in real-time). While the WYSIWYG visual editor is convenient, relatively easy-to-understand, and useful for a wide range of tasks, the built-in widgets are rather limited. Most applications combine off-the-shelf widgets (*e.g.*, textboxes, dropdown menus, buttons) with their own views.

Android Graphical UI Builder

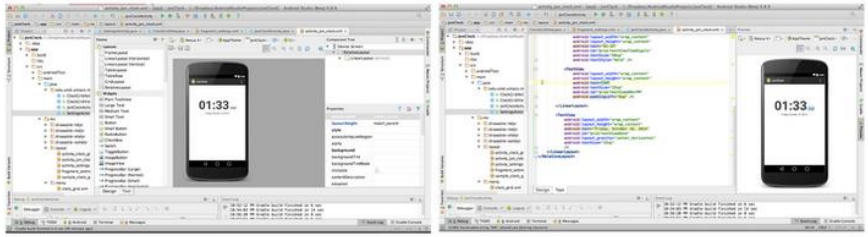


Figure 1. The Android graphical UI builder (WYSIWYG editor) allows you to build your interfaces visually—via a drag-and-drop interface—paired with an XML-based declarative syntax.

Android Custom Views

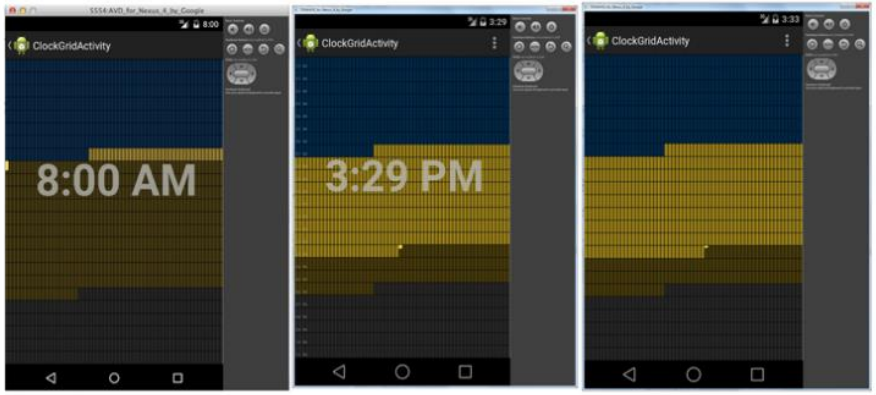


Figure 2. Many applications on your smartphone are built using custom views where the

Wiki Home
Projects +
Recent Changes
Pages and Files +
Members +
Settings

Course Pages
Home
Syllabus
Lectures
Readings

Individual Assignments
Hall of Fame/Shame
IA01 Background Survey - 9/3
IA02 Project Pitch - 9/7
IA03 Pitch Vote - 9/9
IA04 Understanding Metro Users - 10/2
IA05 Building an Android UI

Team Project Assignments
TA01 Group Collaboration Plan - 9/16
TA02 Formative Research - 9/28
TA03 Proposal Presentation - 9/30
TA04 Sketches & Storyboards - 10/14
TA05 Paper Prototypes - 10/28

Project Peer Assessment
Peer Assessment (TA01-TA03) - 10/7

Reading Assignments
R01 Brainstorming - 9/4
R02 Task-Centered Design - 9/9
R03 IDEO Shopping Cart - 9/11
R04 Ethnographic Approach - 9/16
R05 Prototyping - 9/23
R06 Presentation Tips - 9/30
R07 Sketching & Storyboards - 10/7
R08 Principles of Interaction - 10/14
R09 Cog Sci & Design - 10/21

edit navigation

Although this assignment is filed under "individual assignments," you are to do this assignment with a partner. You must tell me who you are partnering with by the end-of-day Thursday, October 23.

Assignment Parts

As you will be working with a partner on this assignment, you must git (a source control system) and host your code on github.

1. You will build the first clock using Android Studio's visual design tool, XML, and Java programming. This clock must show the current time and date, must be easily readable, and must update at least once a second.

2. The second clock must be implemented by deriving your own View class, overriding the onDraw function, and programmatically drawing the clock. Be creative. You have absolute, pixel-level control here. Before you begin, I want you to sketch 10 different ways of visualizing time. You will scan in these sketches and turn them in with your assignment.

3. You must implement a persistent Settings activity using fragments to allow the user to customize their clock views (*e.g.,* the timezone, 24 hour *vs.* 12 hour clock display, color, etc.). The settings should save across openings/closings of the app (hence, persistent).

4. You must brainstorm and implement one new feature not specified above. For example, in the application I showed in class, my ClockGridView downloads the sunrise and sunset times from a web API and visualizes them in the clock. What will you add? Some ideas: alarm functionality (careful, the alarm should persist even if you close the app--so you should implement this as a service), some cool transition animation between numbers changing, a movement logger.

5. Once you've finished your implementation, you will record a demonstration with voiceover (and/or captions) and upload it to YouTube. The video should describe in detail the above four requirements.

Deliverables

On Canvas, you will submit a link to your YouTube video and a github link to the sourcecode. The github root should

Although this assignment is filed under "individual assignments," you are to do this assignment with a partner. You must tell me who you are partnering with by the end-of-day Thursday, October 23.

Assignment Parts

As you will be working with a partner on this assignment, you must git (a source control system) and host your code on github.

1. You will build the first clock using Android Studio's visual design tool, XML, and Java programming. This clock must show the current time and date, must be easily readable, and must update at least once a second.

2. The second clock must be implemented by deriving your own View class, overriding the onDraw function, and programmatically drawing the clock. Be creative. You have absolute, pixel-level control here. Before you begin, I want you to sketch 10 different ways of visualizing time. You will scan in these sketches and turn them in with your assignment.

3. You must implement a persistent Settings activity using fragments to allow the user to customize their clock views (*e.g.,* the timezone, 24 hour *vs.* 12 hour clock display, color, etc.). The settings should save across openings/closings of the app (hence, persistent).

4. You must brainstorm and implement one new feature not specified above. For example, in the application I showed in class, my ClockGridView downloads the sunrise and sunset times from a web API and visualizes them in the clock. What will you add? Some ideas: alarm functionality (careful, the alarm should persist even if you close the app--so you should implement this as a service), some cool transition animation between numbers changing, a movement logger.

5. Once you've finished your implementation, you will record a demonstration with voiceover (and/or captions) and upload it to YouTube. The video should describe in detail the above four requirements.

Deliverables

On Canvas, you will submit a link to your YouTube video and a github link to the sourcecode. The github root should

Although this assignment is filed under "individual assignments," you are to do this assignment with a partner. You must tell me who you are partnering with by the end-of-day Thursday, October 23.

Assignment Parts

As you will be working with a partner on this assignment, you must git (a source control system) and host your code on github.

1. You will build the first clock using Android Studio's visual design tool, XML, and Java programming. This clock must show the current time and date, must be easily readable, and must update at least once a second.

2. The second clock must be implemented by deriving your own View class, overriding the onDraw function, and programmatically drawing the clock. Be creative. You have absolute, pixel-level control here. Before you begin, I want you to sketch 10 different ways of visualizing time. You will scan in these sketches and turn them in with your assignment.

3. You must implement a persistent Settings activity using fragments to allow the user to customize their clock views (*e.g.,* the timezone, 24 hour *vs.* 12 hour clock display, color, etc.). The settings should save across openings/closings of the app (hence, persistent).

4. You must brainstorm and implement one new feature not specified above. For example, in the application I showed in class, my ClockGridView downloads the sunrise and sunset times from a web API and visualizes them in the clock. What will you add? Some ideas: alarm functionality (careful, the alarm should persist even if you close the app--so you should implement this as a service), some cool transition animation between numbers changing, a movement logger.

5. Once you've finished your implementation, you will record a demonstration with voiceover (and/or captions) and upload it to YouTube. The video should describe in detail the above four requirements.

Deliverables

On Canvas, you will submit a link to your YouTube video and a github link to the sourcecode. The github root should

Although this assignment is filed under "individual assignments," you are to do this assignment with a partner. You must tell me who you are partnering with by the end-of-day Thursday, October 23.

Assignment Parts

As you will be working with a partner on this assignment, you must git (a source control system) and host your code on github.

1. You will build the first clock using Android Studio's visual design tool, XML, and Java programming. This clock must show the current time and date, must be easily readable, and must update at least once a second.

2. The second clock must be implemented by deriving your own View class, overriding the onDraw function, and programmatically drawing the clock. Be creative. You have absolute, pixel-level control here. Before you begin, I want you to sketch 10 different ways of visualizing time. You will scan in these sketches and turn them in with your assignment.

3. You must implement a persistent Settings activity using fragments to allow the user to customize their clock views (*e.g.,* the timezone, 24 hour *vs.* 12 hour clock display, color, etc.). The settings should save across openings/closings of the app (hence, persistent).

4. You must brainstorm and implement one new feature not specified above. For example, in the application I showed in class, my ClockGridView downloads the sunrise and sunset times from a web API and visualizes them in the clock. What will you add? Some ideas: alarm functionality (careful, the alarm should persist even if you close the app--so you should implement this as a service), some cool transition animation between numbers changing, a movement logger.

5. Once you've finished your implementation, you will record a demonstration with voiceover (and/or captions) and upload it to YouTube. The video should describe in detail the above four requirements.

Deliverables

On Canvas, you will submit a link to your YouTube video and a github link to the sourcecode. The github root should


10 Best Android Clock Widget
androidWIDGETcenter
MAKE YOUR CLOCK
kalos Hero clocks widget
One More Clock Widget
Battery.
Time.
Weather.
Beautiful Widget on home screen
Waiting for extension AdBlock...


11 12 1
10 2
9
8 4
PEBBLE
7 6 5
3:10
April 20
ITzISapebble
AtQUARTERten
ktwentynfive
halfoPASTtof
sevenfivetwo
onesixeleven
teneninefour
eighttwelves
THREEhoclock
03
13
Sa 0420 45
10

Although this assignment is filed under "individual assignments," you are to do this assignment with a partner. You must tell me who you are partnering with by the end-of-day Thursday, October 23.

Assignment Parts

As you will be working with a partner on this assignment, you must git (a source control system) and host your code on github.

1. You will build the first clock using Android Studio's visual design tool, XML, and Java programming. This clock must show the current time and date, must be easily readable, and must update at least once a second.

2. The second clock must be implemented by deriving your own View class, overriding the onDraw function, and programmatically drawing the clock. Be creative. You have absolute, pixel-level control here. Before you begin, I want you to sketch 10 different ways of visualizing time. You will scan in these sketches and turn them in with your assignment.

3. You must implement a persistent Settings activity using fragments to allow the user to customize their clock views (*e.g.,* the timezone, 24 hour *vs.* 12 hour clock display, color, etc.). The settings should save across openings/closings of the app (hence, persistent).

4. You must brainstorm and implement one new feature not specified above. For example, in the application I showed in class, my ClockGridView downloads the sunrise and sunset times from a web API and visualizes them in the clock. What will you add? Some ideas: alarm functionality (careful, the alarm should persist even if you close the app--so you should implement this as a service), some cool transition animation between numbers changing, a movement logger.

5. Once you've finished your implementation, you will record a demonstration with voiceover (and/or captions) and upload it to YouTube. The video should describe in detail the above four requirements.

Deliverables

On Canvas, you will submit a link to your YouTube video and a github link to the sourcecode. The github root should

Although this assignment is filed under "individual assignments," you are to do this assignment with a partner. You must tell me who you are partnering with by the end-of-day Thursday, October 23.

Assignment Parts

As you will be working with a partner on this assignment, you must git (a source control system) and host your code on github.

1. You will build the first clock using Android Studio's visual design tool, XML, and Java programming. This clock must show the current time and date, must be easily readable, and must update at least once a second.

2. The second clock must be implemented by deriving your own View class, overriding the onDraw function, and programmatically drawing the clock. Be creative. You have absolute, pixel-level control here. Before you begin, I want you to sketch 10 different ways of visualizing time. You will scan in these sketches and turn them in with your assignment.

3. You must implement a persistent Settings activity using fragments to allow the user to customize their clock views (*e.g.,* the timezone, 24 hour *vs.* 12 hour clock display, color, etc.). The settings should save across openings/closings of the app (hence, persistent).

4. You must brainstorm and implement one new feature not specified above. For example, in the application I showed in class, my ClockGridView downloads the sunrise and sunset times from a web API and visualizes them in the clock. What will you add? Some ideas: alarm functionality (careful, the alarm should persist even if you close the app--so you should implement this as a service), some cool transition animation between numbers changing, a movement logger.

5. Once you've finished your implementation, you will record a demonstration with voiceover (and/or captions) and upload it to YouTube. The video should describe in detail the above four requirements.

Deliverables

On Canvas, you will submit a link to your YouTube video and a github link to the sourcecode. The github root should

Although this assignment is filed under "individual assignments," you are to do this assignment with a partner. You must tell me who you are partnering with by the end-of-day Thursday, October 23.

Assignment Parts

As you will be working with a partner on this assignment, you must git (a source control system) and host your code on github.

1. You will build the first clock using Android Studio's visual design tool, XML, and Java programming. This clock must show the current time and date, must be easily readable, and must update at least once a second.

2. The second clock must be implemented by deriving your own View class, overriding the onDraw function, and programmatically drawing the clock. Be creative. You have absolute, pixel-level control here. Before you begin, I want you to sketch 10 different ways of visualizing time. You will scan in these sketches and turn them in with your assignment.

3. You must implement a persistent Settings activity using fragments to allow the user to customize their clock views (*e.g.,* the timezone, 24 hour *vs.* 12 hour clock display, color, etc.). The settings should save across openings/closings of the app (hence, persistent).

4. You must brainstorm and implement one new feature not specified above. For example, in the application I showed in class, my ClockGridView downloads the sunrise and sunset times from a web API and visualizes them in the clock. What will you add? Some ideas: alarm functionality (careful, the alarm should persist even if you close the app--so you should implement this as a service), some cool transition animation between numbers changing, a movement logger.

5. Once you've finished your implementation, you will record a demonstration with voiceover (and/or captions) and upload it to YouTube. The video should describe in detail the above four requirements.

Deliverables

On Canvas, you will submit a link to your YouTube video and a github link to the sourcecode. The github root should

As you will be working with a partner on this assignment, you must git (a source control system) and host your code on github.

1. You will build the first clock using Android Studio's visual design tool, XML, and Java programming. This clock must show the current time and date, must be easily readable, and must update at least once a second.

2. The second clock must be implemented by deriving your own View class, overriding the onDraw function, and programmatically drawing the clock. Be creative. You have absolute, pixel-level control here. Before you begin, I want you to sketch 10 different ways of visualizing time. You will scan in these sketches and turn them in with your assignment.

3. You must implement a persistent Settings activity using fragments to allow the user to customize their clock views (*e.g.,* the timezone, 24 hour *vs.* 12 hour clock display, color, etc.). The settings should save across openings/closings of the app (hence, persistent).

4. You must brainstorm and implement one new feature not specified above. For example, in the application I showed in class, my ClockGridView downloads the sunrise and sunset times from a web API and visualizes them in the clock. What will you add? Some ideas: alarm functionality (careful, the alarm should persist even if you close the app--so you should implement this as a service), some cool transition animation between numbers changing, a movement logger.

5. Once you've finished your implementation, you will record a demonstration with voiceover (and/or captions) and upload it to YouTube. The video should describe in detail the above four requirements.

Deliverables

On Canvas, you will submit a link to your YouTube video and a github link to the sourcecode. The github root should contain a readme that describes how to run your program.

Resources

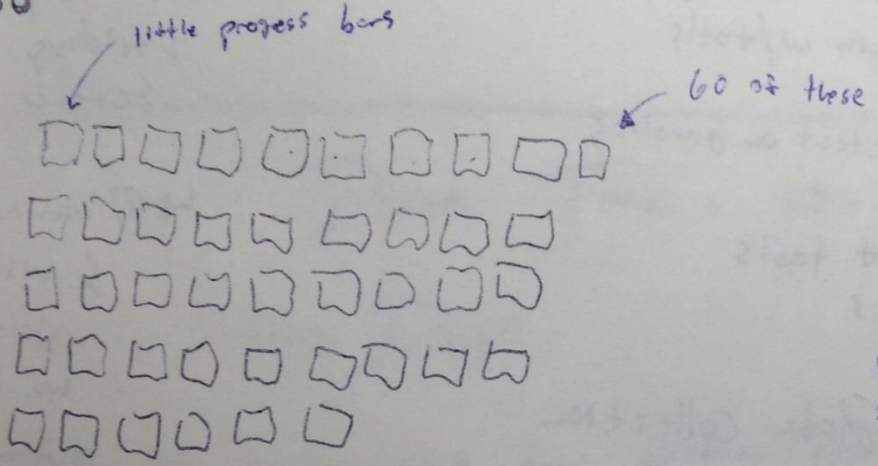
1. Training for Android Developers, Getting Started
2. Training for Android Developers, Building a Simple User Interface
3. Training for Android Developers, Best Practices for User Interface
4. Training for Android Developers, Best Practices for Interaction and Engagement
5. API Guides, User Interface

My Clocks Demo

1440 minutes in a day

1 Day
24 hours
60 minutes
60 seconds

24 x 60

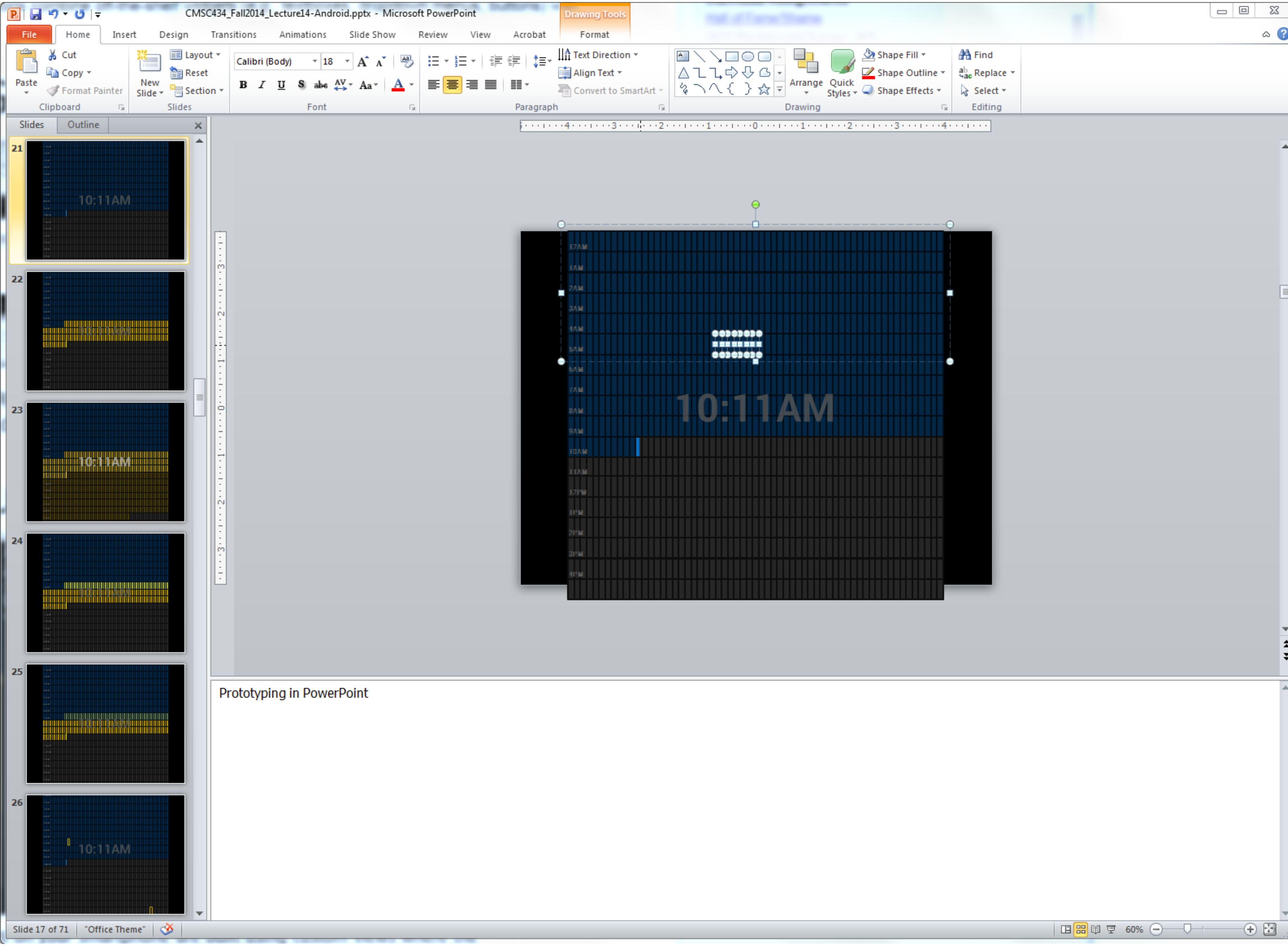


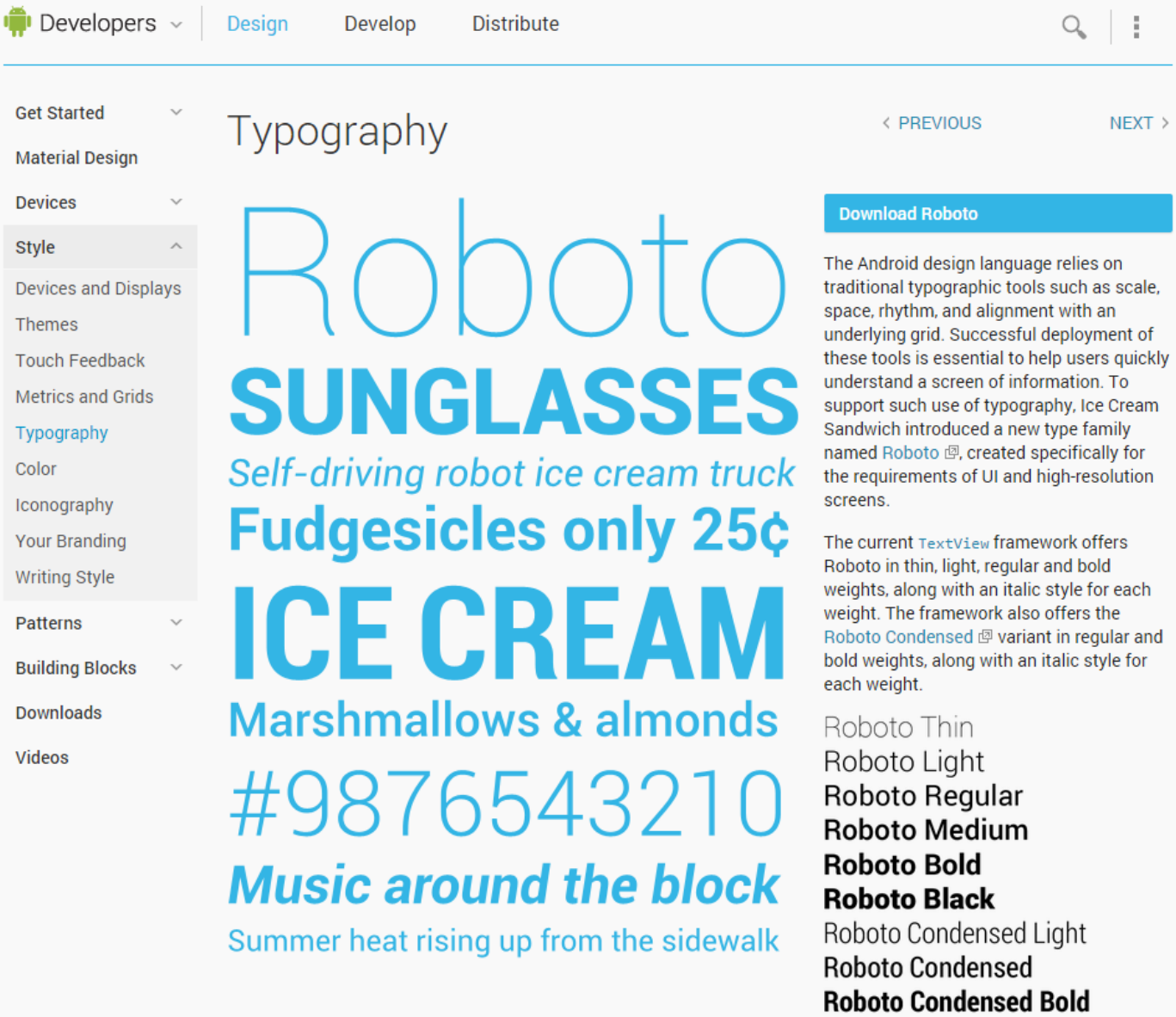
Galaxy S5 is 1080w x 1920h

1080/60 = 18

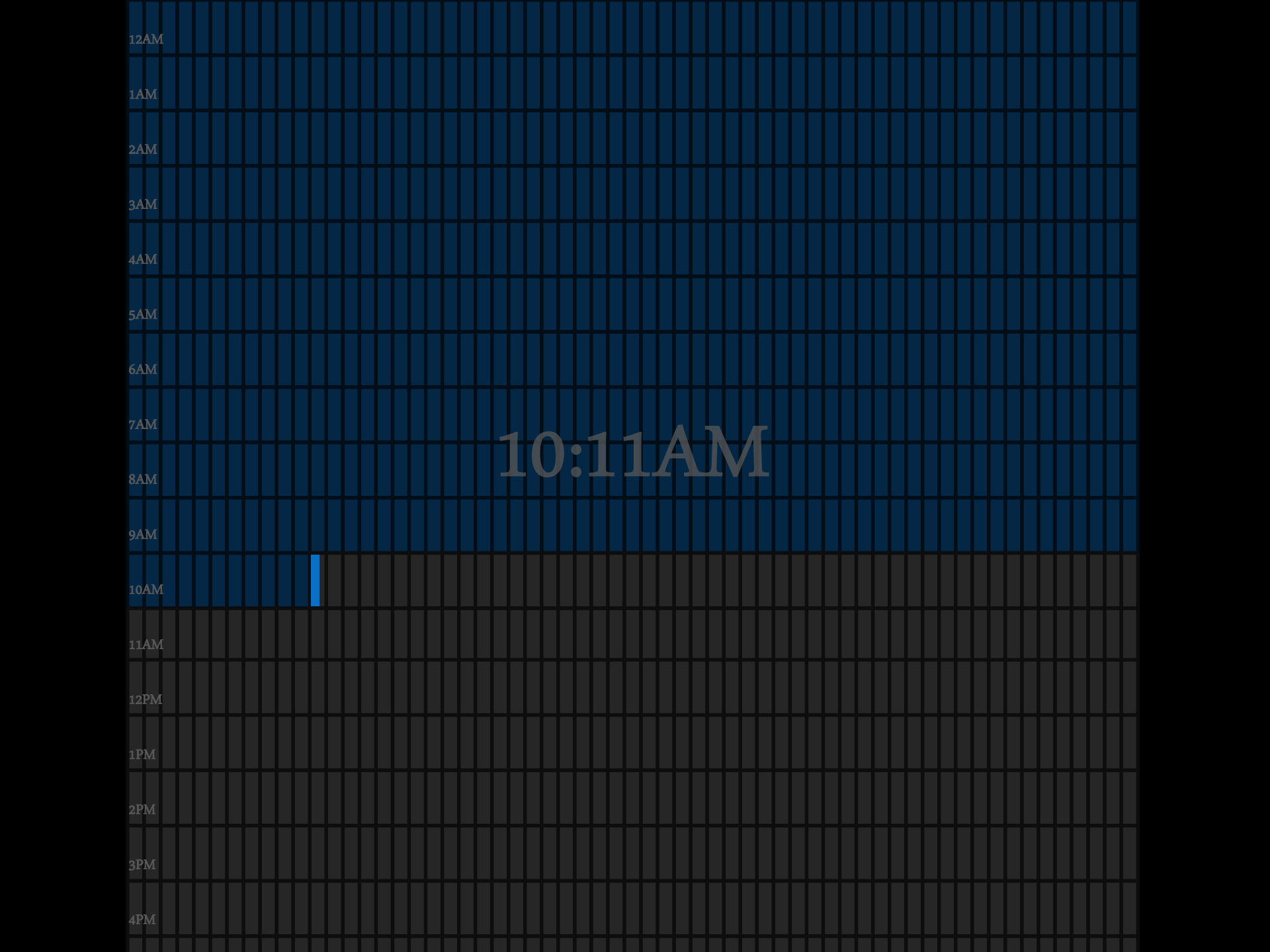
17 pixel boxes (width) w/one pixel space

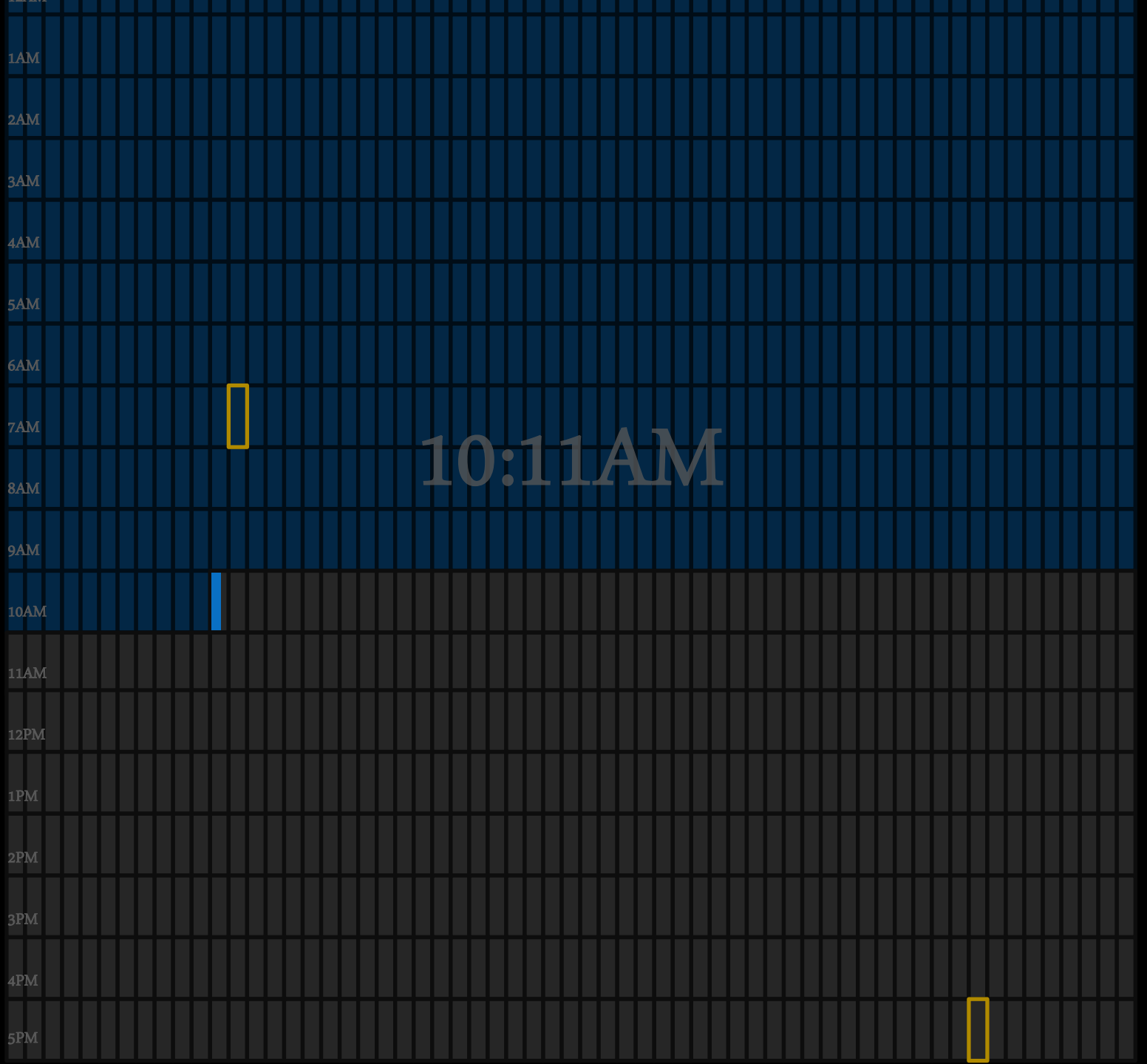
60 per row (1 min chunks)
30 per row (2 min chunks)
17 per row (3 min chunks)

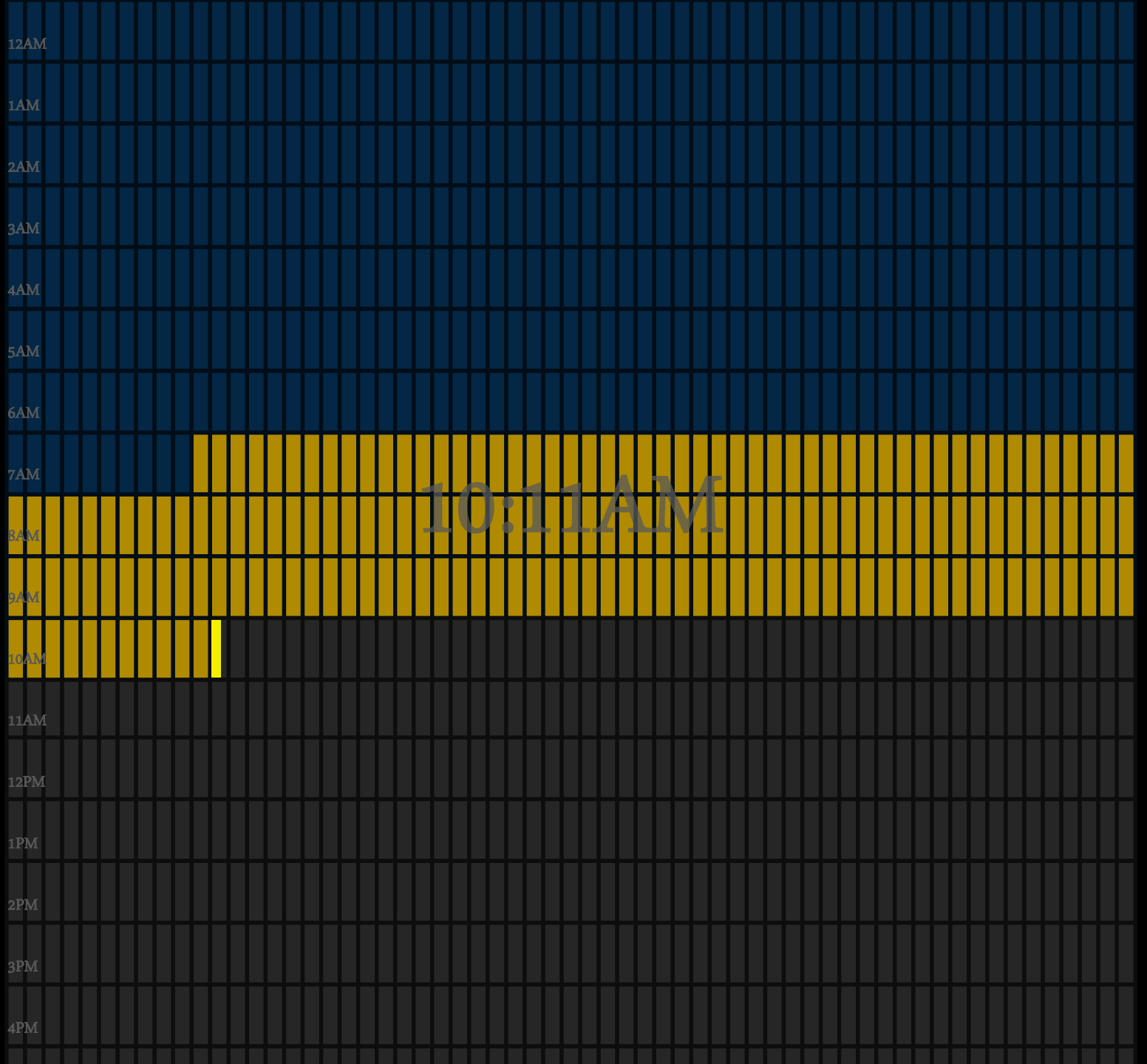

CMSC434_Fall2014_Lecture14-Android.pptx - Microsoft PowerPoint
Drawing Tools
File
Home
Insert
Design
Transitions
Animations
Slide Show
Review
View
Acrobat
Format
Cut
Copy
Format Painter
Paste
Clipboard
Layout
Reset
Section
New Slide
Slides
Calibri (Body)
18
Font
Text Direction
Align Text
Convert to SmartArt
Paragraph
Arrange
Quick Styles
Shape Fill
Shape Outline
Shape Effects
Drawing
Find
Replace
Select
Editing
Slides
Outline
10:11AM
Prototyping in PowerPoint
Slide 17 of 71
"Office Theme"
60%

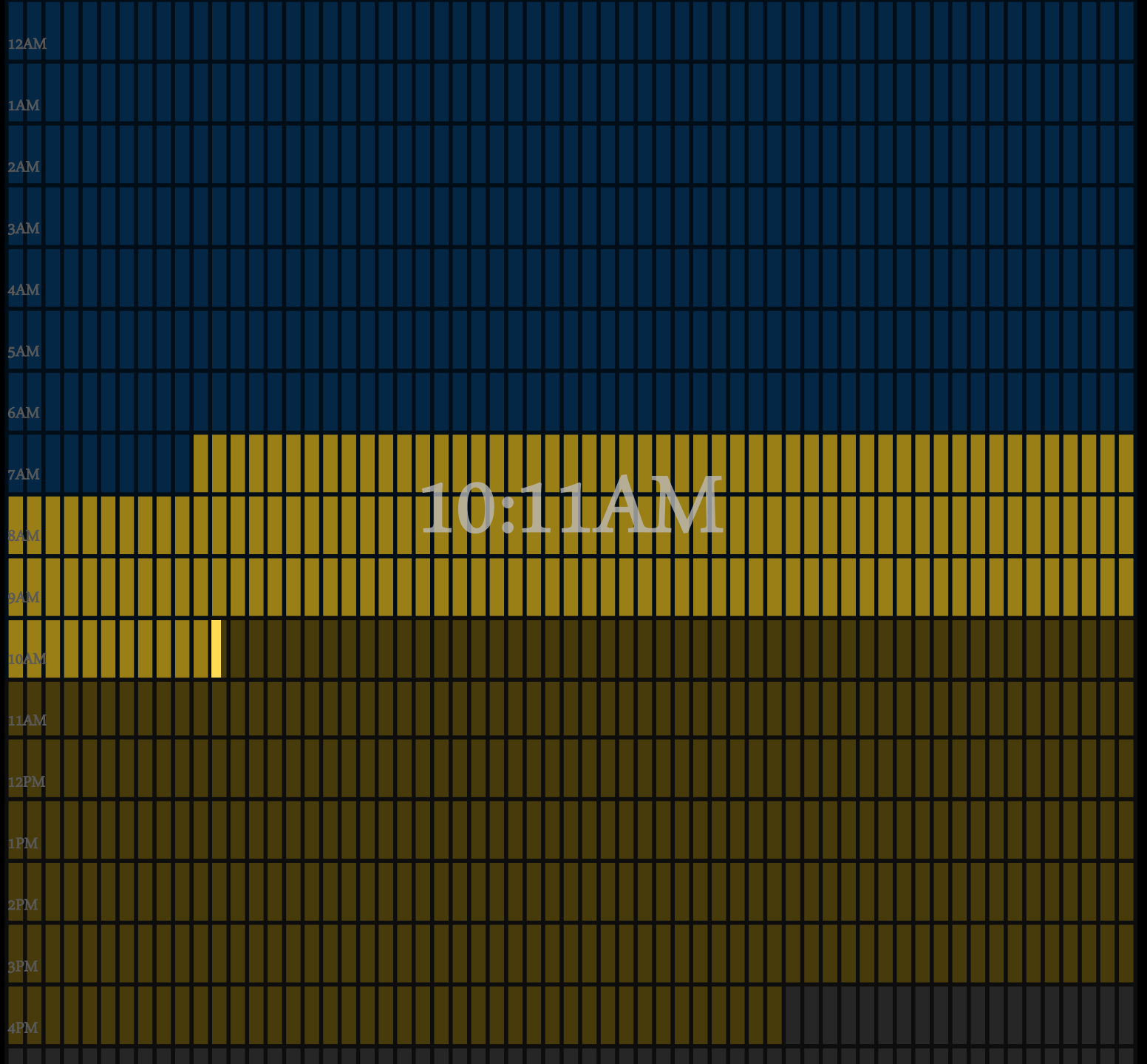


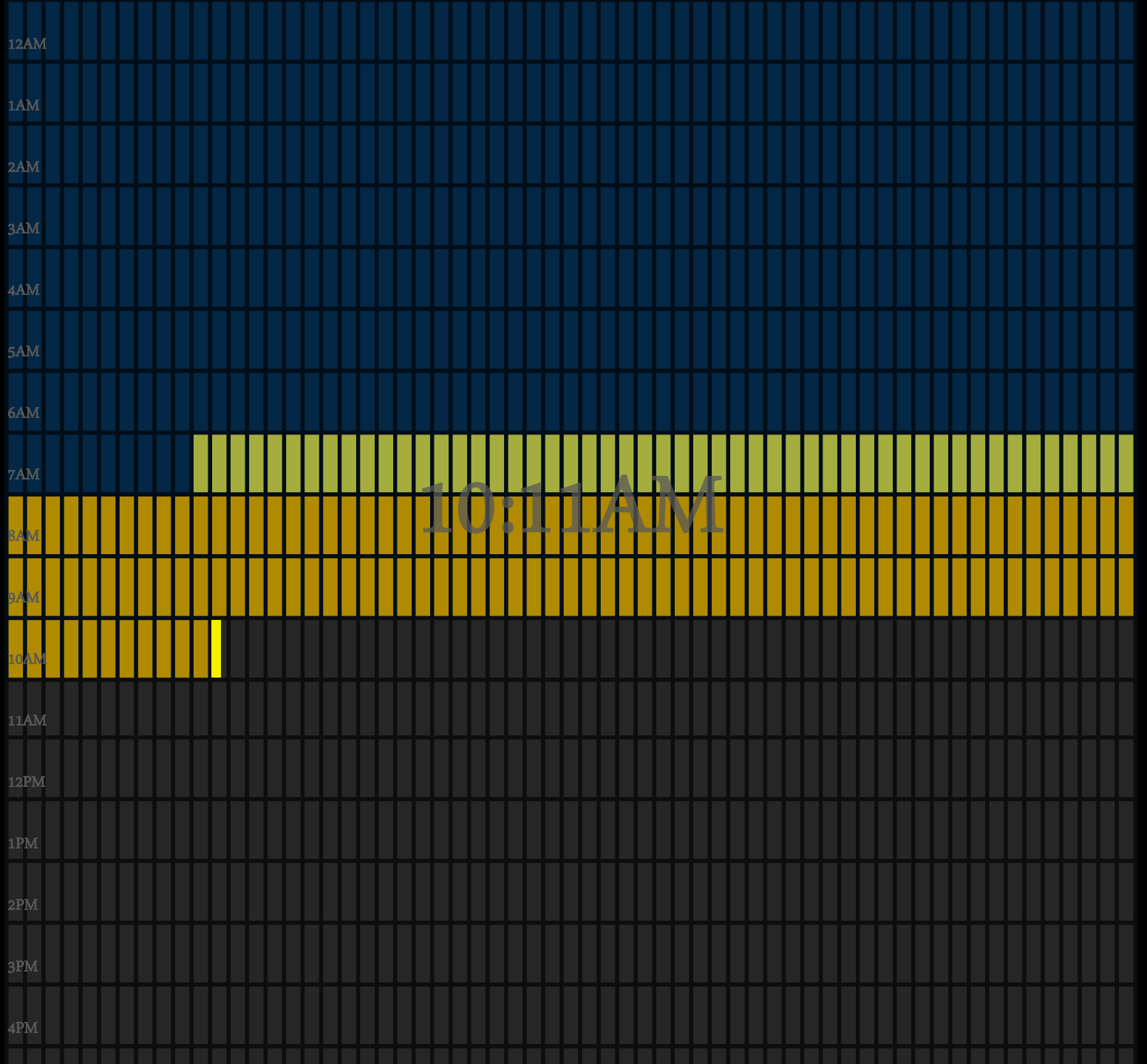
If you want to prototype interfaces say, with PowerPoint or KeyNote, then you need to have the Android font installed, which you can download here: http://developer.android.com/design/style/typography.html

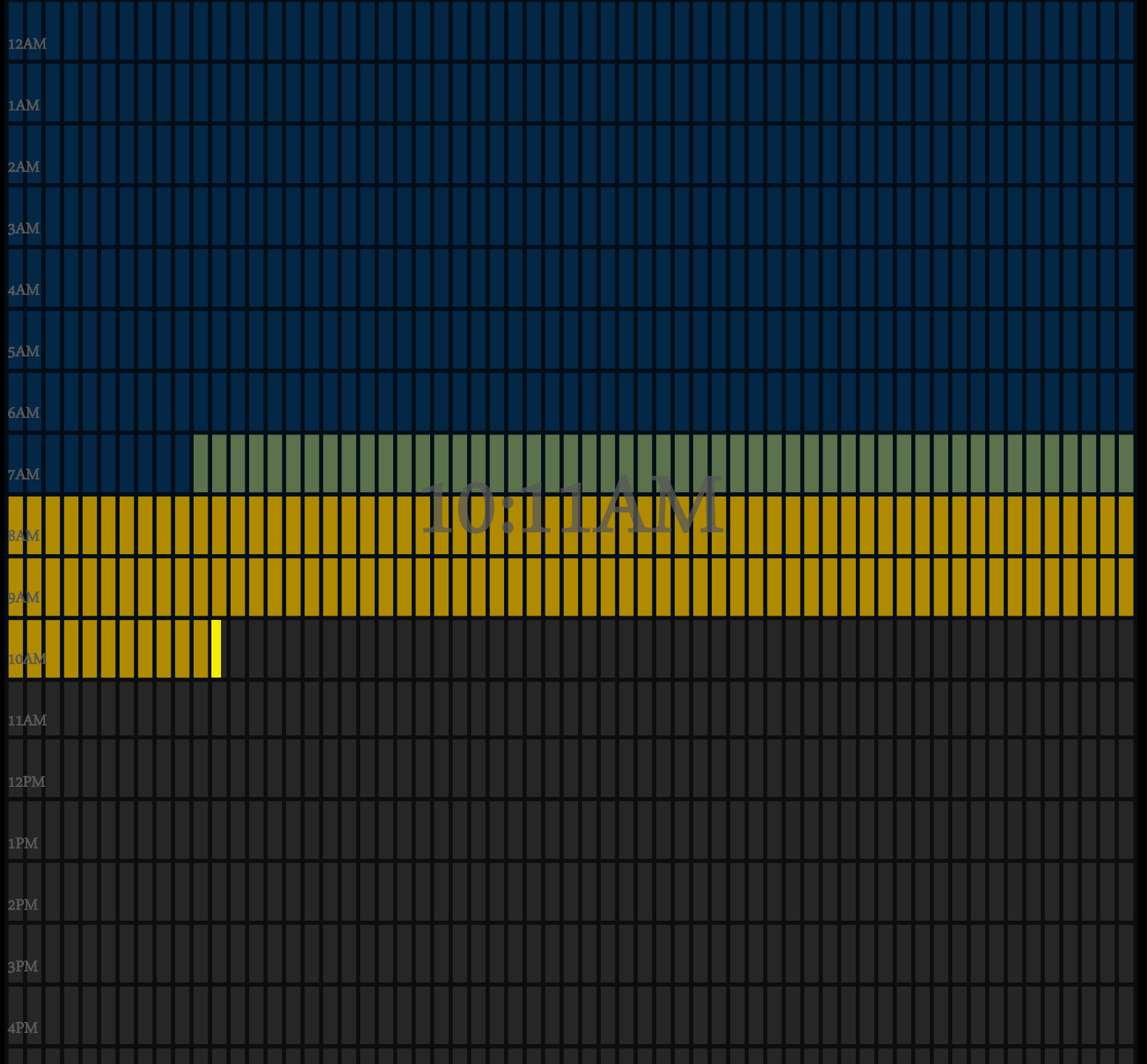

12AM
1AM
2AM
3AM
4AM
5AM
6AM
7AM
8AM
9AM
10AM
11AM
12PM
1PM
2PM
3PM
4PM
10:11AM

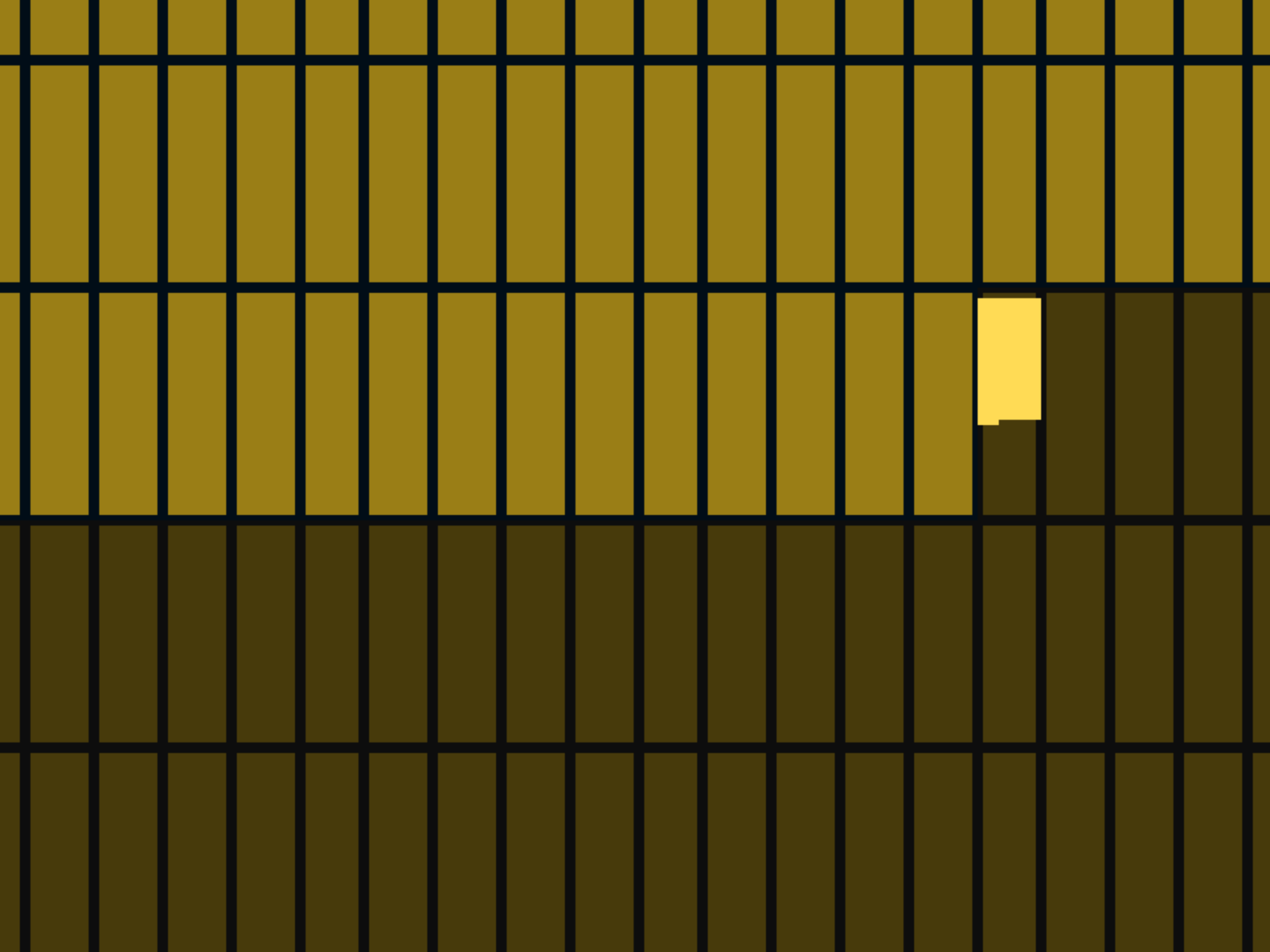

1AM
2AM
3AM
4AM
5AM
6AM
7AM
8AM
9AM
10AM
11AM
12PM
1PM
2PM
3PM
4PM
5PM
10:11AM


12AM
1AM
2AM
3AM
4AM
5AM
6AM
7AM
8AM
9AM
10AM
11AM
12PM
1PM
2PM
3PM
4PM
10:11AM


12AM
1AM
2AM
3AM
4AM
5AM
6AM
7AM
8AM
9AM
10AM
11AM
12PM
1PM
2PM
3PM
4PM
10:11AM


12AM
1AM
2AM
3AM
4AM
5AM
6AM
7AM
10:11AM
8AM
9AM
10AM
11AM
12PM
1PM
2PM
3PM
4PM

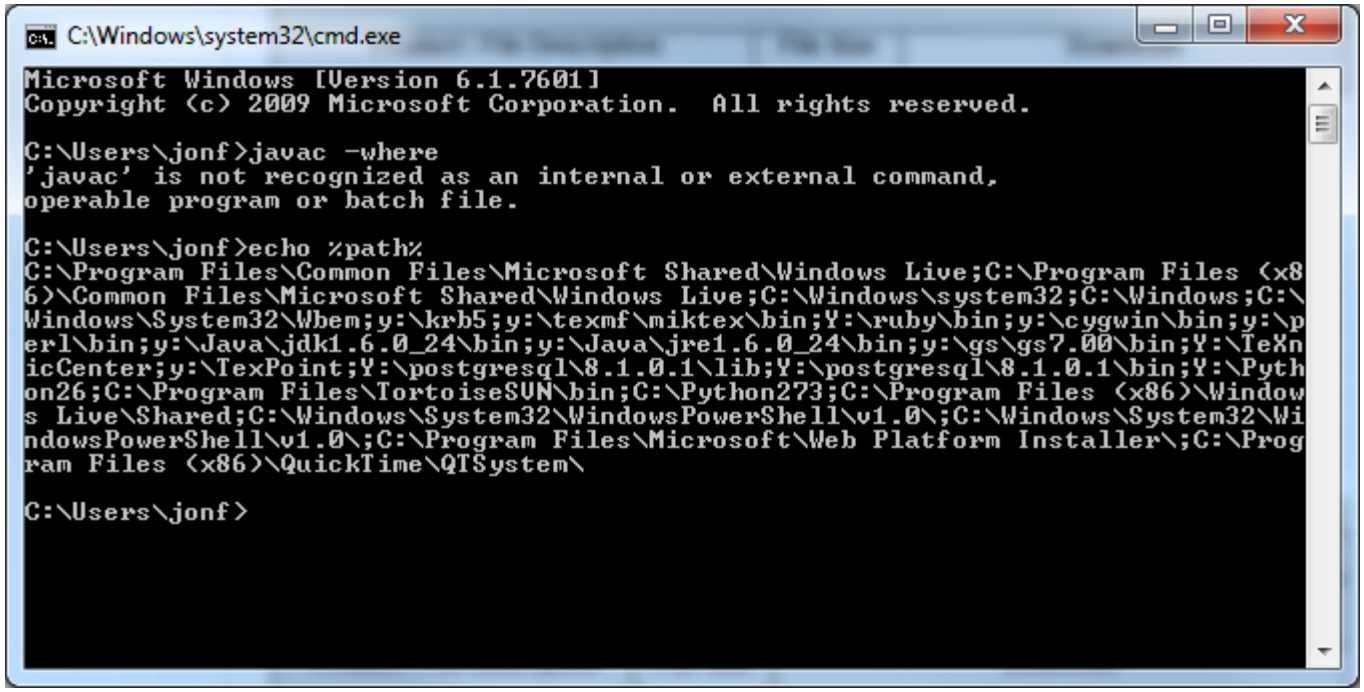

12AM
1AM
2AM
3AM
4AM
5AM
6AM
7AM
8AM
9AM
10AM
11AM
12PM
1PM
2PM
3PM
4PM
10:11AM



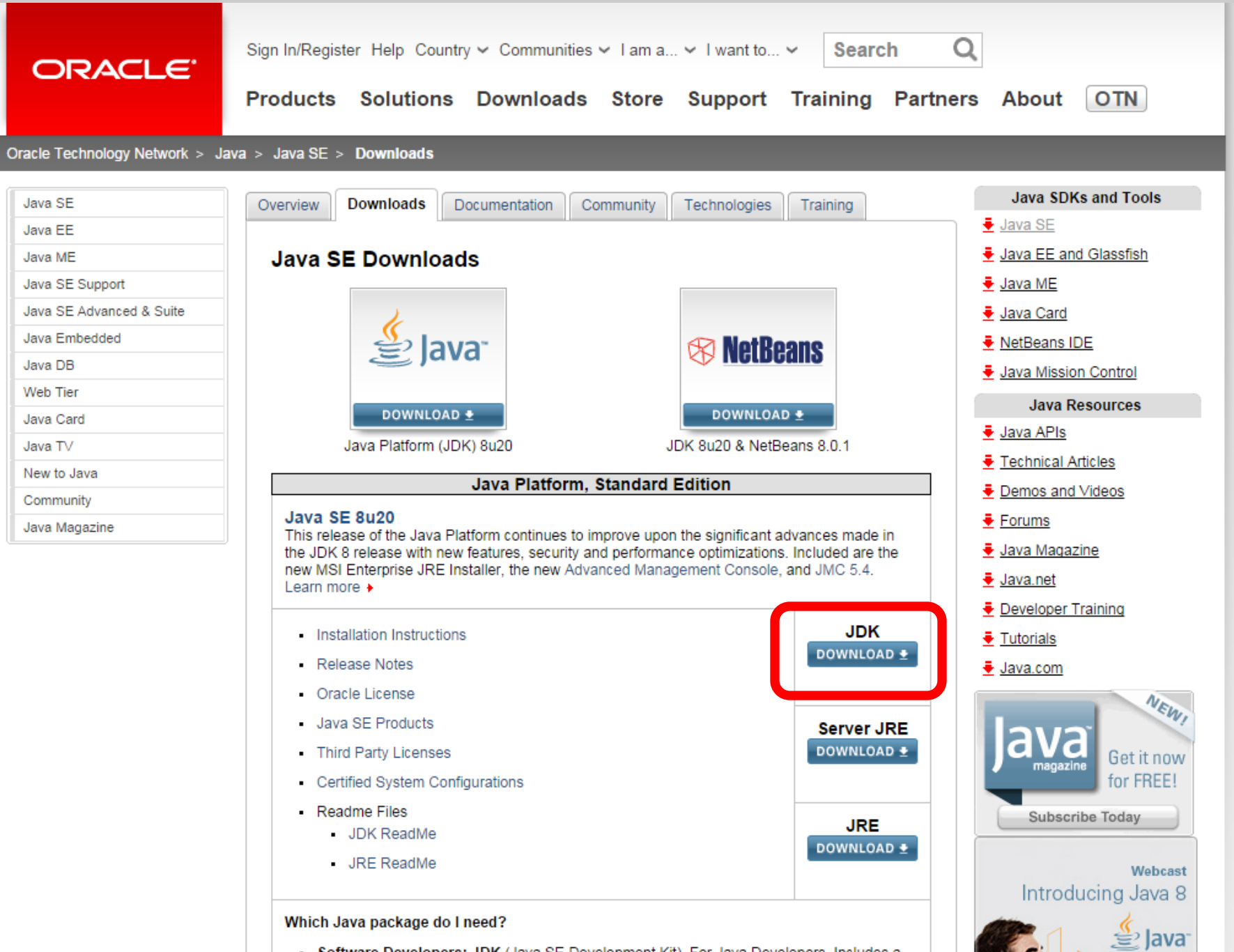
android

setup

Check for Java SDK Installation



Download Java SDK: http://developer.android.com/sdk/installing/studio.html



Download Java SDK: http://developer.android.com/sdk/installing/studio.html

Java SE Development Kit 8u20

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	135.24 MB	jdk-8u20-linux-i586.rpm
Linux x86	154.87 MB	jdk-8u20-linux-i586.tar.gz
Linux x64	135.6 MB	jdk-8u20-linux-x64.rpm
Linux x64	153.42 MB	jdk-8u20-linux-x64.tar.gz
Mac OS X x64	209.11 MB	jdk-8u20-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	137.83 MB	jdk-8u20-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	97.09 MB	jdk-8u20-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	137.16 MB	jdk-8u20-solaris-x64.tar.Z
Solaris x64	94.22 MB	jdk-8u20-solaris-x64.tar.gz
Windows x86	134.88 MB	jdk-8u20-windows-i586.exe
Windows x64	173.08 MB	jdk-8u20-windows-x64.exe

Mac

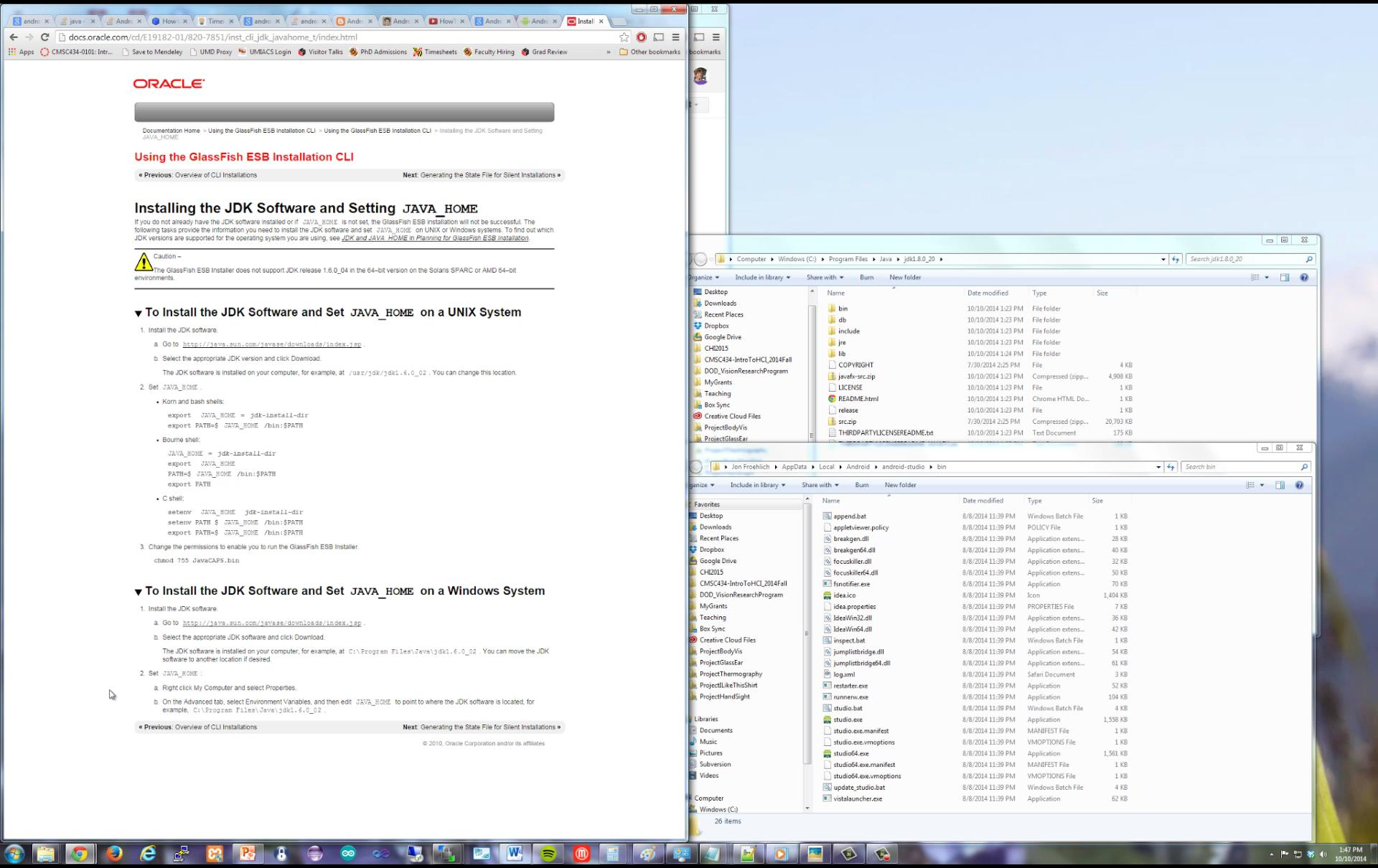
Windows

Java SE Development Kit 8u20 Demos and Samples Downloads

Java SE Development Kit 8u20 Demos and Samples Downloads are released under the Oracle BSD License.

Product / File Description	File Size	Download
Linux x86	58.65 MB	jdk-8u20-linux-i586-demos.rpm
Linux x86	58.49 MB	jdk-8u20-linux-i586-demos.tar.gz
Linux x64	58.71 MB	jdk-8u20-linux-x64-demos.rpm
Linux x64	58.56 MB	jdk-8u20-linux-x64-demos.tar.gz
Mac OS X	59.22 MB	jdk-8u20-macosx-x86_64-demos.zip
Solaris SPARC 64-bit	13.57 MB	jdk-8u20-solaris-sparcv9-demos.tar.Z
Solaris SPARC 64-bit	9.28 MB	jdk-8u20-solaris-sparcv9-demos.tar.gz
Solaris x64	13.5 MB	jdk-8u20-solaris-x64-demos.tar.Z
Solaris x64	9.22 MB	jdk-8u20-solaris-x64-demos.tar.gz
Windows x86	60.33 MB	jdk-8u20-windows-i586-demos.zip
Windows x64	60.43 MB	jdk-8u20-windows-x64-demos.zip

Windows: Setup Environment Variables (As Necessary)



docs.oracle.com/cd/E19182-01/820-7851/inst_cli_jdk_javahome_t/index.html

ORACLE

Documentation Home > Using the GlassFish ESB Installation CLI > Using the GlassFish ESB Installation CLI > Installing the JDK Software and Setting JAVA_HOME

Using the GlassFish ESB Installation CLI

« Previous: Overview of CLI Installations Next: Generating the State File for Silent Installations »

Installing the JDK Software and Setting JAVA_HOME

If you do not already have the JDK software installed or if JAVA_HOME is not set, the GlassFish ESB installation will not be successful. The following tasks provide the information you need to install the JDK software and set JAVA_HOME on UNIX or Windows systems. To find out which JDK versions are supported for the operating system you are using, see *JDK and JAVA_HOME in Planning for GlassFish ESB Installation*.

Caution –
The GlassFish ESB Installer does not support JDK release 1.6.0_04 in the 64–bit version on the Solaris SPARC or AMD 64–bit environments.

▼ To Install the JDK Software and Set JAVA_HOME on a UNIX System

1. Install the JDK software.
   a. Go to http://java.sun.com/javase/downloads/index.jsp .
   b. Select the appropriate JDK version and click Download.
      The JDK software is installed on your computer, for example, at /usr/jdk/jdk1.6.0_02 . You can change this location.
2. Set JAVA_HOME .
   - Korn and bash shells:
     ```
     export  JAVA_HOME = jdk-install-dir
     export PATH=$ JAVA_HOME /bin:$PATH
     ```
   - Bourne shell:
     ```
     JAVA_HOME = jdk-install-dir
     export  JAVA_HOME
     PATH=$ JAVA_HOME /bin:$PATH
     export PATH
     ```
   - C shell:
     ```
     setenv  JAVA_HOME  jdk-install-dir
     setenv PATH $ JAVA_HOME /bin:$PATH
     export PATH=$ JAVA_HOME /bin:$PATH
     ```
3. Change the permissions to enable you to run the GlassFish ESB Installer.
   ```
   chmod 755 JavaCAPS.bin
   ```

▼ To Install the JDK Software and Set JAVA_HOME on a Windows System

1. Install the JDK software.
   a. Go to http://java.sun.com/javase/downloads/index.jsp .
   b. Select the appropriate JDK software and click Download.
      The JDK software is installed on your computer, for example, at C:\Program Files\Java\jdk1.6.0_02 . You can move the JDK software to another location if desired.
2. Set JAVA_HOME :
   a. Right click My Computer and select Properties.
   b. On the Advanced tab, select Environment Variables, and then edit JAVA_HOME to point to where the JDK software is located, for example, C:\Program Files\Java\jdk1.6.0_02 .

« Previous: Overview of CLI Installations Next: Generating the State File for Silent Installations »

© 2010, Oracle Corporation and/or its affiliates

Computer ▸ Windows (C:) ▸ Program Files ▸ Java ▸ jdk1.8.0_20 ▸

Name	Date modified	Type	Size
bin	10/10/2014 1:23 PM	File folder	
db	10/10/2014 1:23 PM	File folder	
include	10/10/2014 1:23 PM	File folder	
jre	10/10/2014 1:23 PM	File folder	
lib	10/10/2014 1:24 PM	File folder	
COPYRIGHT	7/30/2014 2:25 PM	File	4 KB
javafx-src.zip	10/10/2014 1:23 PM	Compressed (zipp...	4,908 KB
LICENSE	10/10/2014 1:23 PM	File	1 KB
README.html	10/10/2014 1:23 PM	Chrome HTML Do...	1 KB
release	10/10/2014 1:23 PM	File	1 KB
src.zip	7/30/2014 2:25 PM	Compressed (zipp...	20,703 KB
THIRDPARTYLICENSEREADME.txt	10/10/2014 1:23 PM	Text Document	175 KB

Jon Froehlich ▸ AppData ▸ Local ▸ Android ▸ android-studio ▸ bin ▸

Name	Date modified	Type	Size
append.bat	8/8/2014 11:39 PM	Windows Batch File	1 KB
appletviewer.policy	8/8/2014 11:39 PM	POLICY File	1 KB
breakgen.dll	8/8/2014 11:39 PM	Application extens...	28 KB
breakgen64.dll	8/8/2014 11:39 PM	Application extens...	40 KB
focuskiller.dll	8/8/2014 11:39 PM	Application extens...	32 KB
focuskiller64.dll	8/8/2014 11:39 PM	Application extens...	50 KB
fsnotifier.exe	8/8/2014 11:39 PM	Application	70 KB
idea.ico	8/8/2014 11:39 PM	Icon	1,404 KB
idea.properties	8/8/2014 11:39 PM	PROPERTIES File	7 KB
IdeaWin32.dll	8/8/2014 11:39 PM	Application extens...	36 KB
IdeaWin64.dll	8/8/2014 11:39 PM	Application extens...	42 KB
inspect.bat	8/8/2014 11:39 PM	Windows Batch File	1 KB
jumplistbridge.dll	8/8/2014 11:39 PM	Application extens...	54 KB
jumplistbridge64.dll	8/8/2014 11:39 PM	Application extens...	61 KB
log.xml	8/8/2014 11:39 PM	Safari Document	3 KB
restarter.exe	8/8/2014 11:39 PM	Application	52 KB
runnerw.exe	8/8/2014 11:39 PM	Application	104 KB
studio.bat	8/8/2014 11:39 PM	Windows Batch File	4 KB
studio.exe	8/8/2014 11:39 PM	Application	1,558 KB
studio.exe.manifest	8/8/2014 11:39 PM	MANIFEST File	1 KB
studio.exe.vmoptions	8/8/2014 11:39 PM	VMOPTIONS File	1 KB
studio64.exe	8/8/2014 11:39 PM	Application	1,561 KB
studio64.exe.manifest	8/8/2014 11:39 PM	MANIFEST File	1 KB
studio64.exe.vmoptions	8/8/2014 11:39 PM	VMOPTIONS File	1 KB
update_studio.bat	8/8/2014 11:39 PM	Windows Batch File	4 KB
vistalauncher.exe	8/8/2014 11:39 PM	Application	62 KB

26 items

Download Android Studio: http://developer.android.com/sdk/installing/studio.html

Android Studio

BETA

Android Studio is a new Android development environment based on IntelliJ IDEA. It provides new features and improvements over Eclipse ADT and will be the official Android IDE once it's ready. On top of the capabilities you expect from IntelliJ, Android Studio offers:

- Flexible Gradle-based build system.
- Build variants and multiple APK generation.
- Expanded template support for Google Services and various device types.
- Rich layout editor with support for theme editing.
- Lint tools to catch performance, usability, version compatibility, and other problems.
- ProGuard and app-signing capabilities.
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine.

Caution: Android Studio is currently in **beta**. Some features are not yet implemented and you may encounter bugs. If you are not comfortable using an unfinished product, you may want to instead download (or continue to use) Eclipse with ADT.

VIEW ALL DOWNLOADS AND SIZES

SYSTEM REQUIREMENTS

Download Android Studio Beta v0.8.6
with the Android SDK for Windows

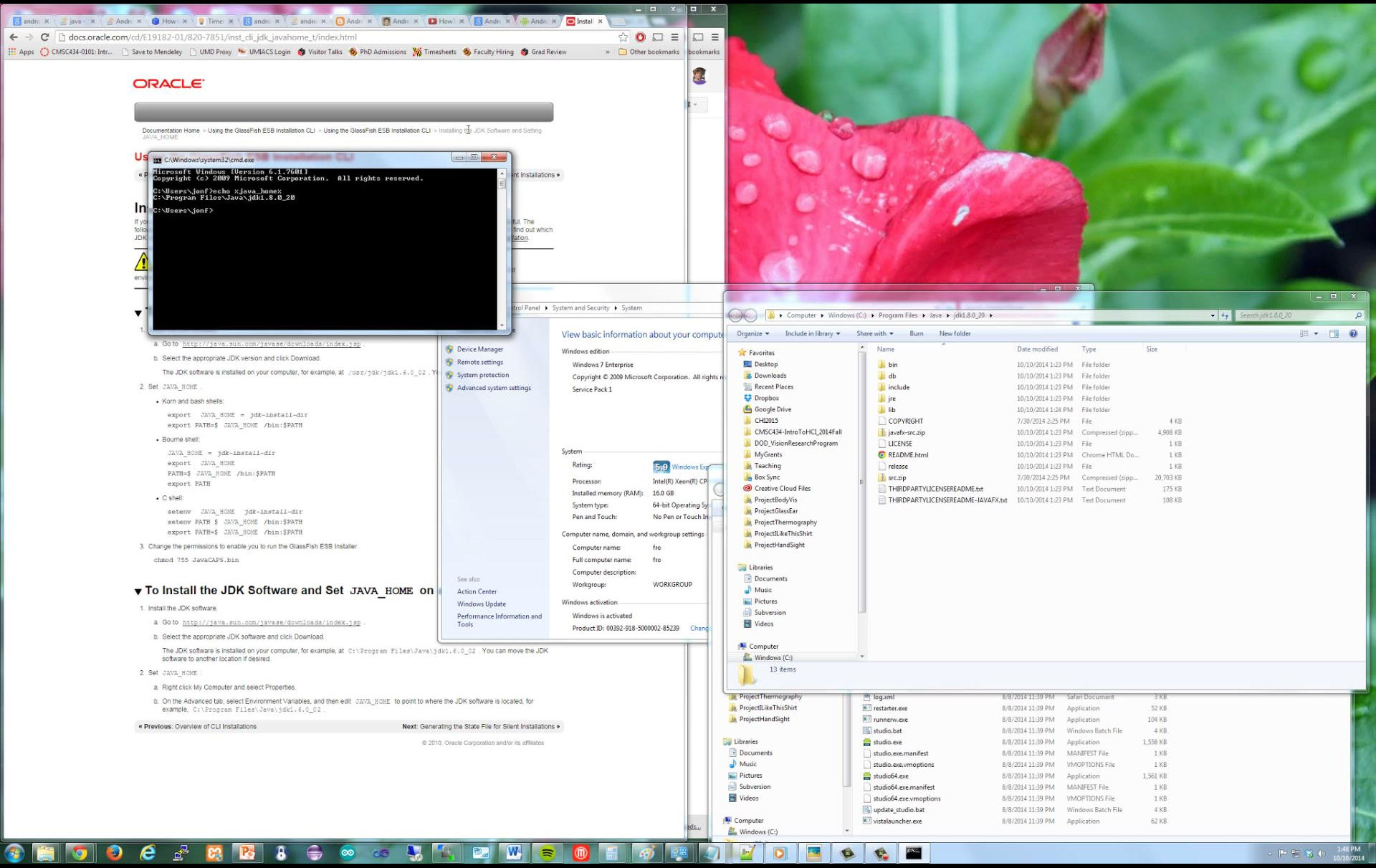
This download includes:

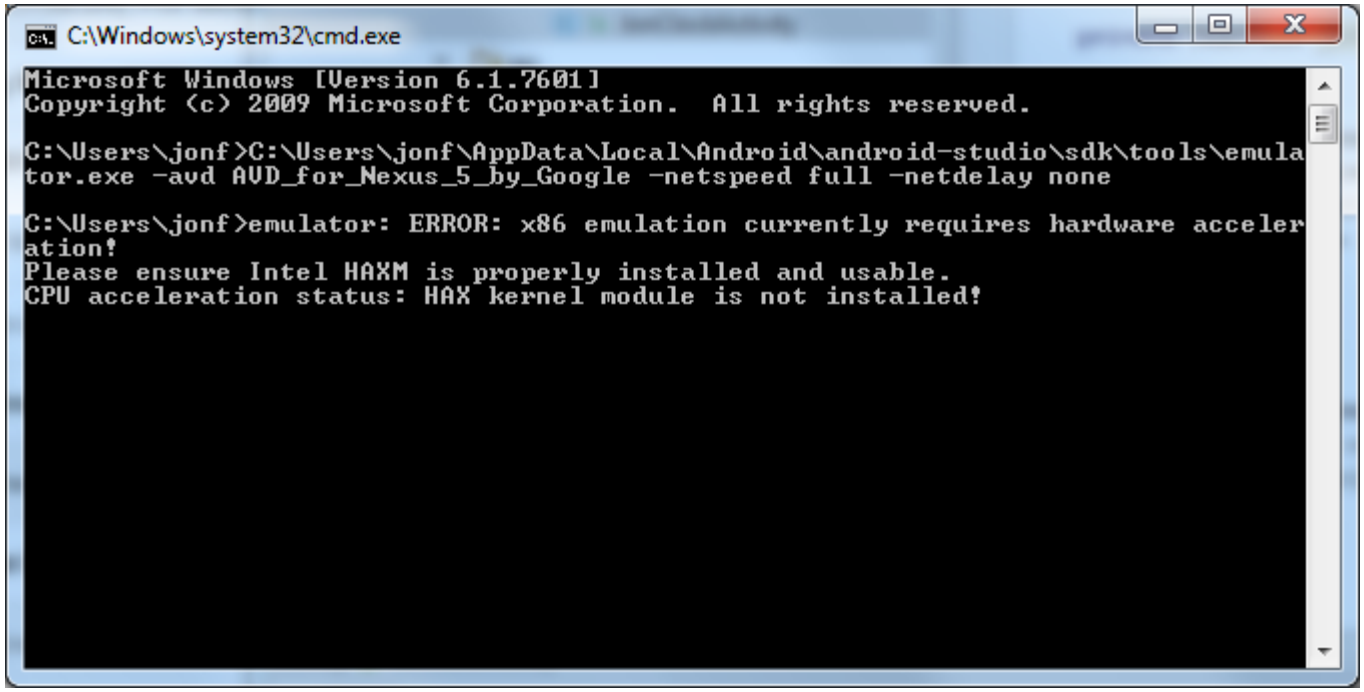
- Android Studio Beta
- All the Android SDK Tools to design, test, and debug your app
- A version of the Android platform to compile your app
- A version of the Android system image to run your app in the emulator

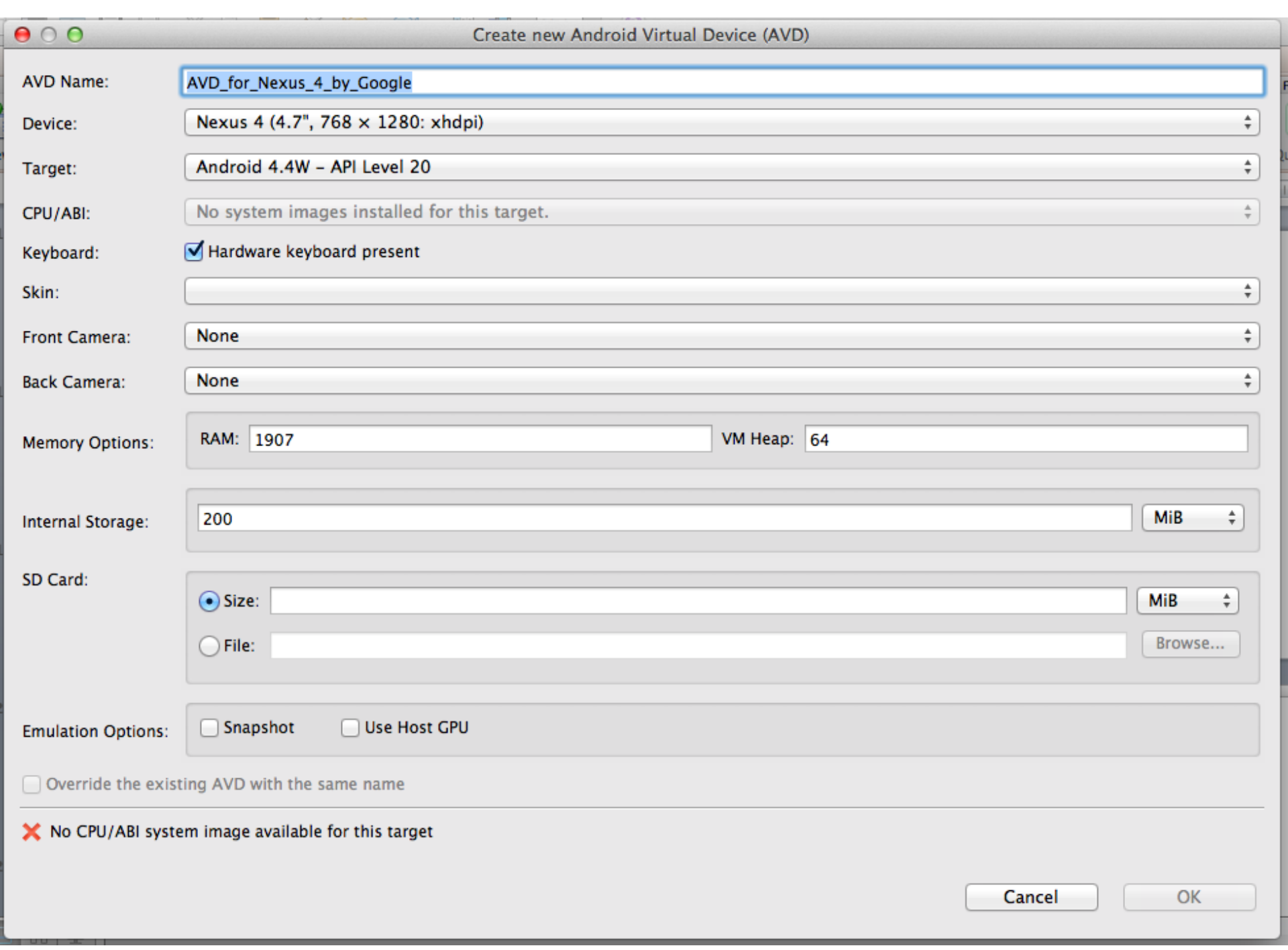
Android Studio vs. Eclipse ADT Comparison

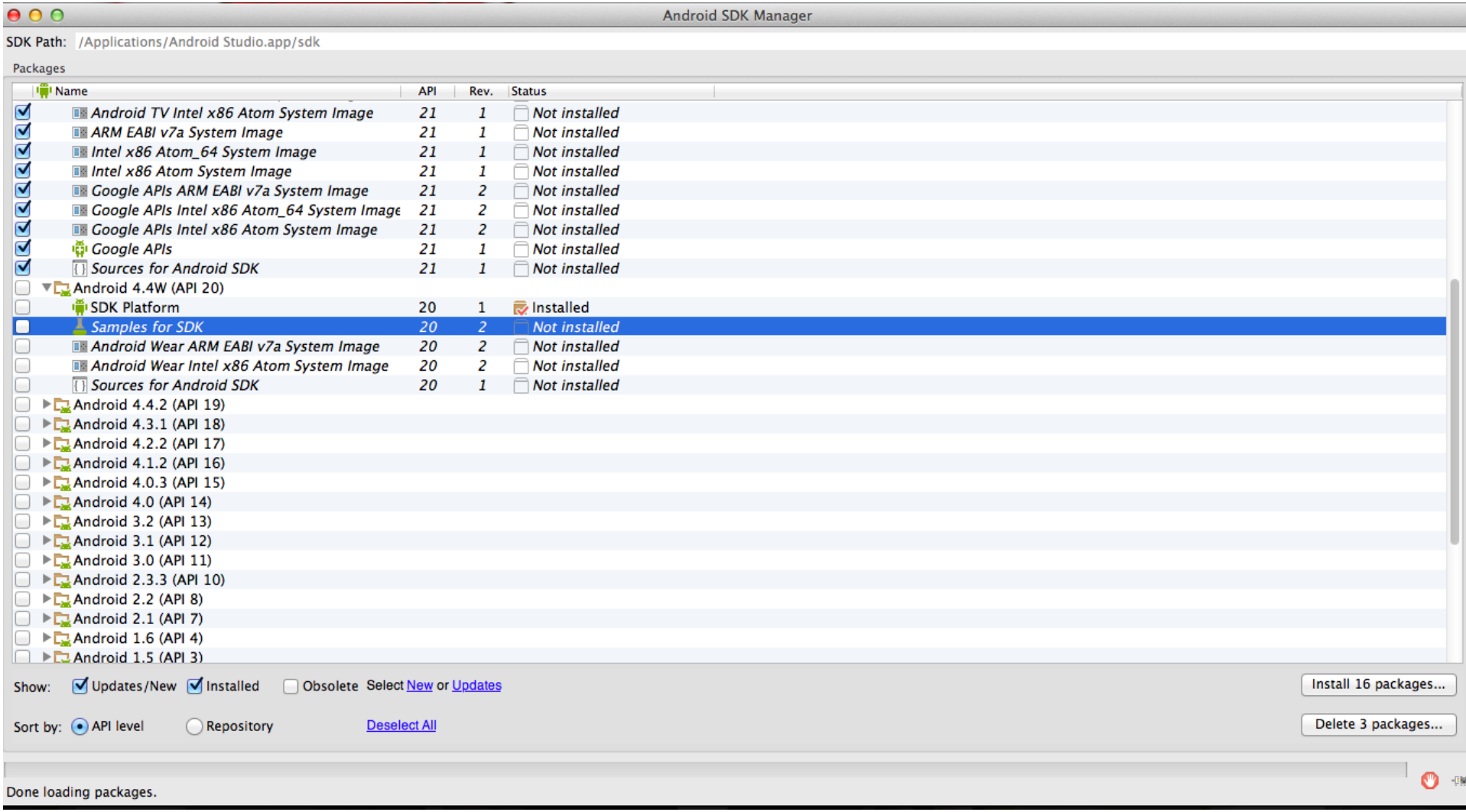
The following table lists some key differences between Android Studio Beta and Eclipse with ADT.

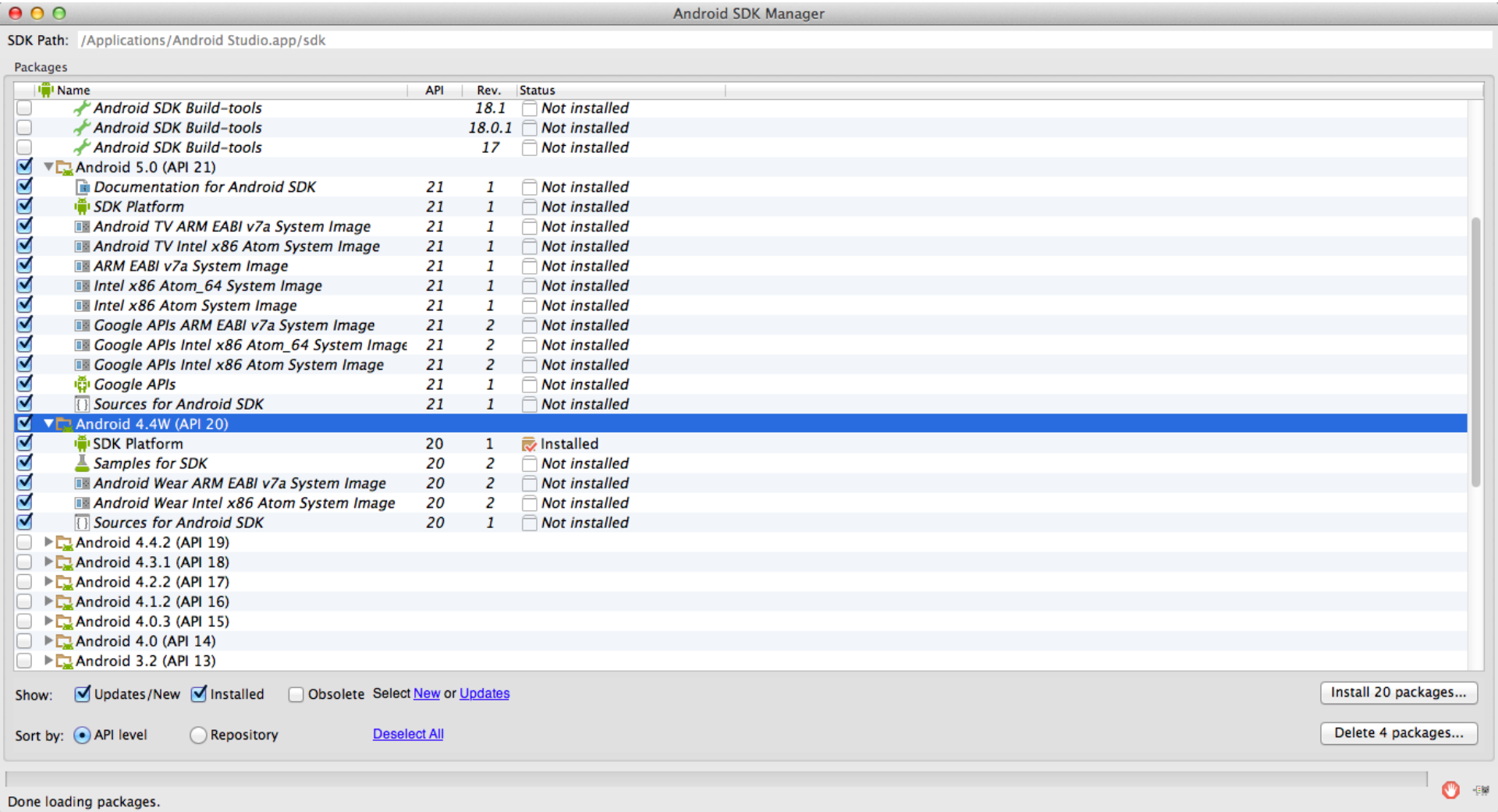
Installing SDKs and System Images

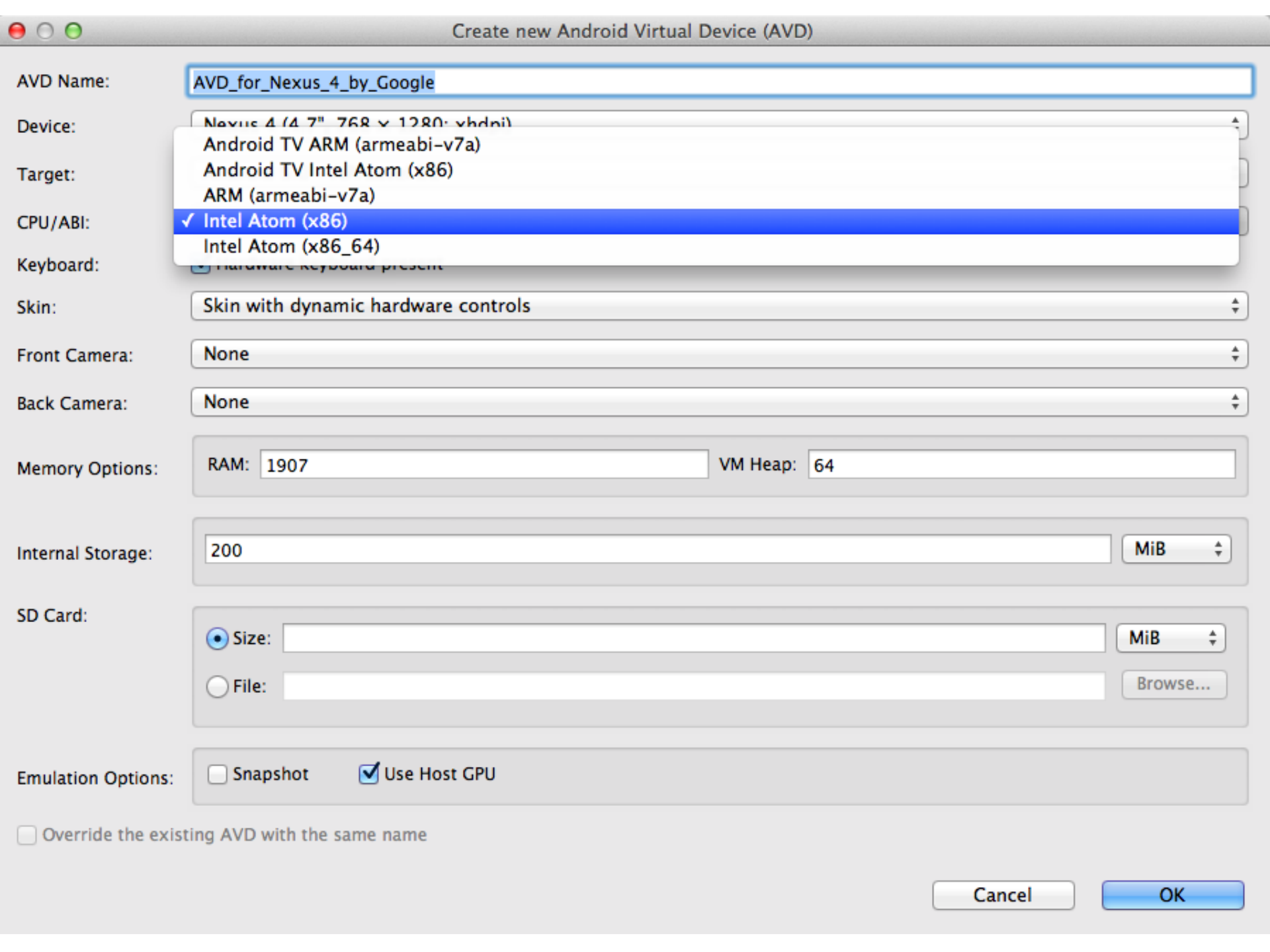


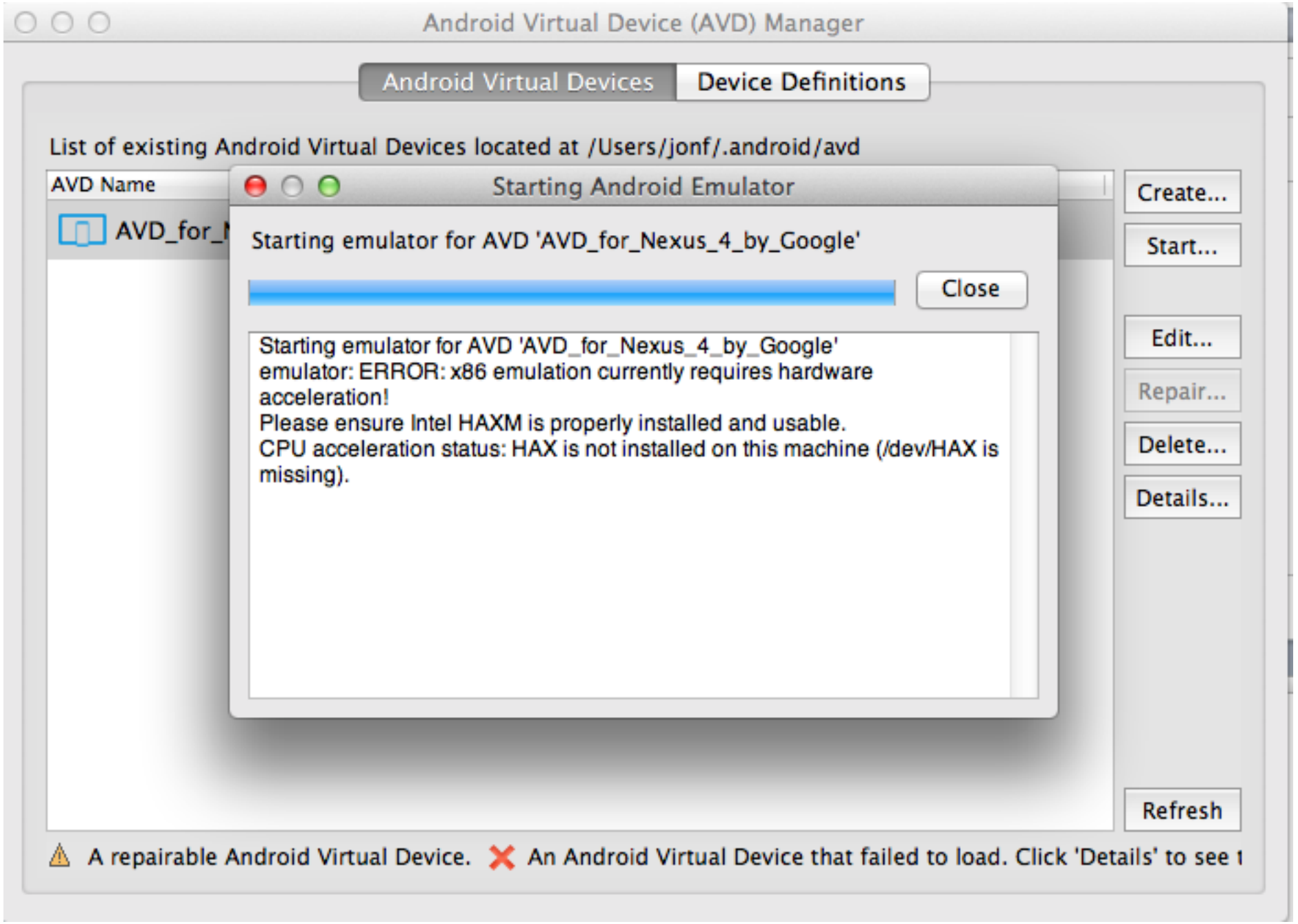

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\jonf>C:\Users\jonf\AppData\Local\Android\android-studio\sdk\tools\emulator.exe -avd AVD_for_Nexus_5_by_Google -netspeed full -netdelay none
C:\Users\jonf>emulator: ERROR: x86 emulation currently requires hardware acceleration!
Please ensure Intel HAXM is properly installed and usable.
CPU acceleration status: HAX kernel module is not installed!

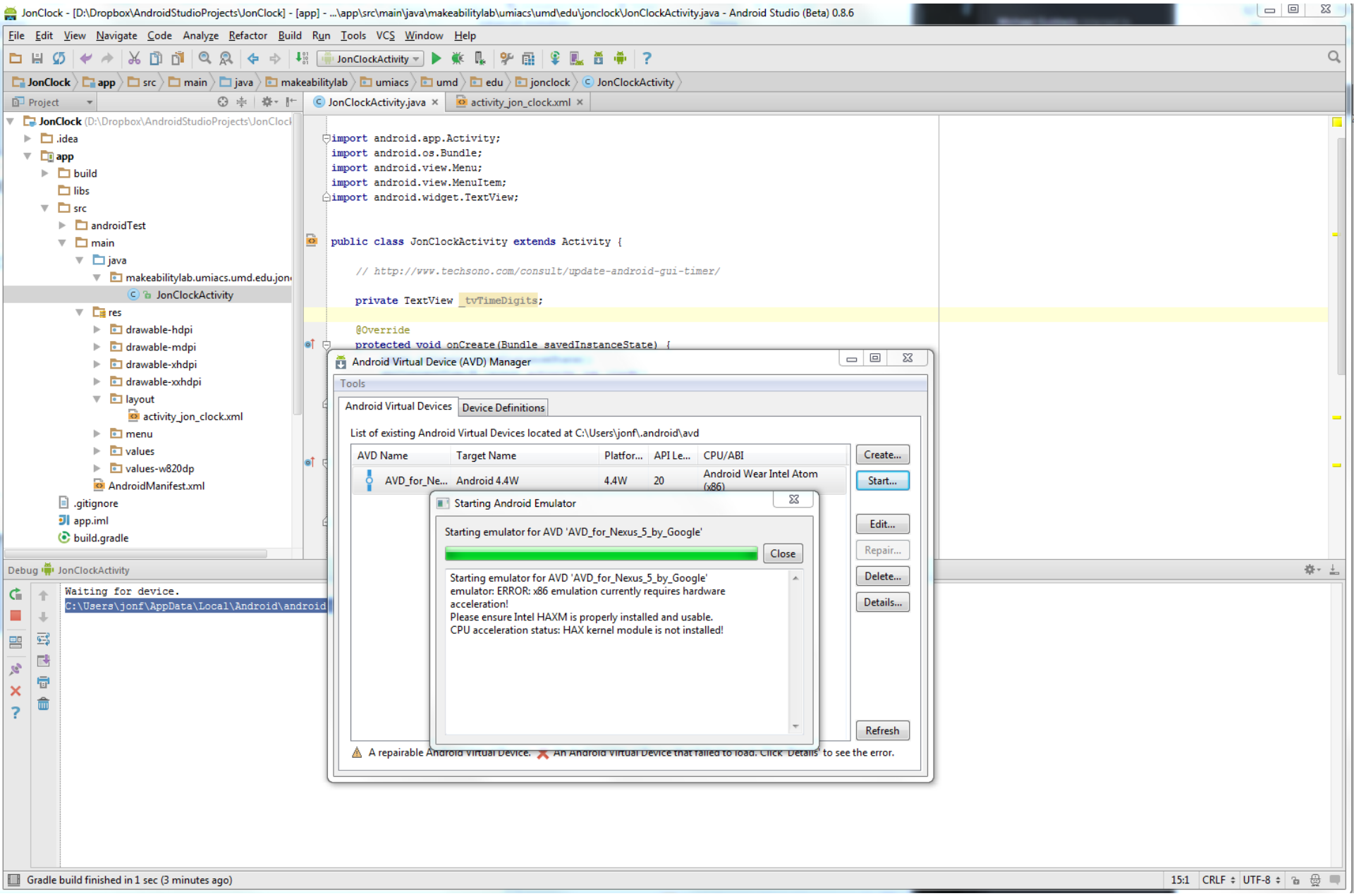

Create new Android Virtual Device (AVD)
AVD Name:
AVD_for_Nexus_4_by_Google
Device:
Nexus 4 (4.7", 768 × 1280: xhdpi)
Target:
Android 4.4W – API Level 20
CPU/ABI:
No system images installed for this target.
Keyboard:
Hardware keyboard present
Skin:
Front Camera:
None
Back Camera:
None
Memory Options:
RAM:
1907
VM Heap:
64
Internal Storage:
200
MiB
SD Card:
Size:
MiB
File:
Browse...
Emulation Options:
Snapshot
Use Host GPU
Override the existing AVD with the same name
No CPU/ABI system image available for this target
Cancel
OK

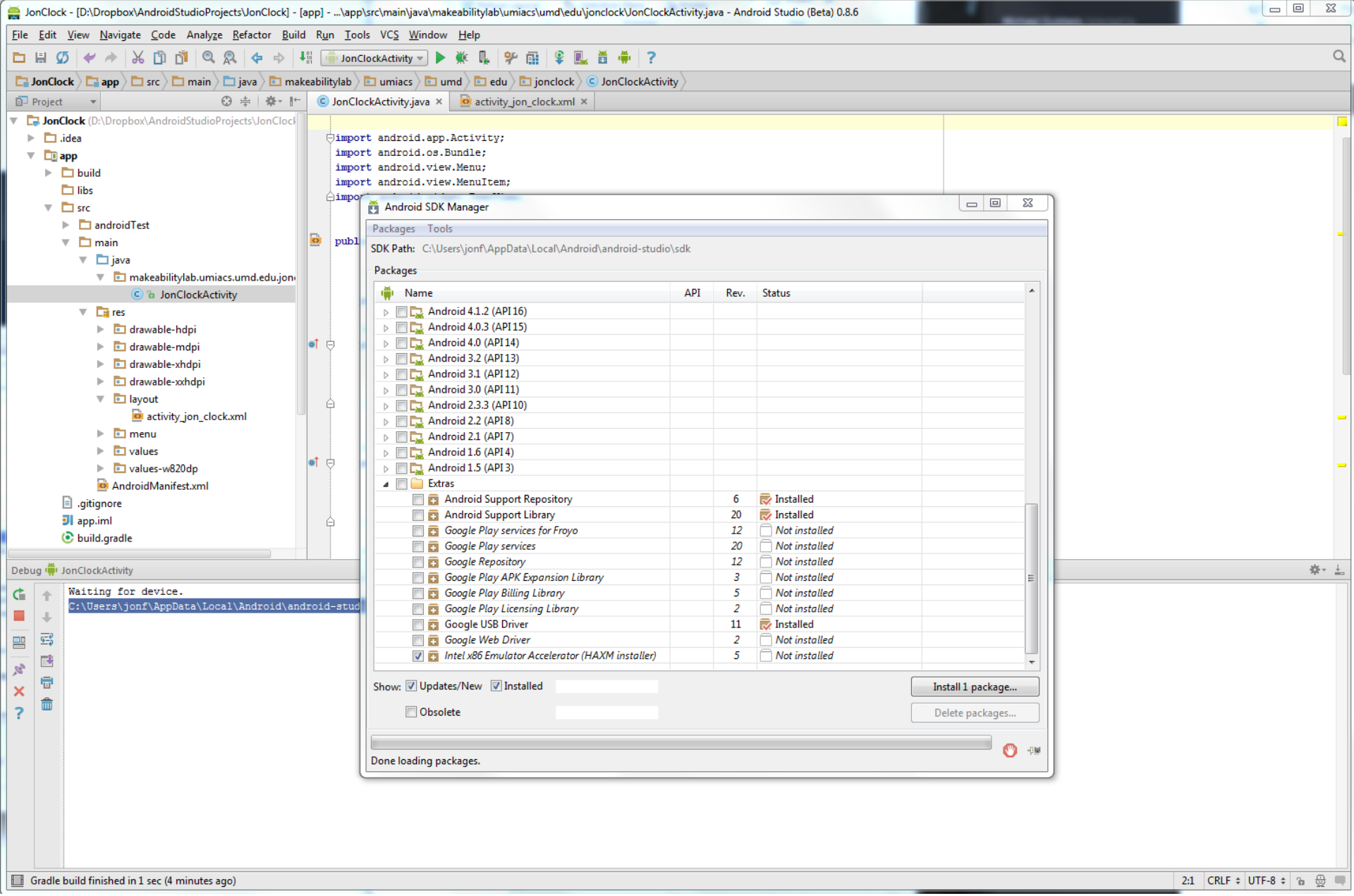

Android SDK Manager
SDK Path: /Applications/Android Studio.app/sdk
Packages
Name
API
Rev.
Status
Android TV Intel x86 Atom System Image 21 1 Not installed
ARM EABI v7a System Image 21 1 Not installed
Intel x86 Atom_64 System Image 21 1 Not installed
Intel x86 Atom System Image 21 1 Not installed
Google APIs ARM EABI v7a System Image 21 2 Not installed
Google APIs Intel x86 Atom_64 System Image 21 2 Not installed
Google APIs Intel x86 Atom System Image 21 2 Not installed
Google APIs 21 1 Not installed
Sources for Android SDK 21 1 Not installed
Android 4.4W (API 20)
SDK Platform 20 1 Installed
Samples for SDK 20 2 Not installed
Android Wear ARM EABI v7a System Image 20 2 Not installed
Android Wear Intel x86 Atom System Image 20 2 Not installed
Sources for Android SDK 20 1 Not installed
Android 4.4.2 (API 19)
Android 4.3.1 (API 18)
Android 4.2.2 (API 17)
Android 4.1.2 (API 16)
Android 4.0.3 (API 15)
Android 4.0 (API 14)
Android 3.2 (API 13)
Android 3.1 (API 12)
Android 3.0 (API 11)
Android 2.3.3 (API 10)
Android 2.2 (API 8)
Android 2.1 (API 7)
Android 1.6 (API 4)
Android 1.5 (API 3)
Show: Updates/New Installed Obsolete Select New or Updates
Install 16 packages...
Sort by: API level Repository Deselect All
Delete 3 packages...
Done loading packages.


Android SDK Manager
SDK Path: /Applications/Android Studio.app/sdk
Packages
Name
API
Rev.
Status
Android SDK Build-tools 18.1 Not installed
Android SDK Build-tools 18.0.1 Not installed
Android SDK Build-tools 17 Not installed
Android 5.0 (API 21)
Documentation for Android SDK 21 1 Not installed
SDK Platform 21 1 Not installed
Android TV ARM EABI v7a System Image 21 1 Not installed
Android TV Intel x86 Atom System Image 21 1 Not installed
ARM EABI v7a System Image 21 1 Not installed
Intel x86 Atom_64 System Image 21 1 Not installed
Intel x86 Atom System Image 21 1 Not installed
Google APIs ARM EABI v7a System Image 21 2 Not installed
Google APIs Intel x86 Atom_64 System Image 21 2 Not installed
Google APIs Intel x86 Atom System Image 21 2 Not installed
Google APIs 21 1 Not installed
Sources for Android SDK 21 1 Not installed
Android 4.4W (API 20)
SDK Platform 20 1 Installed
Samples for SDK 20 2 Not installed
Android Wear ARM EABI v7a System Image 20 2 Not installed
Android Wear Intel x86 Atom System Image 20 2 Not installed
Sources for Android SDK 20 1 Not installed
Android 4.4.2 (API 19)
Android 4.3.1 (API 18)
Android 4.2.2 (API 17)
Android 4.1.2 (API 16)
Android 4.0.3 (API 15)
Android 4.0 (API 14)
Android 3.2 (API 13)
Show: Updates/New Installed Obsolete Select New or Updates
Install 20 packages...
Sort by: API level Repository Deselect All
Delete 4 packages...
Done loading packages.


Create new Android Virtual Device (AVD)
AVD Name:
AVD_for_Nexus_4_by_Google
Device:
Target:
CPU/ABI:
Keyboard:
Skin:
Front Camera:
Back Camera:
Memory Options:
Internal Storage:
SD Card:
Emulation Options:
Override the existing AVD with the same name
Android TV ARM (armeabi-v7a)
Android TV Intel Atom (x86)
ARM (armeabi-v7a)
Intel Atom (x86)
Intel Atom (x86_64)
Skin with dynamic hardware controls
None
None
RAM:
1907
VM Heap:
64
200
MiB
Size:
MiB
File:
Browse...
Snapshot
Use Host GPU
Cancel
OK


Android Virtual Device (AVD) Manager
Android Virtual Devices
Device Definitions
List of existing Android Virtual Devices located at /Users/jonf/.android/avd
AVD Name
AVD_for_N
Create...
Start...
Edit...
Repair...
Delete...
Details...
Refresh
A repairable Android Virtual Device.
An Android Virtual Device that failed to load. Click 'Details' to see t
Starting Android Emulator
Starting emulator for AVD 'AVD_for_Nexus_4_by_Google'
Close
Starting emulator for AVD 'AVD_for_Nexus_4_by_Google'
emulator: ERROR: x86 emulation currently requires hardware acceleration!
Please ensure Intel HAXM is properly installed and usable.
CPU acceleration status: HAX is not installed on this machine (/dev/HAX is missing).


JonClock - [D:\Dropbox\AndroidStudioProjects\JonClock] - [app] - ...\app\src\main\java\makeabilitylab\umiacs\umd\edu\jonclock\JonClockActivity.java - Android Studio (Beta) 0.8.6
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
JonClockActivity
JonClock app src main java makeabilitylab umiacs umd edu jonclock JonClockActivity
Project
JonClock (D:\Dropbox\AndroidStudioProjects\JonClock)
.idea
app
build
libs
src
androidTest
main
java
makeabilitylab.umiacs.umd.edu.jon
JonClockActivity
res
drawable-hdpi
drawable-mdpi
drawable-xhdpi
drawable-xxhdpi
layout
activity_jon_clock.xml
menu
values
values-w820dp
AndroidManifest.xml
.gitignore
app.iml
build.gradle
JonClockActivity.java
activity_jon_clock.xml
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
public class JonClockActivity extends Activity {
// http://www.techsono.com/consult/update-android-gui-timer/
private TextView tvTimeDigits;
@Override
protected void onCreate(Bundle savedInstanceState) {
Android Virtual Device (AVD) Manager
Tools
Android Virtual Devices
Device Definitions
List of existing Android Virtual Devices located at C:\Users\jonf\.android\avd
AVD Name
Target Name
Platfor...
API Le...
CPU/ABI
AVD_for_Ne...
Android 4.4W
4.4W
20
Android Wear Intel Atom (x86)
Create...
Start...
Edit...
Repair...
Delete...
Details...
Refresh
Starting Android Emulator
Starting emulator for AVD 'AVD_for_Nexus_5_by_Google'
Close
Starting emulator for AVD 'AVD_for_Nexus_5_by_Google'
emulator: ERROR: x86 emulation currently requires hardware acceleration!
Please ensure Intel HAXM is properly installed and usable.
CPU acceleration status: HAX kernel module is not installed!
A repairable Android Virtual Device.
An Android Virtual Device that failed to load. Click 'Details' to see the error.
Debug JonClockActivity
Waiting for device.
C:\Users\jonf\AppData\Local\Android\android
Gradle build finished in 1 sec (3 minutes ago)
15:1
CRLF
UTF-8


JonClock - [D:\Dropbox\AndroidStudioProjects\JonClock] - [app] - ...\app\src\main\java\makeabilitylab\umiacs\umd\edu\jonclock\JonClockActivity.java - Android Studio (Beta) 0.8.6
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
JonClockActivity
JonClock app src main java makeabilitylab umiacs umd edu jonclock JonClockActivity
Project
JonClock (D:\Dropbox\AndroidStudioProjects\JonClock)
.idea
app
build
libs
src
androidTest
main
java
makeabilitylab.umiacs.umd.edu.jonclock
JonClockActivity
res
drawable-hdpi
drawable-mdpi
drawable-xhdpi
drawable-xxhdpi
layout
activity_jon_clock.xml
menu
values
values-w820dp
AndroidManifest.xml
.gitignore
app.iml
build.gradle
JonClockActivity.java
activity_jon_clock.xml
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
Android SDK Manager
Packages Tools
SDK Path: C:\Users\jonf\AppData\Local\Android\android-studio\sdk
Packages
Name API Rev. Status
Android 4.1.2 (API 16)
Android 4.0.3 (API 15)
Android 4.0 (API 14)
Android 3.2 (API 13)
Android 3.1 (API 12)
Android 3.0 (API 11)
Android 2.3.3 (API 10)
Android 2.2 (API 8)
Android 2.1 (API 7)
Android 1.6 (API 4)
Android 1.5 (API 3)
Extras
Android Support Repository 6 Installed
Android Support Library 20 Installed
Google Play services for Froyo 12 Not installed
Google Play services 20 Not installed
Google Repository 12 Not installed
Google Play APK Expansion Library 3 Not installed
Google Play Billing Library 5 Not installed
Google Play Licensing Library 2 Not installed
Google USB Driver 11 Installed
Google Web Driver 2 Not installed
Intel x86 Emulator Accelerator (HAXM installer) 5 Not installed
Show: Updates/New Installed
Obsolete
Install 1 package...
Delete packages...
Done loading packages.
Debug JonClockActivity
Waiting for device.
C:\Users\jonf\AppData\Local\Android\android-stud
Gradle build finished in 1 sec (4 minutes ago)
2:1 CRLF UTF-8

Intel® Hardware Accelerated Execution Manager

Submitted by HAOREN J. (Intel) on Wed, 11/27/2013 - 08:44

Last Updated April 15, 2014

The Intel Hardware Accelerated Execution Manager (Intel® HAXM) is a hardware-assisted virtualization engine (hypervisor) that uses Intel Virtualization Technology (Intel® VT) to speed up Android app emulation on a host machine. In combination with Android x86 emulator images provided by Intel and the official Android SDK Manager, HAXM allows for faster Android emulation on Intel VT enabled systems.

The following platforms are supported by the Intel HAXM:

Microsoft Windows*
Windows 8 and 8.1 (32/64-bit), Windows 7 (32/64-bit), Windows Vista* (32/64-bit)

Installation Guide & System Requirements - Windows

haxm-windows_r04.zip (1.0.8)

Description:
System Driver
(Apr. 15, 2014)

Size: 1.93MB

Checksums:
(MD5)
dbec9d4145a2a7acdf19cb10eb7a9539
(SHA-1)
cc72b38962fc53823f969d4fb9155f9efa3558b8

Forum >
Intel Hardware Accelerated Execution Manager (HAXM) >

In-Class Exercises

The Debug Monitor is where you can find the complete set of DDMS tools for profiling your app, controlling device behaviors, and more. It also includes the Hierarchy Viewer tools to help optimize your layouts.

Keyboard Commands

The following tables list keyboard shortcuts for common operations.

> **Note:** If you're using Mac OS X, update your keymap to use the Mac OS X 10.5+ version keymaps under **Android Studio > Preferences > Keymap**.

Table 1. Programming key commands

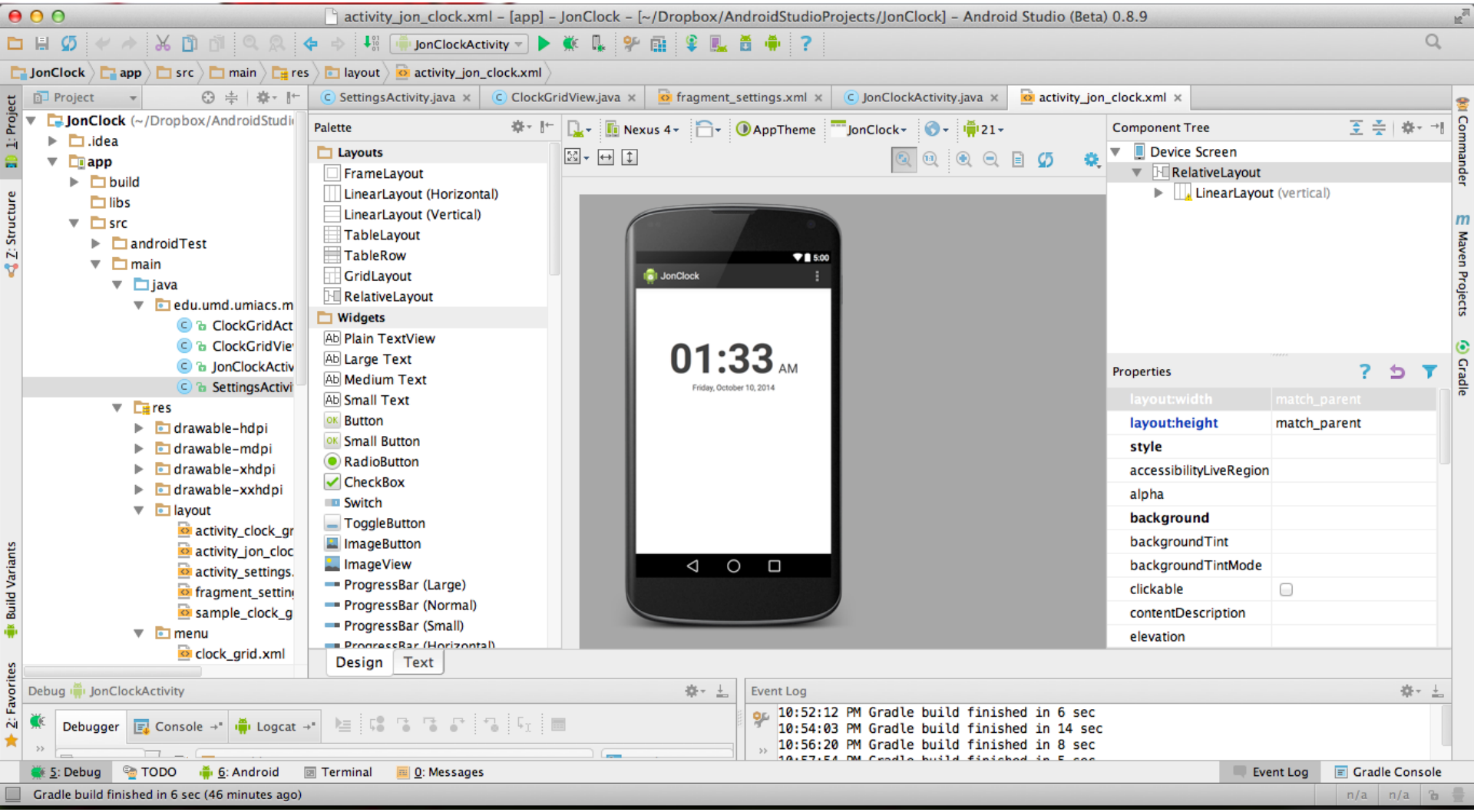
Action	Android Studio Key Command
Command look-up (autocomplete command name)	CTRL + SHIFT + A
Project quick fix	ALT + ENTER
Reformat code	CTRL + ALT + L (Win) OPTION + CMD + L (Mac)
Show docs for selected API	CTRL + Q (Win) F1 (Mac)
Show parameters for selected method	CTRL + P
Generate method	ALT + Insert (Win) CMD + N (Mac)
Jump to source	F4 (Win) CMD + down-arrow (Mac)
Delete line	CTRL + Y (Win) CMD + Backspace (Mac)
Search by symbol name	CTRL + ALT + SHIFT + N (Win) OPTION + CMD + O (Mac)

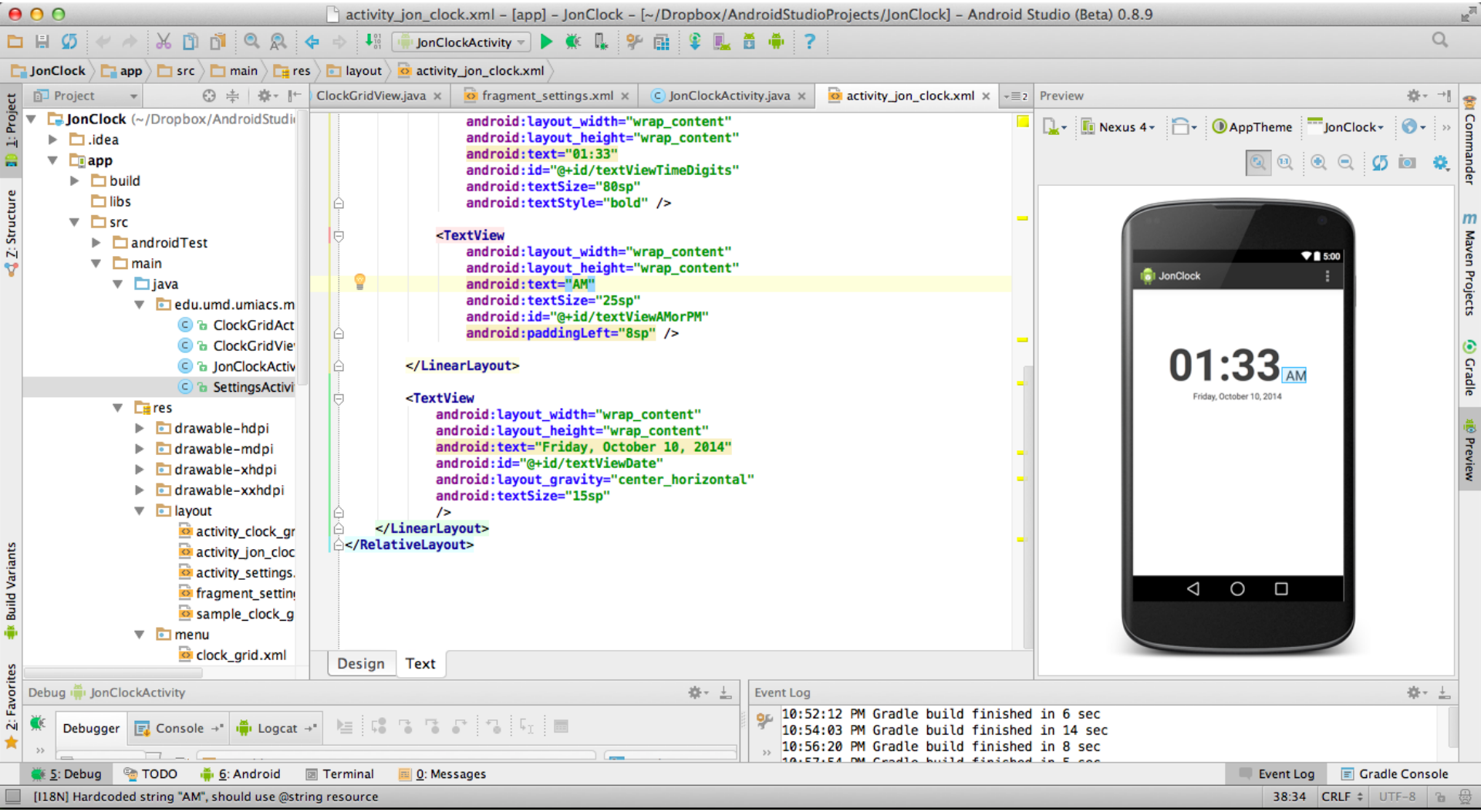
Table 2. Project and editor key commands

Action	Android Studio Key Command
Build	CTRL + F9 (Win) CMD + F9 (Mac)
Build and run	SHIFT + F10 (Win) CTRL + R (Mac)
Toggle project visibility	ALT + 1 (Win) CMD + 1 (Mac)
Navigate open tabs	ALT + left-arrow; ALT + right-arrow (Win)

building

android uis


activity_jon_clock.xml – [app] – JonClock – [~/Dropbox/AndroidStudioProjects/JonClock] – Android Studio (Beta) 0.8.9
JonClockActivity
JonClock
app
src
main
res
layout
activity_jon_clock.xml
Project
SettingsActivity.java
ClockGridView.java
fragment_settings.xml
JonClockActivity.java
activity_jon_clock.xml
Palette
Layouts
FrameLayout
LinearLayout (Horizontal)
LinearLayout (Vertical)
TableLayout
TableRow
GridLayout
RelativeLayout
Widgets
Plain TextView
Large Text
Medium Text
Small Text
Button
Small Button
RadioButton
CheckBox
Switch
ToggleButton
ImageButton
ImageView
ProgressBar (Large)
ProgressBar (Normal)
ProgressBar (Small)
Nexus 4
AppTheme
JonClock
21
01:33 AM
Friday, October 10, 2014
Component Tree
Device Screen
RelativeLayout
LinearLayout (vertical)
Properties
layout:width
match_parent
layout:height
match_parent
style
accessibilityLiveRegion
alpha
background
backgroundTint
backgroundTintMode
clickable
contentDescription
elevation
Design
Text
Debug
JonClockActivity
Debugger
Console
Logcat
Event Log
10:52:12 PM Gradle build finished in 6 sec
10:54:03 PM Gradle build finished in 14 sec
10:56:20 PM Gradle build finished in 8 sec
5: Debug
TODO
6: Android
Terminal
0: Messages
Event Log
Gradle Console
Gradle build finished in 6 sec (46 minutes ago)


activity_jon_clock.xml – [app] – JonClock – [~/Dropbox/AndroidStudioProjects/JonClock] – Android Studio (Beta) 0.8.9
JonClockActivity
JonClock app src main res layout activity_jon_clock.xml
Project
JonClock (~/Dropbox/AndroidStudi
.idea
app
build
libs
src
androidTest
main
java
edu.umd.umiacs.m
ClockGridAct
ClockGridVie
JonClockActiv
SettingsActivi
res
drawable-hdpi
drawable-mdpi
drawable-xhdpi
drawable-xxhdpi
layout
activity_clock_gr
activity_jon_cloc
activity_settings.
fragment_settin
sample_clock_g
menu
clock_grid.xml
ClockGridView.java
fragment_settings.xml
JonClockActivity.java
activity_jon_clock.xml
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="01:33"
android:id="@+id/textViewTimeDigits"
android:textSize="80sp"
android:textStyle="bold" />
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="AM"
android:textSize="25sp"
android:id="@+id/textViewAMorPM"
android:paddingLeft="8sp" />
</LinearLayout>
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Friday, October 10, 2014"
android:id="@+id/textViewDate"
android:layout_gravity="center_horizontal"
android:textSize="15sp"
/>
</LinearLayout>
</RelativeLayout>
Design
Text
Preview
Nexus 4
AppTheme
JonClock
5:00
JonClock
01:33 AM
Friday, October 10, 2014
Debug JonClockActivity
Debugger
Console
Logcat
Event Log
10:52:12 PM Gradle build finished in 6 sec
10:54:03 PM Gradle build finished in 14 sec
10:56:20 PM Gradle build finished in 8 sec
5: Debug
TODO
6: Android
Terminal
0: Messages
Event Log
Gradle Console
[I18N] Hardcoded string "AM", should use @string resource
38:34
CRLF
UTF-8

Android Graphical UI Builder

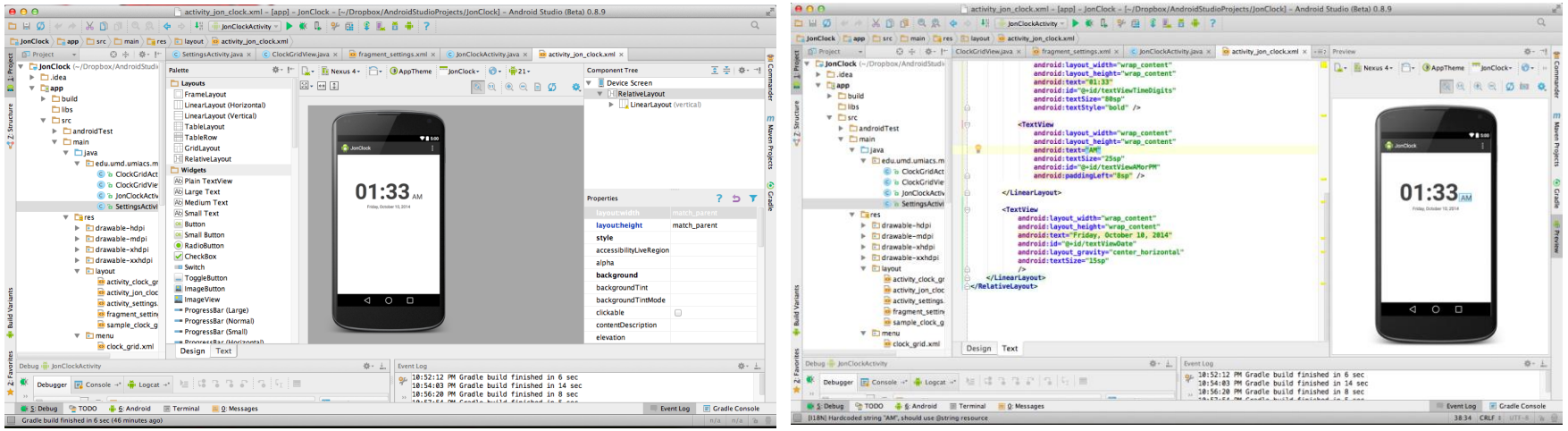


Figure 1. The Android graphical UI builder (WYSIWYG editor) allows you to build your interfaces visually—via a drag-and-drop interface—paired with an XML-based declarative syntax.

Android Custom Views

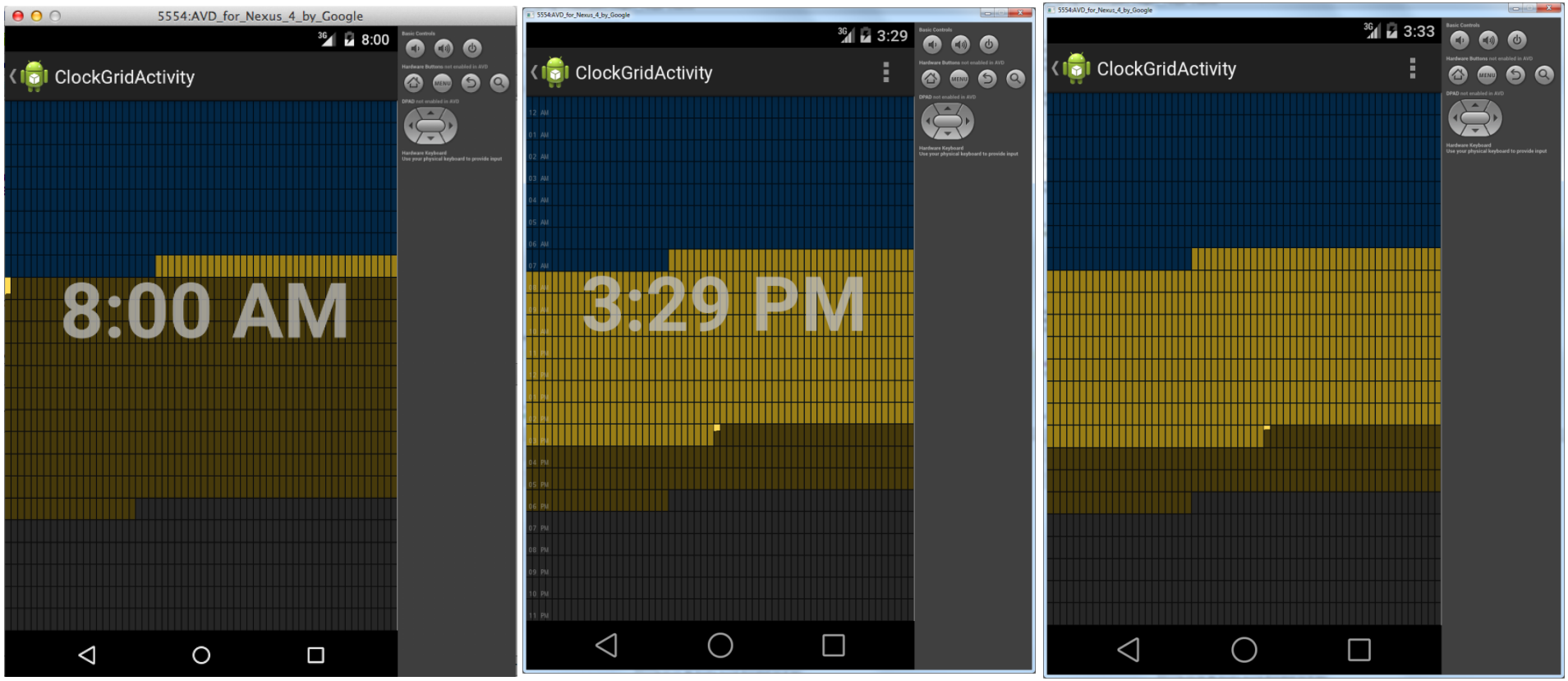
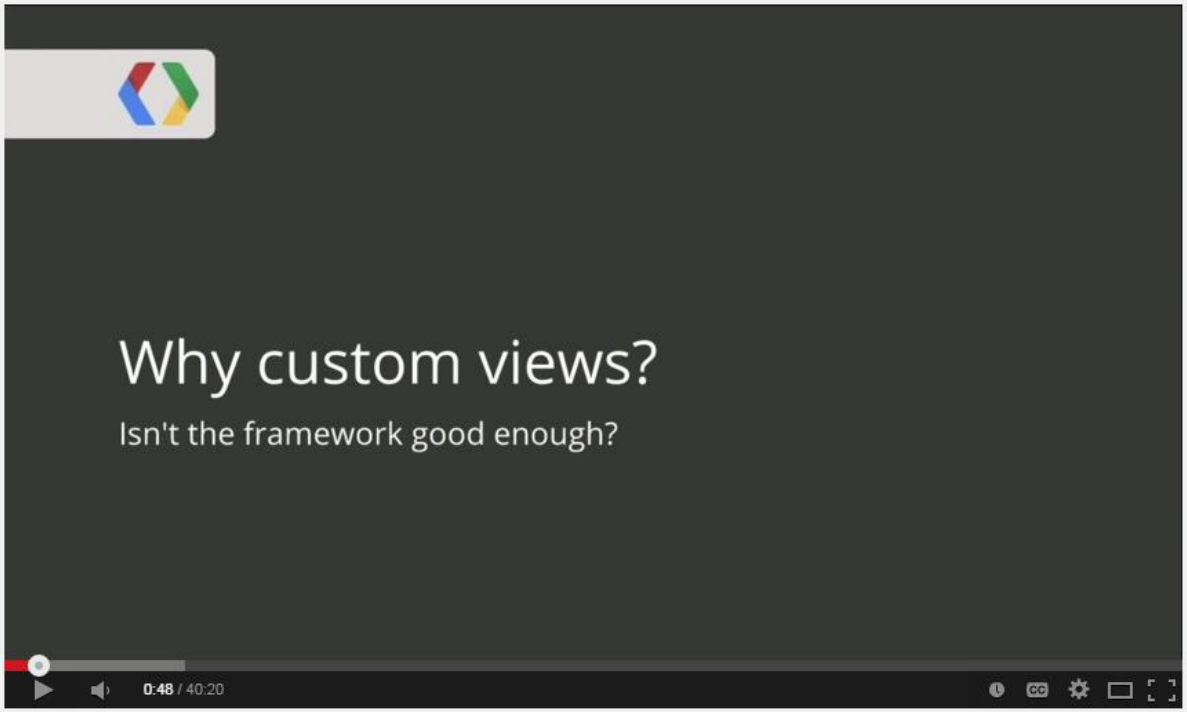


Figure 2. Many applications on your smartphone are built using custom views where the designer/programmer implements their own drawing code (*e.g.,* using DrawLine, DrawRect, etc.). In this case, I've extended the View class to create a daily progress bar visualization of time. Each block represents one minute, yellow blocks are daylight (as obtained via a web service API), and the current minute block fills progressively. My intention here is to provoke the user into rethinking how they spend their finite time—*i.e.,* there are only so many "minute blocks" in a day.



Google I/O 2013 - Writing Custom Views for Android

Google Developers

Subscribe 608,465

56,520

Add to Share More

242 7

Published on May 16, 2013
Adam Powell, Romain Guy

While the Android framework provides a number of layouts and prebuilt composable pieces for building UIs, targeted optimization or building

SHOW MORE

ALL COMMENTS (17)

Share your thoughts

Top comments


55:17
Google I/O 2009 - The Myth Genius Programmer
by Google Developers
525,416 views


41:31
Google I/O 2014 - Unlock the era of UI development with
by Google Developers
53,175 views


37:32
Google I/O 2013 - When And Meets Maps
by Google Developers
20,489 views


40:14
Google I/O 2013 - Android D for UI Developers
by Google Developers
172,957 views

1:02:53
Taming Android UIs with Eri of Square
by NewCircle Training
16,234 views

1:16:22
Tutorial: Android Internals - a Custom ROM, Pt. 1 of 2
by NewCircle Training
160,111 views

Google I/O 2013 Android
by Google Developers


40:00
Google I/O 2013 - Design De in AngularJS
by Google Developers
114,369 views


45:03
Google I/O 2014 - Material de principles
by Google Developers
64,854 views


57:05
Learn about Android Graphi Animations from Google's A
by UserGroupsatGoogle
48,756 views


43:09
Google I/O 2012 - SQL vs No Battle of the Backends
by Google Developers
224,383 views


Google I/O 2013 - Volley: Ea Networking for Android
by Google Developers
93,690 views

Navigation with Back and Up

PREVIOUS NEXT

Consistent navigation is an essential component of the overall user experience. Few things frustrate users more than basic navigation that behaves in inconsistent and unexpected ways. Android 3.0 introduced significant changes to the global navigation behavior. Thoughtfully following the guidelines for Back and Up will make your app's navigation predictable and reliable for your users.

DEVELOPER DOCS
Implementing Effective Navigation

Android 2.3 and earlier relied upon the system *Back* button for supporting navigation within an app. With the introduction of action bars in Android 3.0, a second navigation mechanism appeared: the *Up* button, consisting of the app icon and a left-point caret.

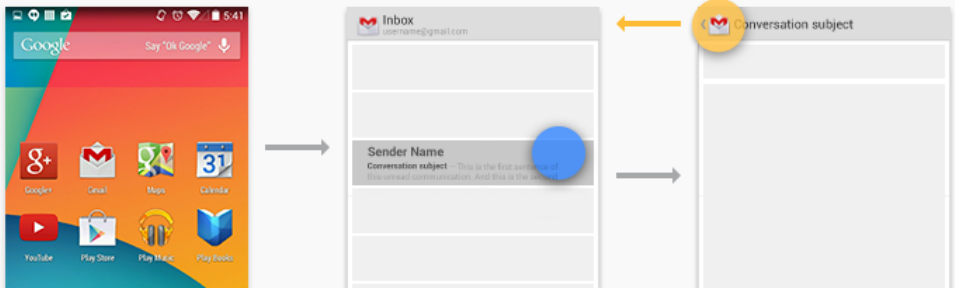
Up vs. Back

The Up button is used to navigate within an app based on the hierarchical relationships between screens. For instance, if screen A displays a list of items, and selecting an item leads to screen B (which presents that item in more detail), then screen B should offer an Up button that returns to screen A.

If a screen is the topmost one in an app (that is, the app's home), it should not present an Up button.

The system Back button is used to navigate, in reverse chronological order, through the history of screens the user has recently worked with. It is generally based on the temporal relationships between screens, rather than the app's hierarchy.

When the previously viewed screen is also the hierarchical parent of the current screen, pressing the Back button has the same result as pressing an Up button—this is a common occurrence. However, unlike the Up button, which ensures the user remains within your app, the Back button can return the user to the Home screen, or even to a different app.



Providing Up Navigation

PREVIOUS | NEXT

THIS LESSON TEACHES YOU TO:

1. Specify the Parent Activity
2. Add Up Action
3. Navigate Up to Parent Activity

YOU SHOULD ALSO READ

- Providing Ancestral and Temporal Navigation
- Tasks and Back Stack
- Android Design: Navigation

TRY IT OUT

Download the sample app

EffectiveNavigation.zip

All screens in your app that are not the main entrance to your app (the "home" screen) should offer the user a way to navigate to the logical parent screen in the app's hierarchy by pressing the *Up* button in the action bar. This lesson shows you how to properly implement this behavior.

> **Up Navigation Design**
>
> The concepts and principles for *Up* navigation are described in Designing Effective Navigation and the Navigation design guide.

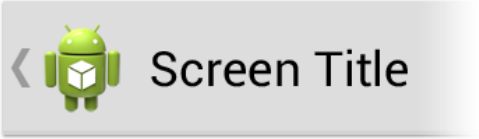


Figure 1. The *Up* button in the action bar.

Specify the Parent Activity

To implement *Up* navigation, the first step is to declare which activity is the appropriate parent for each activity. Doing so allows the system to facilitate navigation patterns such as *Up* because the system can determine the logical parent activity from the manifest file.

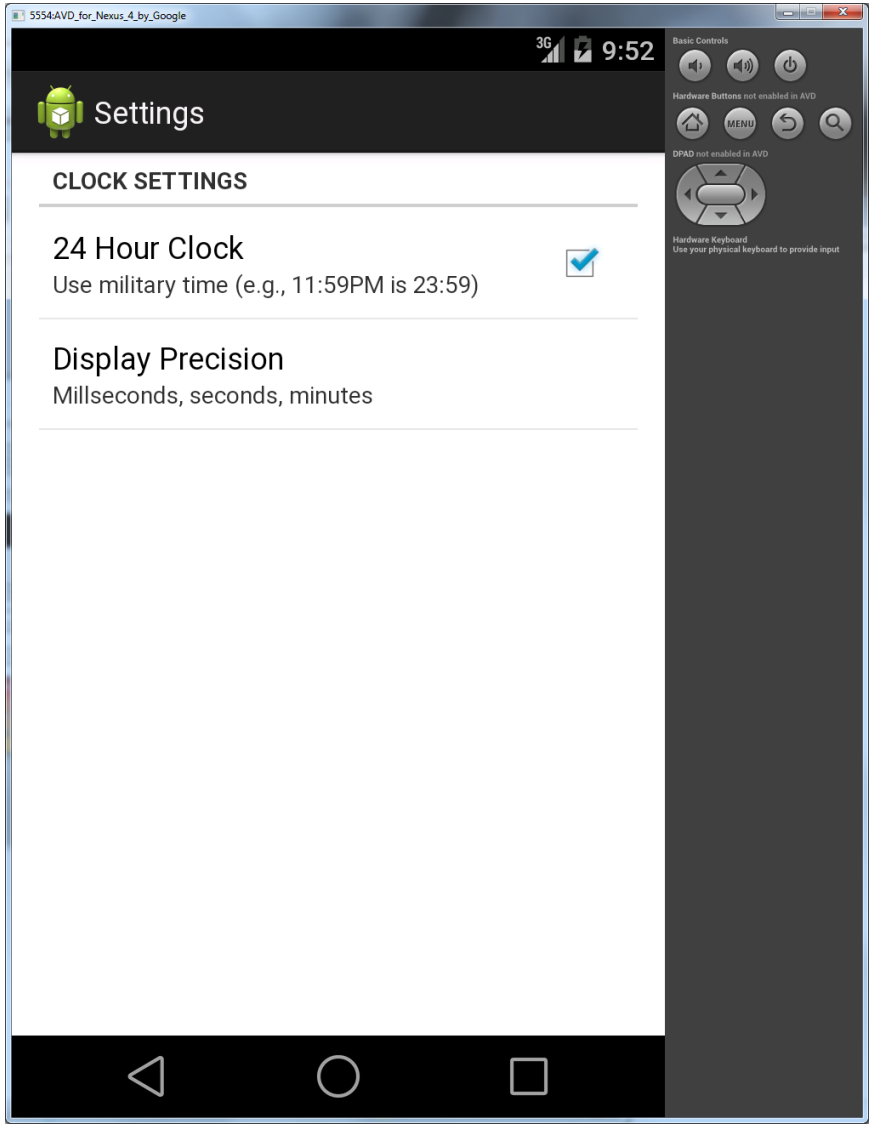
Beginning in Android 4.1 (API level 16), you can declare the logical parent of each activity by specifying the `android:parentActivityName` attribute in the `<activity>` element.

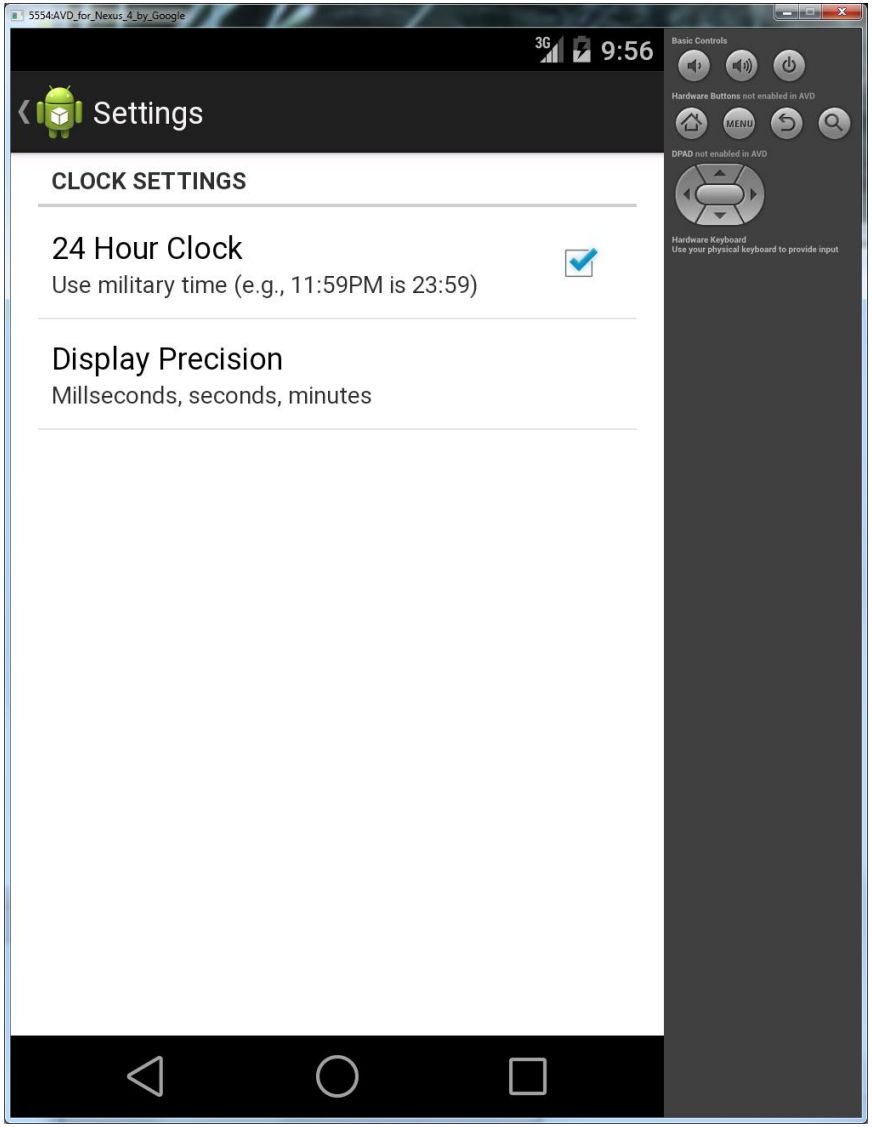
If your app supports Android 4.0 and lower, include the Support Library with your app and add a `<meta-data>` element inside the `<activity>`. Then specify the parent activity as the value for `android.support.PARENT_ACTIVITY`, matching the `android:parentActivityName` attribute.

For example:

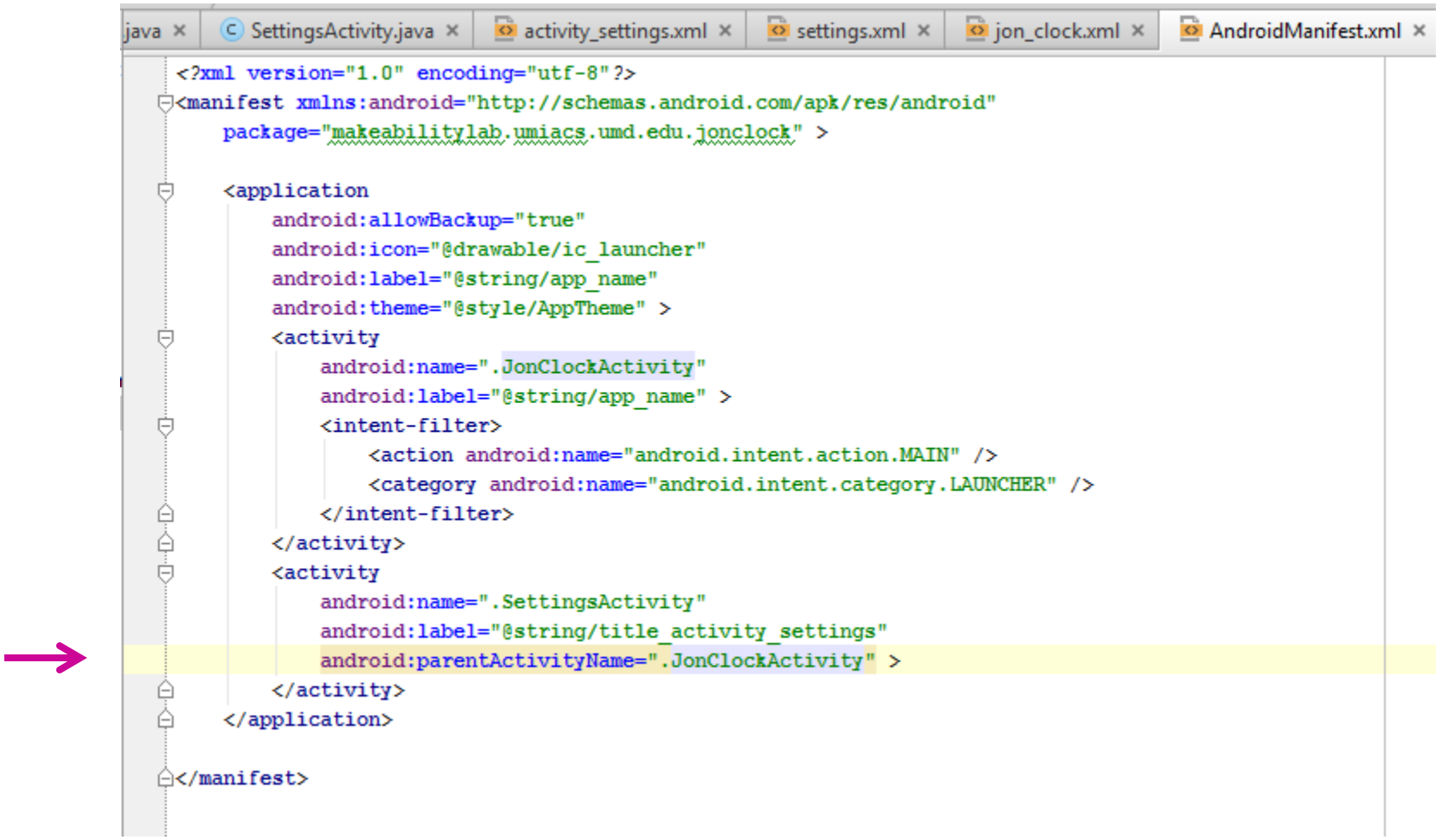
```
<application ... >
    ...
    <!-- The main/home activity (it has no parent activity) -->
    <activity
        android:name="com.example.myfirstapp.MainActivity" ...>
        ...
    </activity>
```

Up Icon Added Automatically





Up Icon Added Automatically

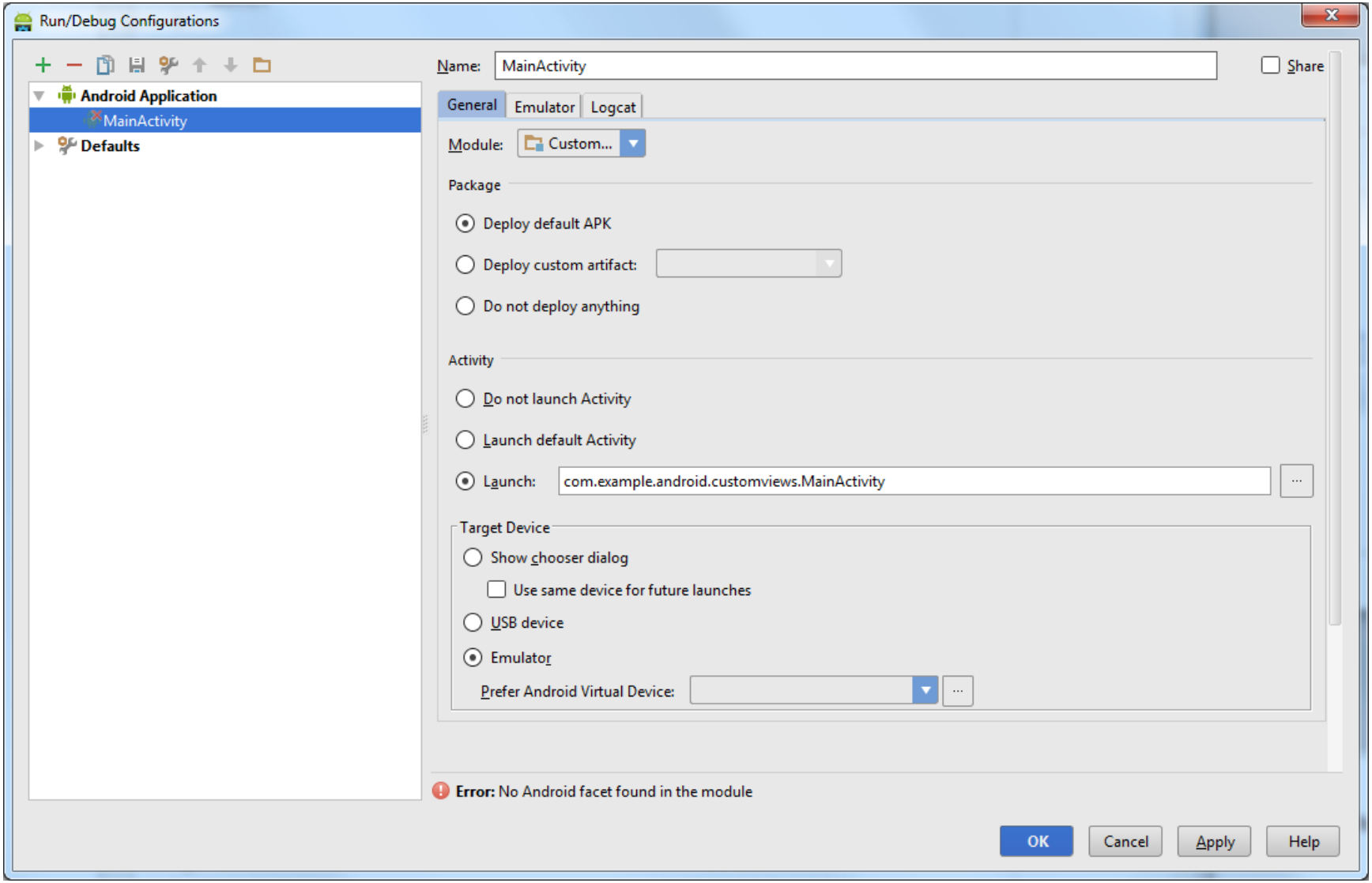


With this one-line definition, the "up" icon will show up automatically in the specified child activity; however, the icon will be non-functional (*i.e.,* nothing will happen when you press it).

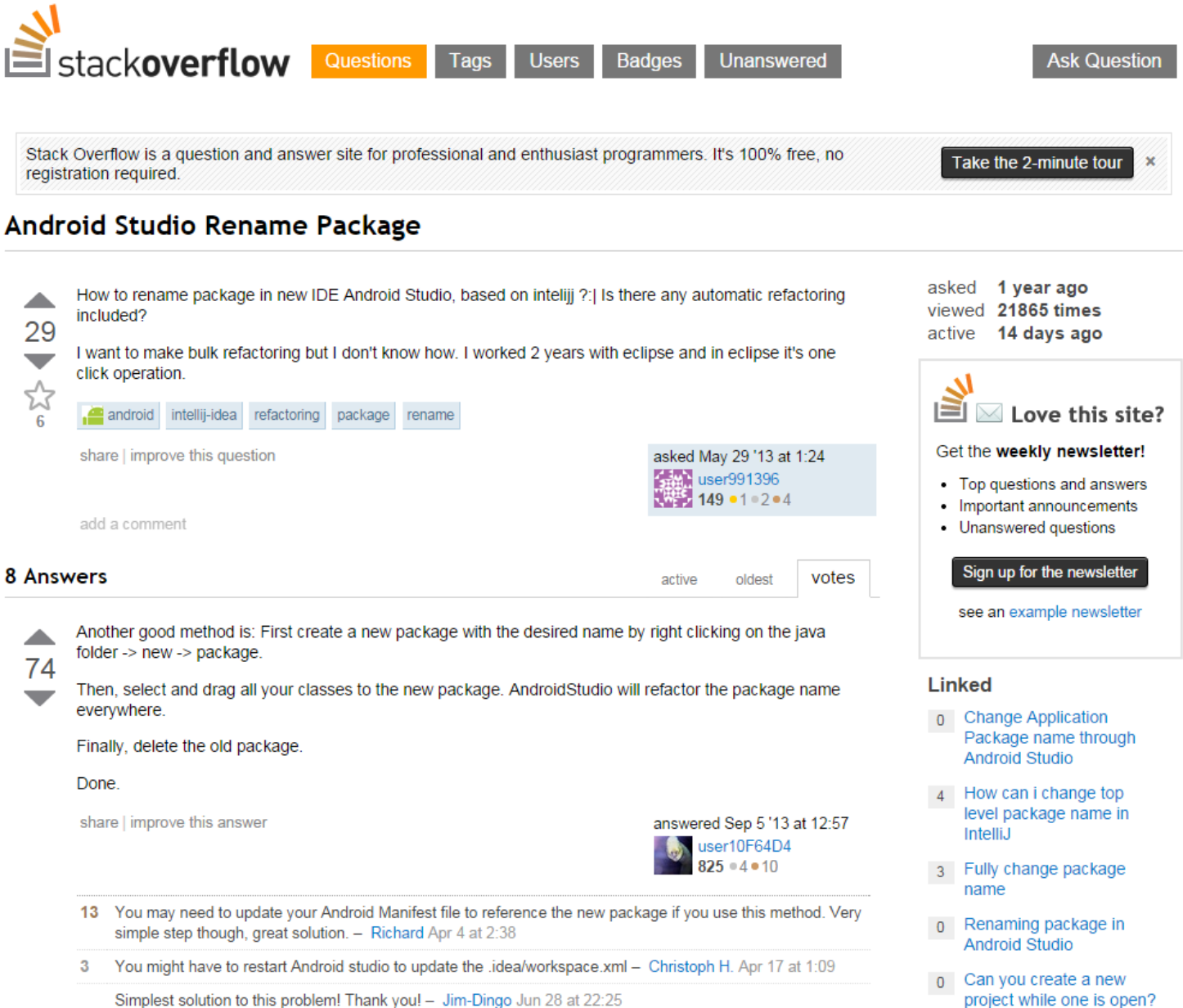
android

troubleshooting

No Android Facet Found



Android Studio Package Rename



stackoverflow Questions Tags Users Badges Unanswered Ask Question

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required. Take the 2-minute tour ×

Android Studio Rename Package

29

6

How to rename package in new IDE Android Studio, based on intelijj ?:| Is there any automatic refactoring included?

I want to make bulk refactoring but I don't know how. I worked 2 years with eclipse and in eclipse it's one click operation.

android intellij-idea refactoring package rename

share | improve this question

asked May 29 '13 at 1:24
user991396
149 ●1 ●2 ●4

add a comment

asked 1 year ago
viewed 21865 times
active 14 days ago

Love this site?

Get the **weekly newsletter!**

- Top questions and answers
- Important announcements
- Unanswered questions

Sign up for the newsletter

see an example newsletter

8 Answers

active oldest votes

74

Another good method is: First create a new package with the desired name by right clicking on the java folder -> new -> package.

Then, select and drag all your classes to the new package. AndroidStudio will refactor the package name everywhere.

Finally, delete the old package.

Done.

share | improve this answer

answered Sep 5 '13 at 12:57
user10F64D4
825 ●4 ●10

13 You may need to update your Android Manifest file to reference the new package if you use this method. Very simple step though, great solution. – Richard Apr 4 at 2:38

3 You might have to restart Android studio to update the .idea/workspace.xml – Christoph H. Apr 17 at 1:09

Simplest solution to this problem! Thank you! – Jim-Dingo Jun 28 at 22:25

Linked

0 Change Application Package name through Android Studio

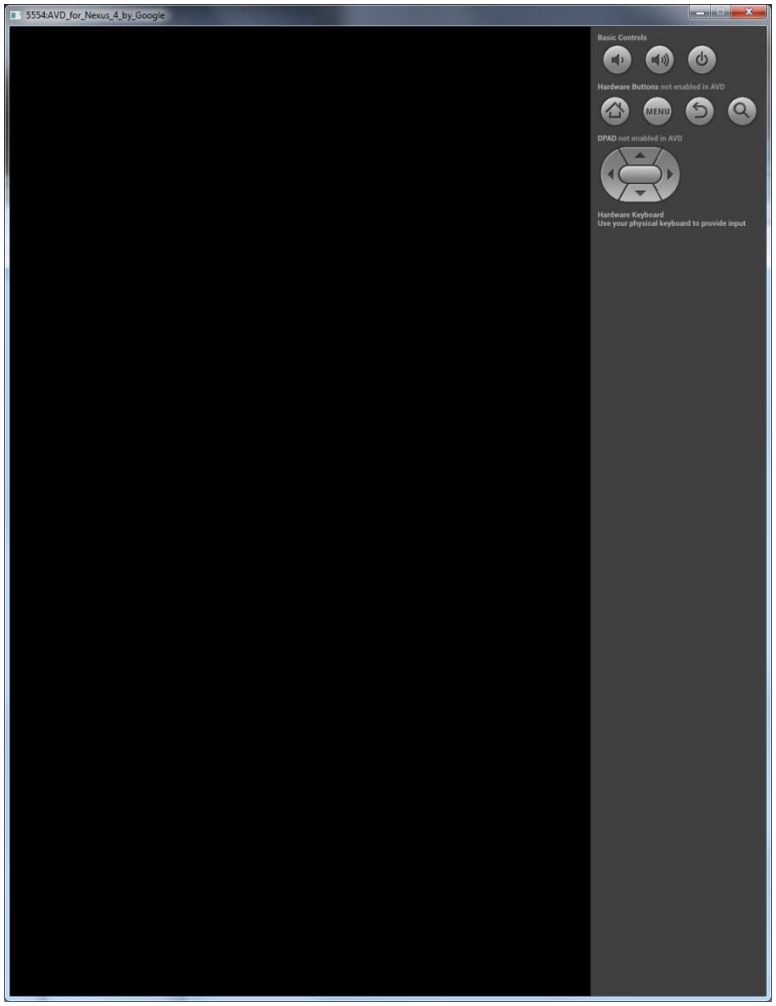
4 How can i change top level package name in IntelliJ

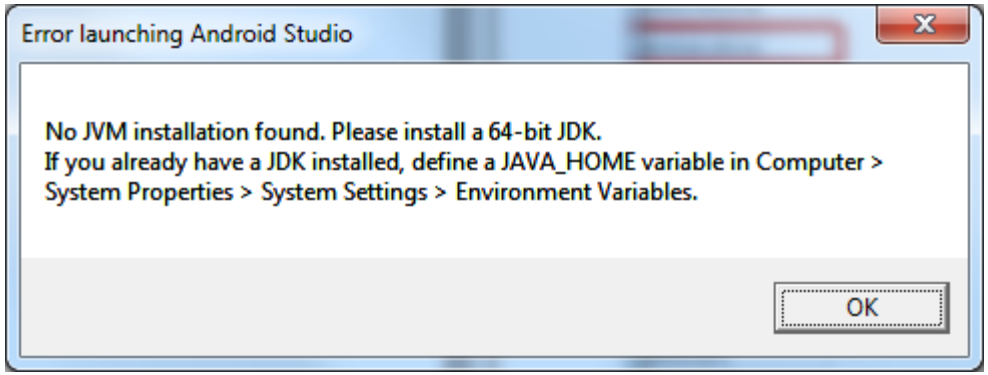
3 Fully change package name

0 Renaming package in Android Studio

0 Can you create a new project while one is open?

Blank Emulator Screen




Error launching Android Studio
No JVM installation found. Please install a 64-bit JDK.
If you already have a JDK installed, define a JAVA_HOME variable in Computer > System Properties > System Settings > Environment Variables.
OK

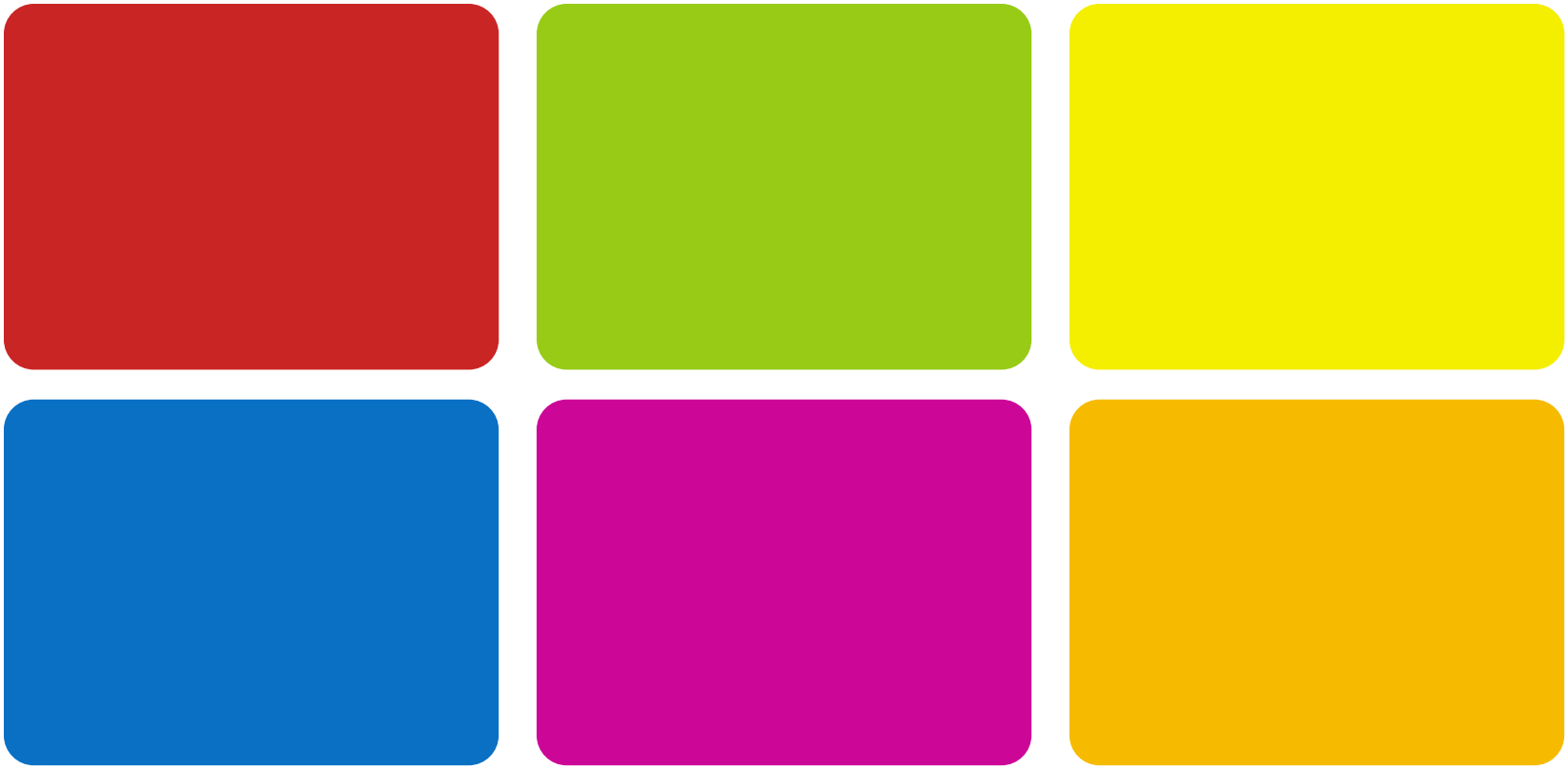
Debug JonClockActivity

Debugger | Console | Logcat

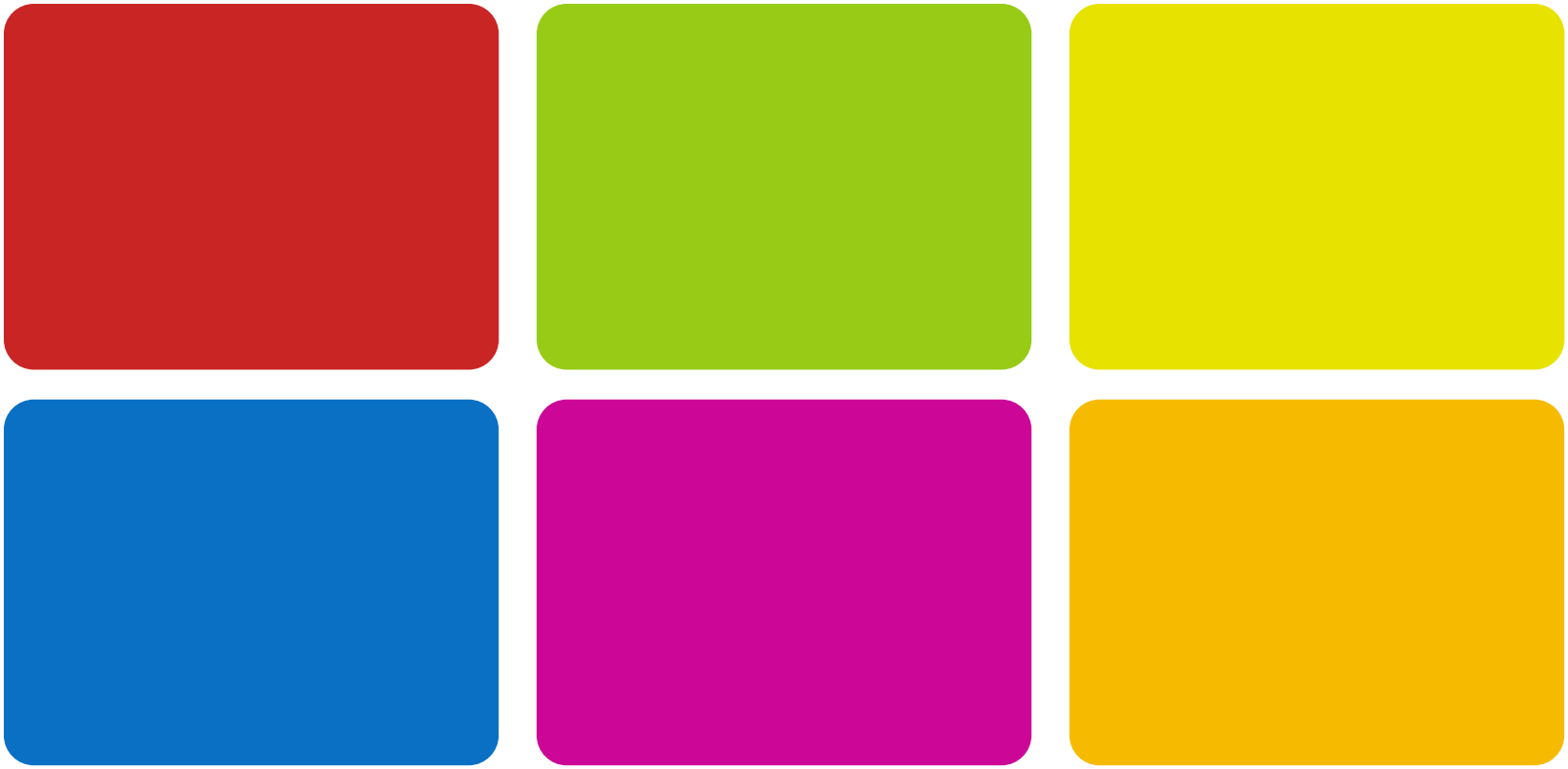
```
10-11 00:11:51.471    1775-1775/makeabilitylab.umiacs.umd.edu.jonclock I/System.out:  waiting for debugger to settle...
10-11 00:11:51.681    1775-1775/makeabilitylab.umiacs.umd.edu.jonclock I/System.out:  debugger has settled (1386)
10-11 00:11:51.711    1775-1775/makeabilitylab.umiacs.umd.edu.jonclock D/AndroidRuntime:  Shutting down VM
10-11 00:11:51.711    1775-1775/makeabilitylab.umiacs.umd.edu.jonclock W/dalvikvm:  threadid=1: thread exiting with uncaught exception (group=0xb0dfbce8)
10-11 00:11:51.711    1775-1775/makeabilitylab.umiacs.umd.edu.jonclock E/AndroidRuntime:  FATAL EXCEPTION: main
    Process: makeabilitylab.umiacs.umd.edu.jonclock, PID: 1775
    java.lang.RuntimeException: Unable to start activity ComponentInfo{makeabilitylab.umiacs.umd.edu.jonclock/makeabilitylab.umiacs.umd.edu.jonclock.JonClockActivity}: android.util.Andro
            at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2197)
            at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2258)
            at android.app.ActivityThread.access$800(ActivityThread.java:138)
            at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1209)
            at android.os.Handler.dispatchMessage(Handler.java:102)
            at android.os.Looper.loop(Looper.java:136)
            at android.app.ActivityThread.main(ActivityThread.java:5026)
            at java.lang.reflect.Method.invokeNative(Native Method) <1 internal calls>
            at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:777)
            at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:602)
            at dalvik.system.NativeStart.main(Native Method)
     Caused by: android.util.AndroidRuntimeException: You cannot combine swipe dismissal and the action bar.
            at com.android.internal.policy.impl.PhoneWindow.requestFeature(PhoneWindow.java:277)
            at com.android.internal.policy.impl.PhoneWindow.generateLayout(PhoneWindow.java:2889)
            at com.android.internal.policy.impl.PhoneWindow.installDecor(PhoneWindow.java:3154)
            at com.android.internal.policy.impl.PhoneWindow.setContentView(PhoneWindow.java:305)
            at android.app.Activity.setContentView(Activity.java:1930)
            at makeabilitylab.umiacs.umd.edu.jonclock.JonClockActivity.onCreate(JonClockActivity.java:14)
            at android.app.Activity.performCreate(Activity.java:5242)
            at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1087)
            at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2161)
            at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2258)
            at android.app.ActivityThread.access$800(ActivityThread.java:138)
            at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1209)
            at android.os.Handler.dispatchMessage(Handler.java:102)
            at android.os.Looper.loop(Looper.java:136)
            at android.app.ActivityThread.main(ActivityThread.java:5026)
            at java.lang.reflect.Method.invokeNative(Native Method)
            at java.lang.reflect.Method.invoke(Method.java:515) <3 more...>
10-11 00:11:51.721    1250-1261/system_process W/ActivityManager:  Force finishing activity makeabilitylab.umiacs.umd.edu.jonclock/.JonClockActivity
10-11 00:11:51.721    1250-1527/system_process W/InputMethodManagerService:  Window already focused, ignoring focus gain of: com.android.internal.view.IInputMethodClient$Stub$Proxy@b139e7
10-11 00:11:52.231    1448-1448/com.google.android.wearable.app E/HostWithRpcCallback:  Failed to send RPC
    com.google.android.wearable.gmsclient.WearableException: sendRpcAsync failed: Status{statusCode=unknown status code: 4000, resolution=null}
            at com.google.android.wearable.gmsclient.GoogleApiClientHelper.throwIfFailed(GoogleApiClientHelper.java:98)
            at com.google.android.wearable.gmsclient.MessageManager$1$1.getResult(MessageManager.java:81)
            at com.google.android.wearable.gmsclient.MessageManager$1$1.getResult(MessageManager.java:78)
            at com.google.android.clockwork.actions.WearableHostWithRpcCallback$SendRpcCallbackWithId.onSendRpc(WearableHostWithRpcCallback.java:292)
            at com.google.android.wearable.gmsclient.MessageManager$1.onResult(MessageManager.java:78)
            at com.google.android.wearable.gmsclient.MessageManager$1.onResult(MessageManager.java:75)
            at com.google.android.gms.common.api.a$c.b(Unknown Source)
            at com.google.android.gms.common.api.a$c.handleMessage(Unknown Source)
            at android.os.Handler.dispatchMessage(Handler.java:102)
            at android.os.Looper.loop(Looper.java:136)
            at android.app.ActivityThread.main(ActivityThread.java:5026)
            at java.lang.reflect.Method.invokeNative(Native Method) <1 internal calls>
            at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:777)
            at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:602)
            at dalvik.system.NativeStart.main(Native Method)
```

Find Your Project Team

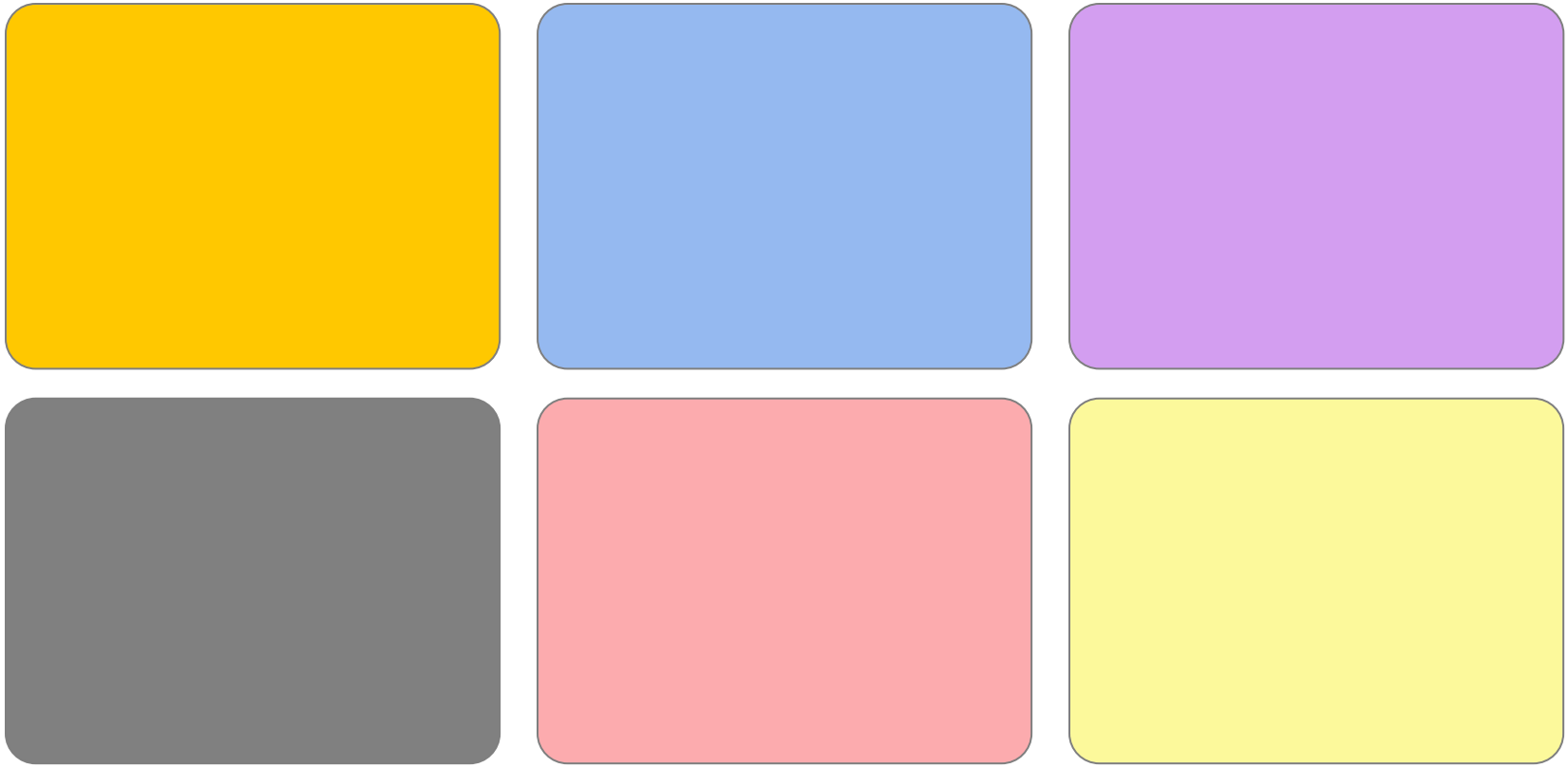
Let’s work on TA05



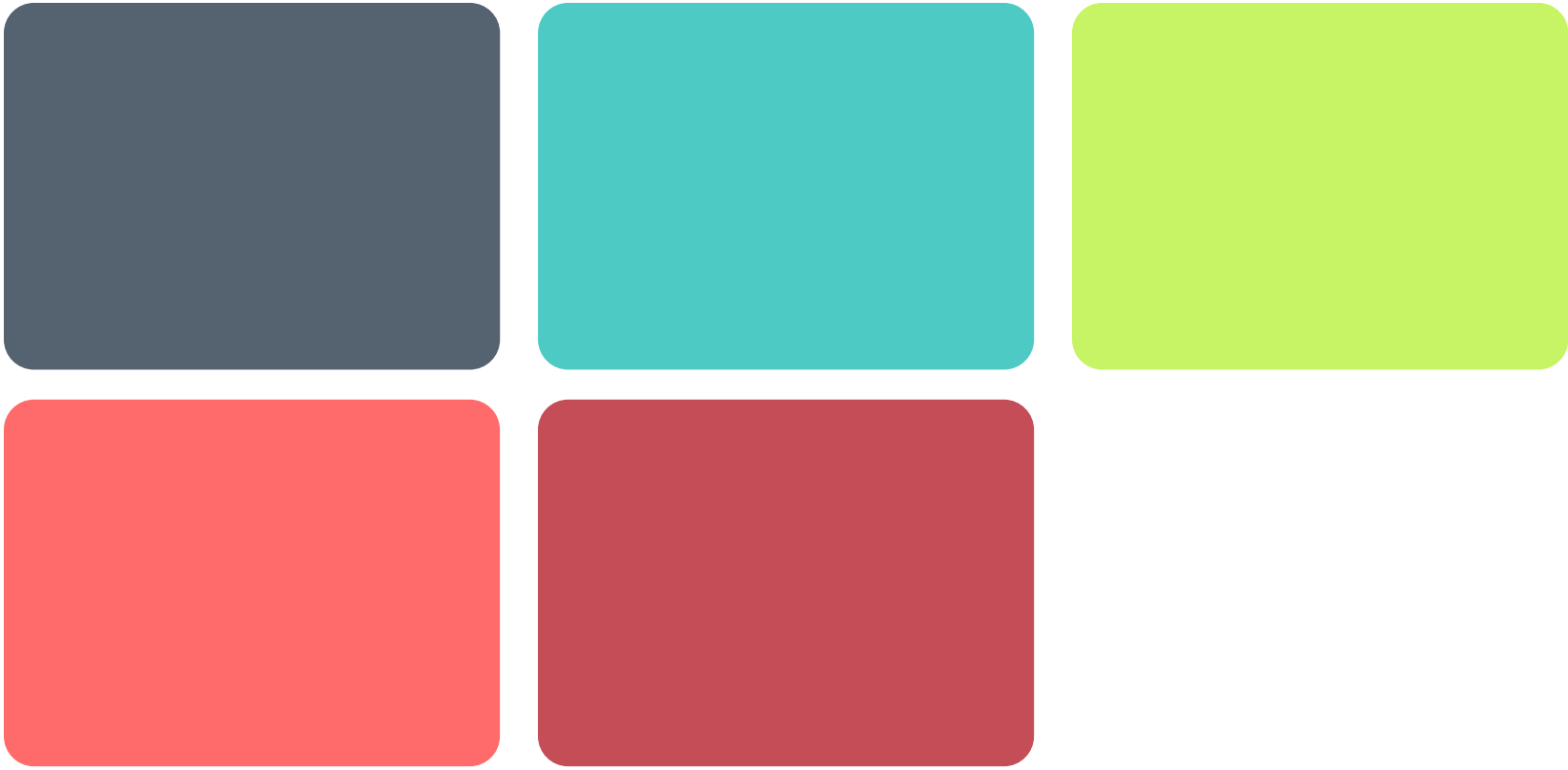
Dark Palette



Light Palette



Smartsheet Gantt Palette



Light Palette