



SoftUni  
Foundation

android development

for beginners

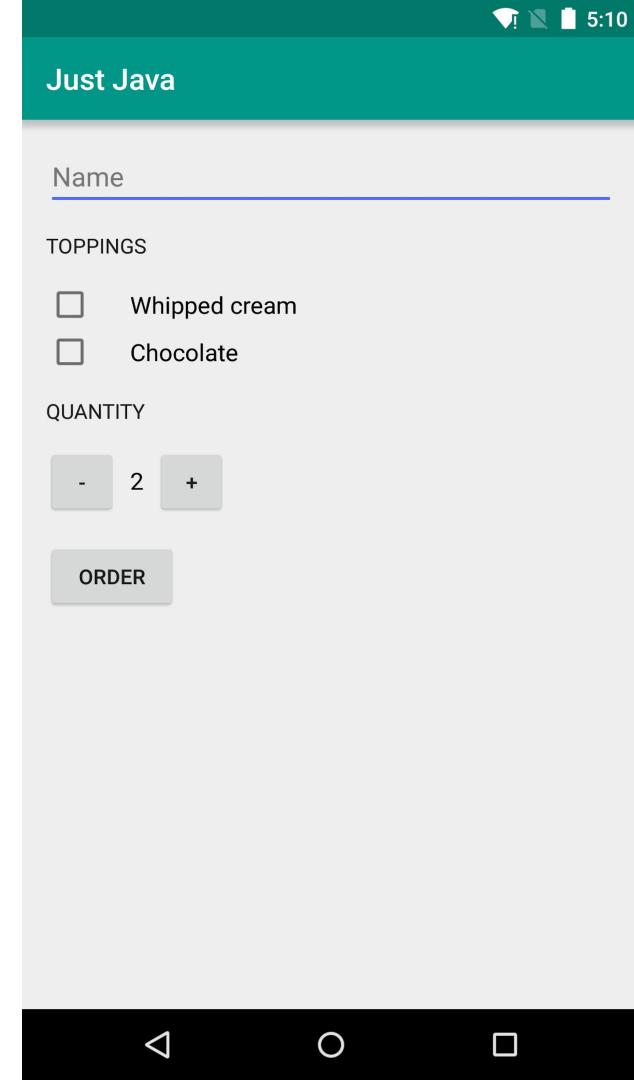
Join at [Slido.com](https://www.slido.com) with #AndroidSoftUni

# Part 2

# connecting to Java

# What we'll learn

- We'll have the user interact with our app
- Collect data
- Send it
- We'll make a simple coffee ordering app





## CREATE NEW PROJECT

- create new project and run it on your device

Application Name: Just Java

Domain Name: android.example.com

Package Name: com.example.android.justjava

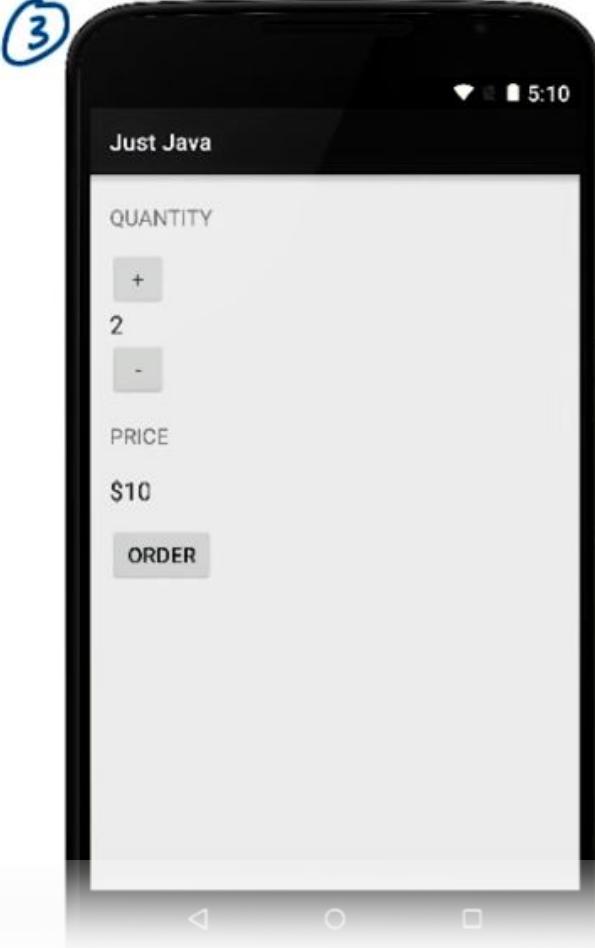
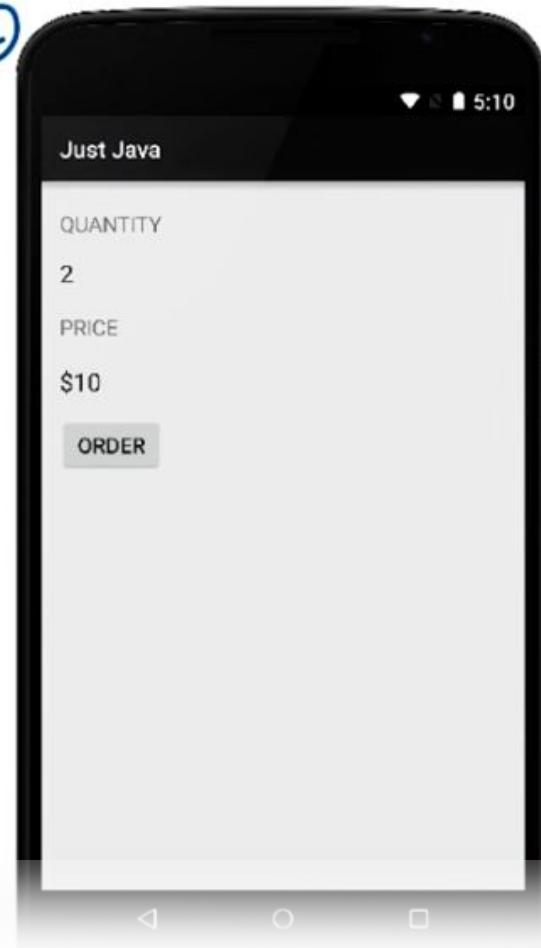
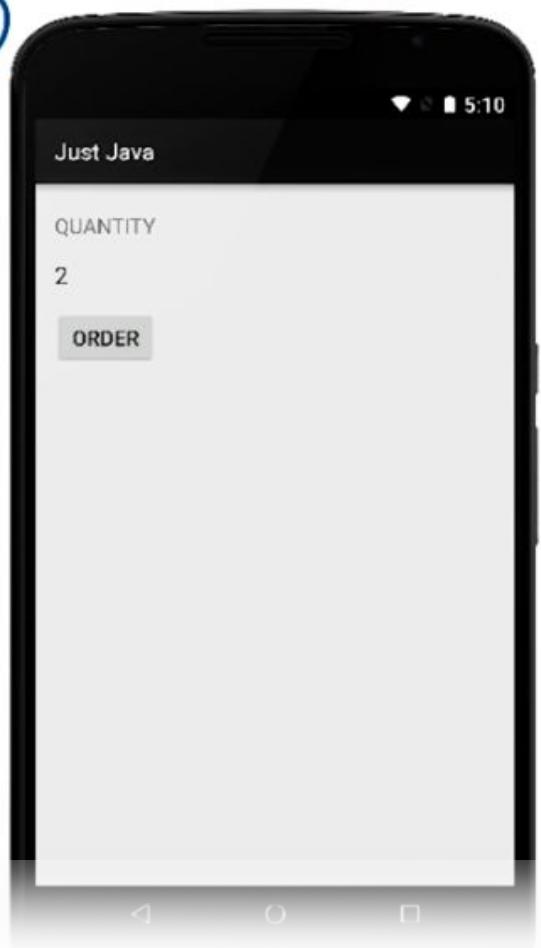
Use Empty Activity template

Minimum SDK Level: API 15 Android 4.0.3

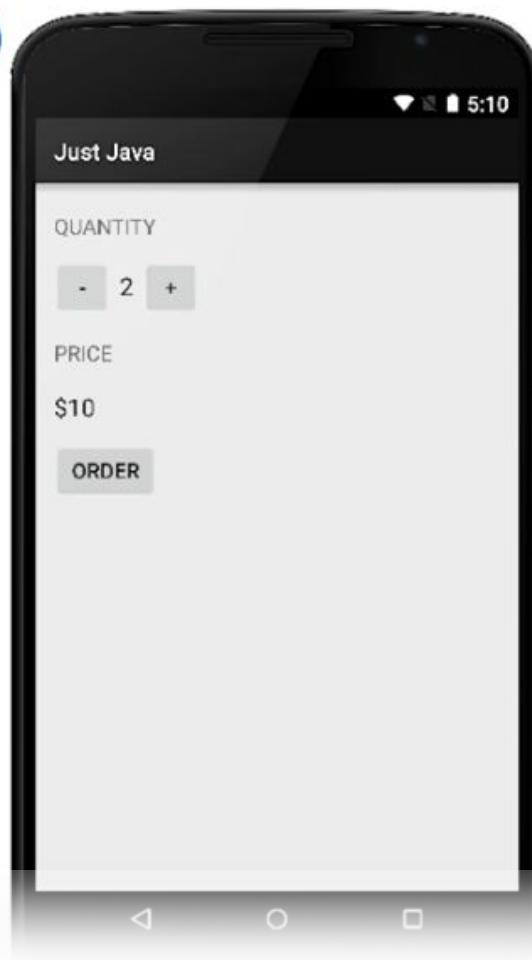
Ice Cream Sandwich



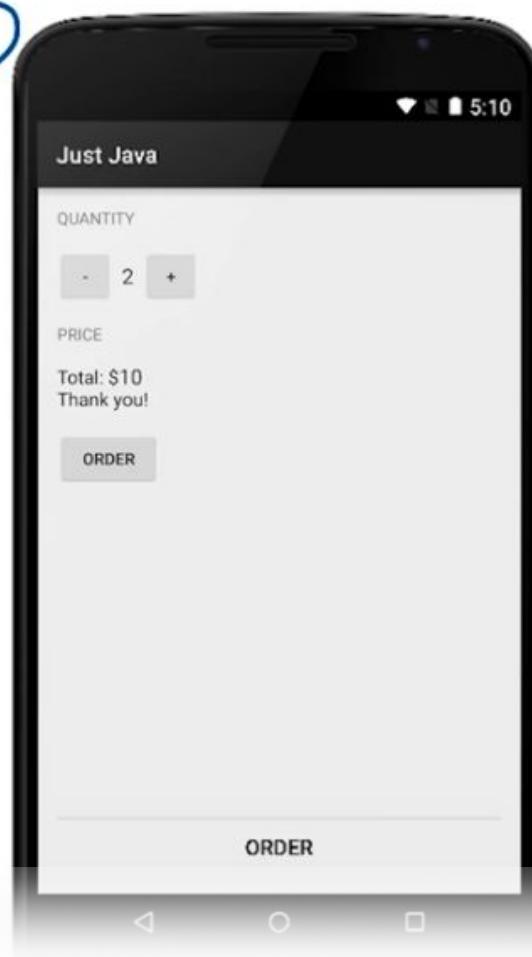
ui first



④

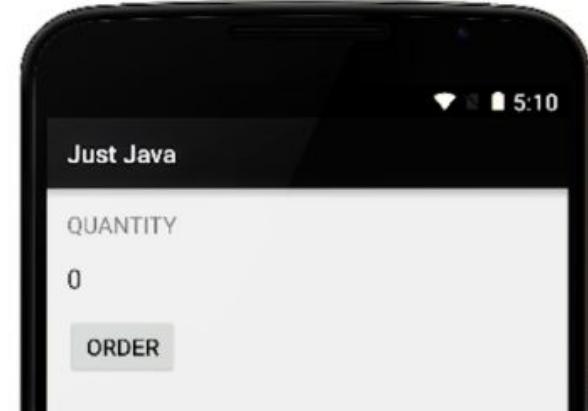


⑤



# PLAN! HOW TO BUILD THIS LAYOUT

STEP 1: Select Views (which views?)



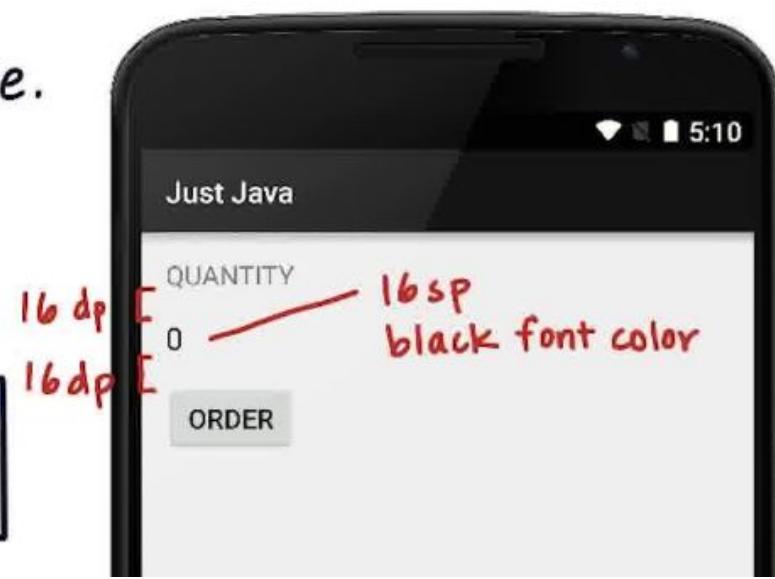
STEP 2: Position Views (Which ViewGroup will be root view?)

STEP 3: Style Views (Anything we need to do here?)

# BUILD THIS LAYOUT

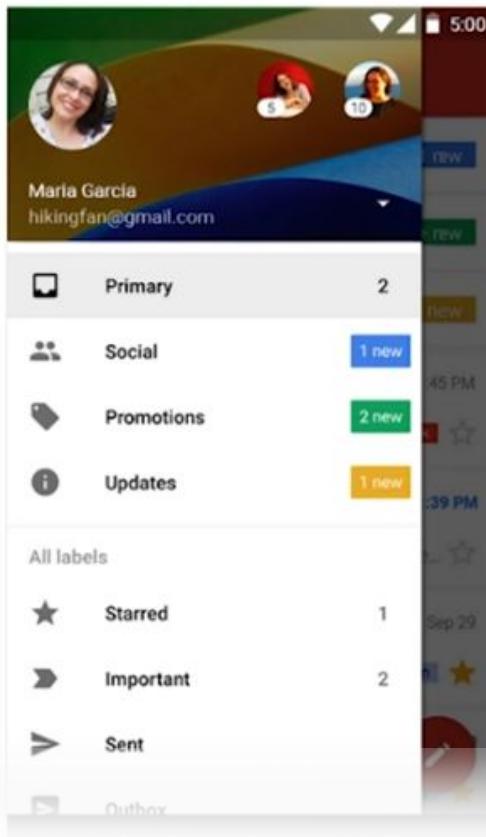
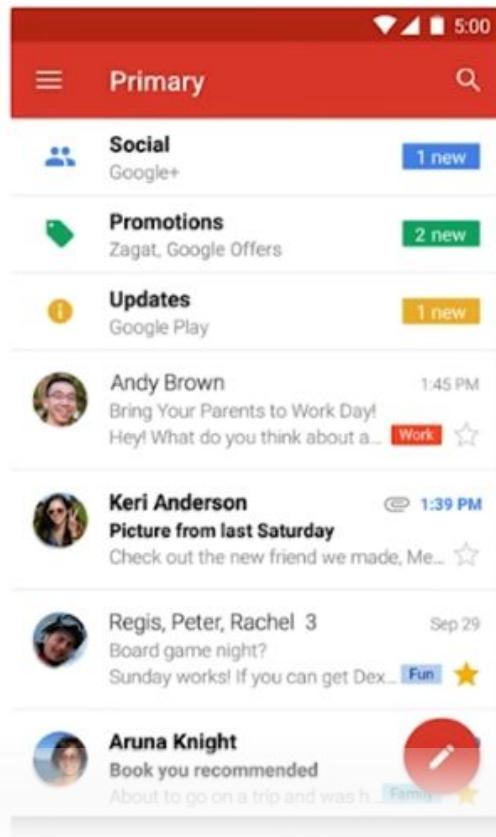
1. Modify `activity-main.xml` to build this layout.
2. Assign the second `TextView` (that shows 0) a view ID name of `@+id/quantity-text-view`
3. Run the app on your device.

What happens when you click on the button?

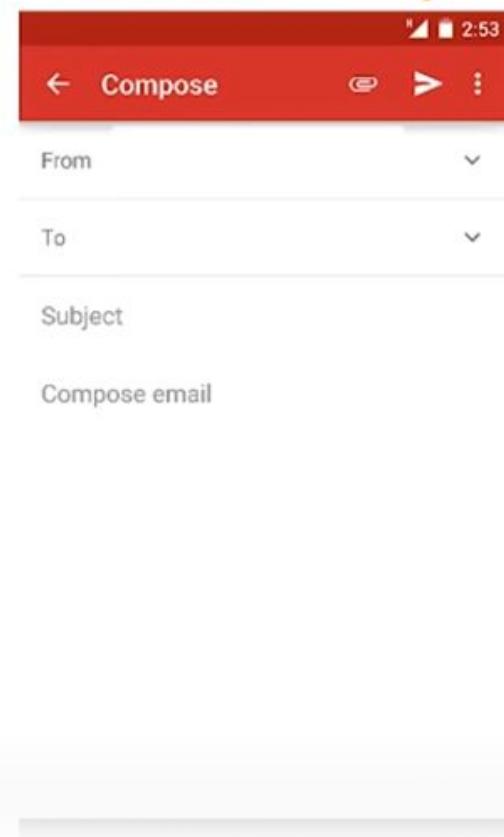


java code

## ConversationList Activity



## Compose Activity



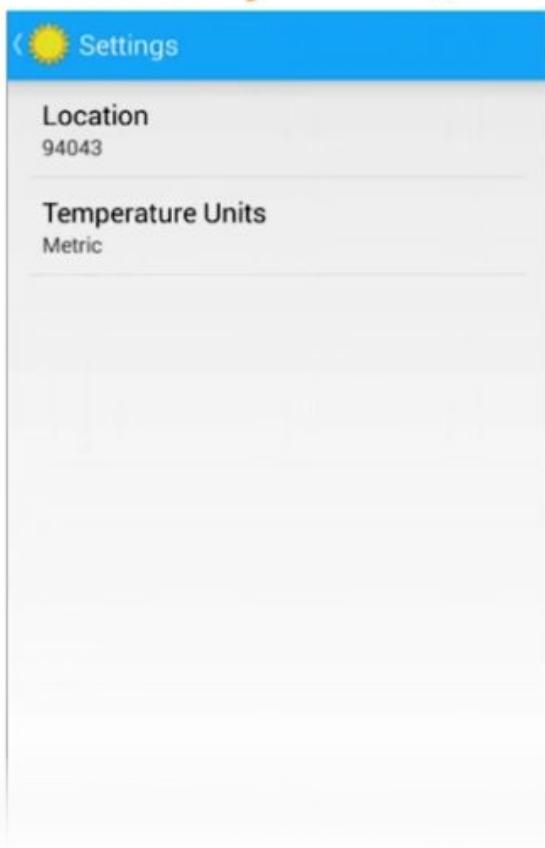
## MainActivity



## DetailActivity



## SettingsActivity



# Open our MainActivity.java

Find it in app/your\_package\_name/MainActivity

And Don't Panic!

# Steps to follow

1. Delete the code in your MainActivity
2. Open this link:  
<https://gist.github.com/udacityandroid/83242daf8a43b743d05e98733a35b19f>
3. Copy the new code and paste it inside your MainActivity
4. Add to your xml button the line: android:onClick="submitOrder"
5. Run the app

# BUTTON CLICK

1. Modify `activity-main.xml` to add this Button XML attribute.

`android:onClick = "submitOrder"`

2. Replace entire `MainActivity.java` file with the file provided in the link in instructor notes.

3. Run app on device.



What happens when you click on the Button?

1 2  
3

## DISPLAYING A NUMBER

Experiment with the line of code in  
`MainActivity.java` that says:

```
display(1);
```

- Try changing the number inside the parentheses.

4 5  
6

# ANDROID WILL DO THE MATH FOR YOU

## ARITHMETIC OPERATORS

Addition + Subtraction -

Multiplication \* Division /

Experiment with different math expressions.

```
display(18 * 3 + 4 / (2 + 2) - 1);
```



Display # of coffees needed if...

There are 77 Android developers who drink 2 cups each and 1 person walks in late and needs a cup too.

# ADD TEXTVIEW FOR PRICE \$

- 1. Add 2 TextViews to layout

assign view ID

`@+id/price_text_view`

to view displaying price →



- 2. Modify Main Activity to include the new `displayPrice(View view)` method (see link in notes)

\*Make sure Auto Import is on in Android Studio

- 3. Add another line of code  
in `submitOrder(View view)`

example →

```
display(2);  
displayPrice(2*5);
```

# variables

## USING LITERAL

Quantity is:

2

Price is:

2 \* 5

Paper cup charge is:

2 \* 2

## USING VARIABLE

Quantity is:

numberOfCoffees

Price is:

numberOfCoffees \* 5

Paper cup charge is:

numberOfCoffees \* 2

# USING VARIABLE

```
int numberOfCoffees = 2;
```

```
display(numberOfCoffees);
```

```
displayPrice(NumberOfCoffees * 5);
```



# USING VARIABLE

```
int numberOfCoffees = 3;
```

```
display(numberOfCoffees);
```

```
displayPrice(numberOfCoffees * 5);
```



## DECLARE A VARIABLE

```
int numberOfCoffees = 2;
```

Data  
type

Variable  
name

=

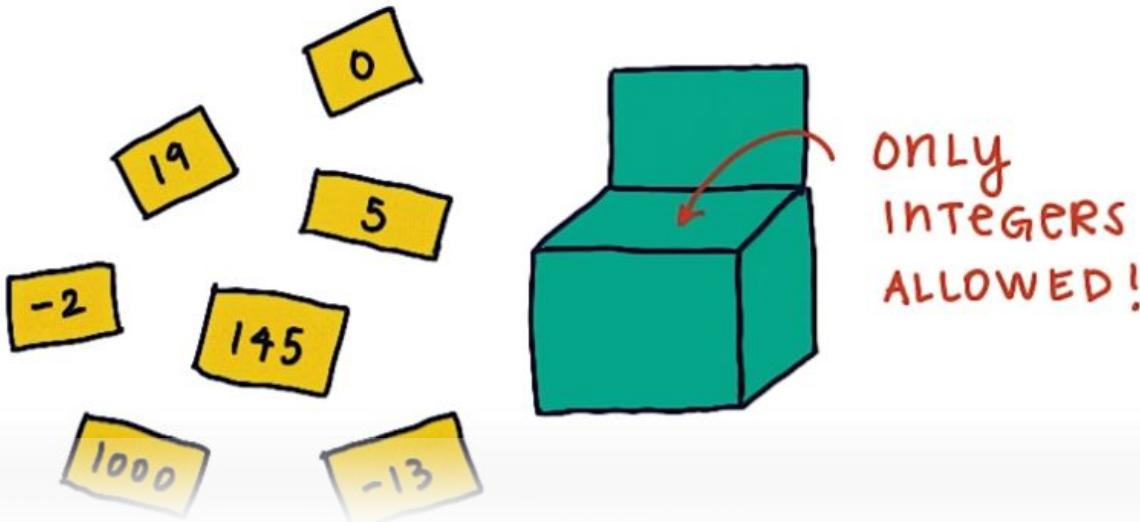
Initial  
value

;

# DECLARE A VARIABLE

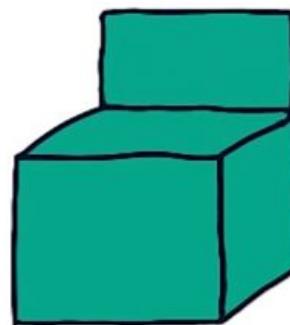
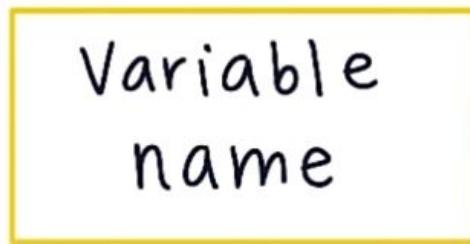
```
int numberOfCoffees = 2;
```

Data type



# DECLARE A VARIABLE

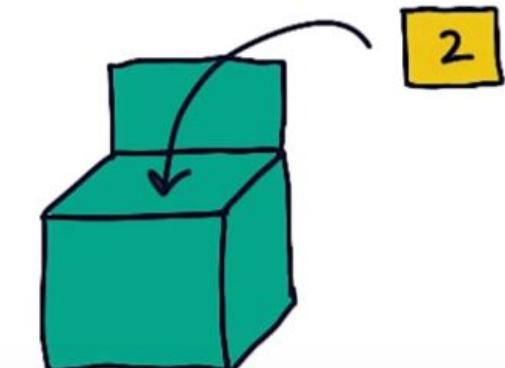
```
int numberOfCoffees = 2;
```



number Of Coffees

# DECLARE A VARIABLE

```
int numberOfCoffees = 2;
```



numberOfCoffees

Assignment  
operator

"Assign value to  
variable"

# DECLARE A VARIABLE

```
int numberOfCoffees = 2;
```

Data  
type

Variable  
name

=

Initial  
value

;

# CREATE AND USE VARIABLES, CONT'D

- ☐ 1. Create an integer variable to store #of coffees

```
int numberOfCoffees = 2;
```

- ☐ 2. Use variable

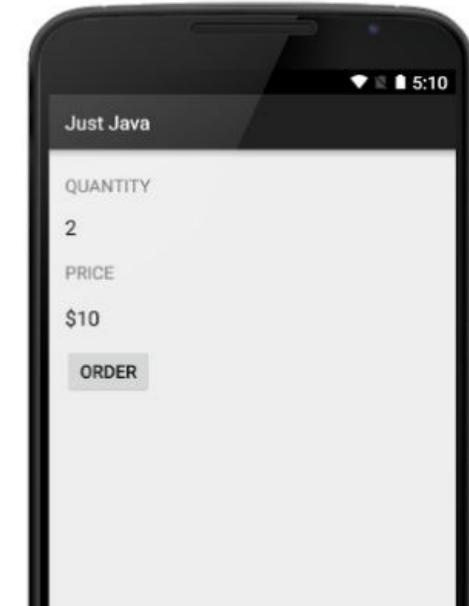
```
display(numberOfCoffees);  
displayPrice(numberOfCoffees * 5);
```

- ☐ 4. Assign a new initial value

```
int numberOfCoffees = 2;
```

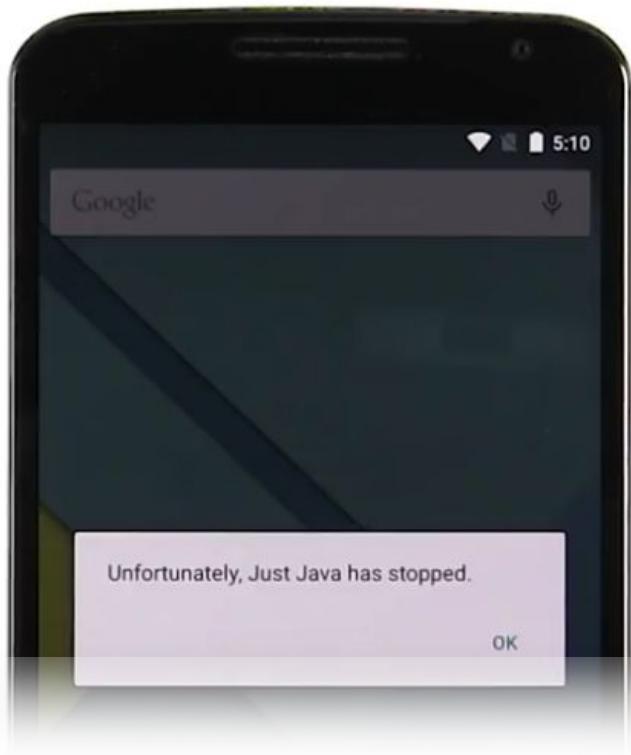
- ☐ 5. Assign a new variable name like quantity

- ☐ 3. Check app



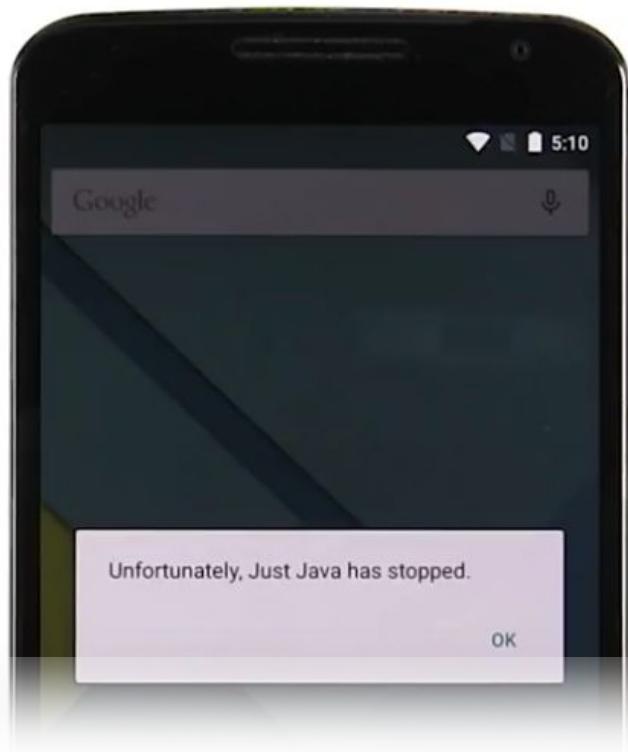
# crashes

UH OH ... THE APP CRASHED !



# UH OH ... THE APP CRASHED !

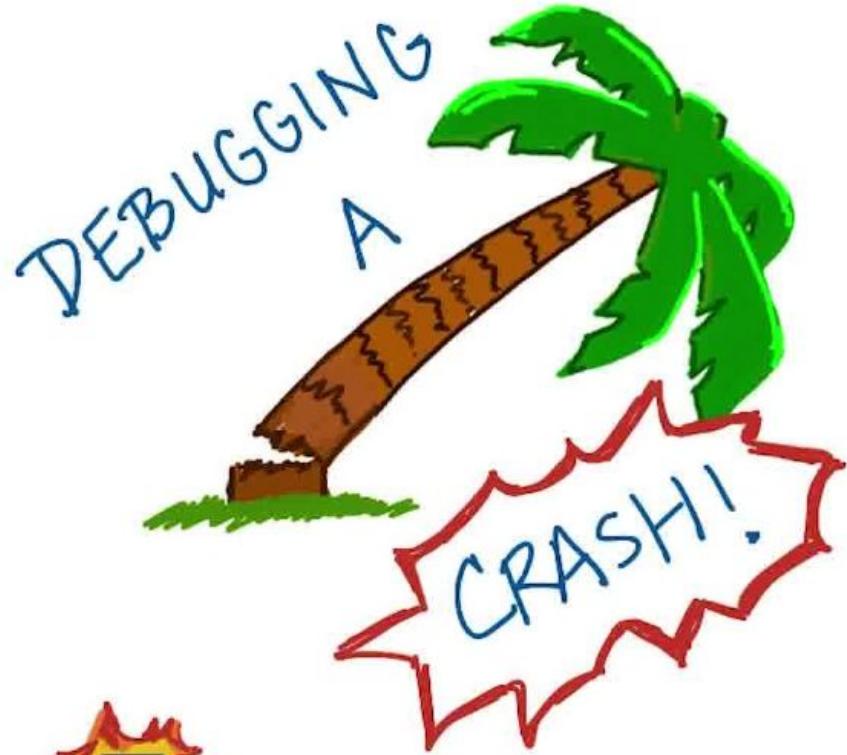
What do  
I do?



Debug  
the  
app!

# How to make it crash

- Change the method name submitOrder to something else
- Run the app
- Click the Order button
- Observe the crash log



Check this box when  
you're done!

1. Create a crash in your app by changing submitOrder
2. Check the logs for the error stack trace  
↳ read the error message
3. Fix the error so your app works again

# Quantity Picker



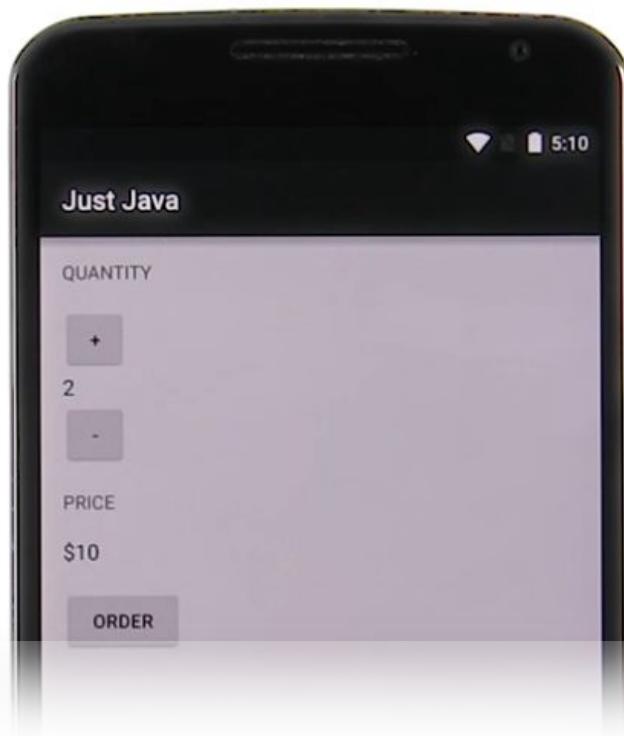
- 1. Modify **activity-main**
  - change layout
  - when call increment
  - when call decrement

- 2. For increment method
  - create **quantity** variable and initialize to **3**
  - display **quantity**

- 3. For decrement method
  - create **quantity** variable and initialize to **1**
  - display **quantity**

# debugging

# DEBUGGING IN ANDROID



# Steps

- Click in front of the code to add a breakpoint
- Add a breakpoint on the first lines of the increment and decrement methods
- Run the app in debug mode (bug icon)
- Use F8 to step over lines of code and F7 to step into code
- Observe the behavior of the app while you are debugging

# DEBUGGING IN ANDROID

- Add red breakpoint in first line of increment & decrement methods

29  
30  
31  
32



public void  
int qua  
display

- Run in Debug Mode



- Step through each line of code.  
Click to resume execution of app.

# updating variables

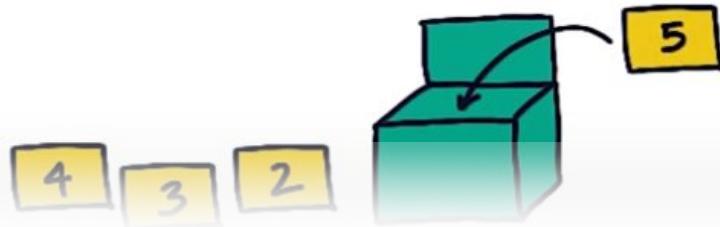
# UPDATING A VARIABLE *in the app!*

```
int quantity = 2;
```

```
quantity = 3;
```

```
quantity = 4;
```

```
quantity = 5;
```



# UPDATING A VARIABLE

- Modify the `increment()` method in the `MainActivity.java` file to update the `quantity` variable.

```
int quantity = 2;  
quantity = 3;  
display(quantity);
```



moar  
coffee

- Use the debugger to step through and verify that `quantity` variable is updated as expected.

- Repeat above steps for the `decrement()` method. Initialize `quantity` to 2, then update to 1 cup of coffee.



# UPDATING A VARIABLE

## Pseudocode

Create **quantity** variable, set it to **2**.

Take current **quantity** value, add **1**,  
and make that the new **quantity** value.

Take current **quantity** value, add **1**,  
and make that the new **quantity** value.

## Java Code

```
int quantity = 2;
```

```
quantity = quantity + 1;
```

```
quantity = quantity + 1;
```

# UPDATING A VARIABLE

```
int quantity = 2;  
quantity = quantity + 1;  
quantity = quantity + 1;
```

Diagram illustrating the state of memory after each assignment:

- Initial state: A green box labeled "2".
- After first assignment: A green box labeled "3".
- After second assignment: A green box labeled "4".

The diagram shows three identical green rectangular boxes, each with a small arrow pointing to a yellow square containing a number (2, 3, or 4). This visualizes how the variable's value is updated in memory.

# UPDATING A VARIABLE

To completely  
new value

Based on current  
value of variable

```
int quantity = 2;
```

```
int quantity = 2;
```

```
quantity = 3;
```

```
quantity = quantity + 1;
```

```
quantity = 4;
```

```
quantity = quantity + 1;
```



## INCREMENT METHOD



- Modify the `increment()` method in the `MainActivity.java` file to update the `quantity` variable.

```
int quantity = 2;  
quantity = quantity + 1;  
display(quantity);
```



- Use the debugger to step through and verify that `quantity` variable is updated as expected.



- Experiment with updating the `quantity` variable with other expressions like `quantity = 2 * quantity;` Then revert code back.

# DECREMENT METHOD



- Modify the `decrement()` method in the `MainActivity.java` file to initialize the `quantity` variable to 2 cups of coffee. Then decrease `quantity` by 1 coffee cup.



- Use the debugger to step through and verify that `quantity` variable is updated as expected.

# variable scope

## LOCAL VARIABLE SCOPE

```
public class MainActivity {  
    public void increment(View view){  
        int quantity = 2;  
        quantity = quantity + 1;  
        display(quantity);  
    }  
  
    public void decrement(View view){  
        int quantity = 2;  
        quantity = quantity - 1;  
        display(quantity);  
    }  
    :  
}
```

## GLOBAL VARIABLE SCOPE

```
public class MainActivity {  
    int quantity = 2;  
    public void increment(View view){  
        quantity = quantity + 1;  
        display(quantity);  
    }  
  
    public void decrement(View view){  
        quantity = quantity - 1;  
        display(quantity);  
    }  
    :  
}
```

# CREATE A GLOBAL VARIABLE

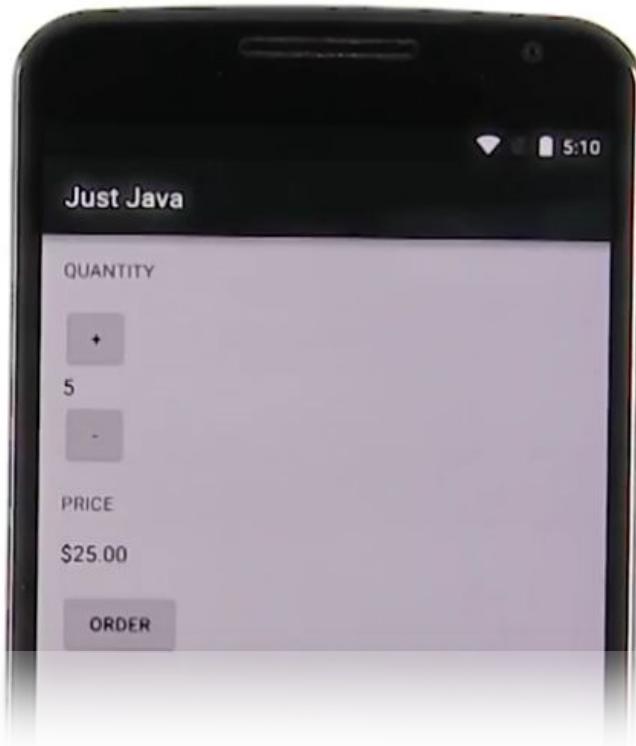
- Replace 2 local **quantity** variables with 1 global **quantity** variable.
- Use the debugger to verify that **quantity** variable is updated as expected.
- Experiment. Try different initial **quantity** value like 0. Or try doubling or halving **quantity** each time + or - is clicked.



# FIX THE ORDER BUTTON



Incorrect





# FIX THE ORDER BUTTON

Describe the gap between what we have now and what we want.



Just Java

QUANTITY	5
+ -	+
PRICE	\$25.00
\$10	
ORDER	ORDER

Just Java

QUANTITY	6
+ -	+
PRICE	\$10
\$30.00	
ORDER	ORDER

# variable scope

# QUANTITY PICKER

Makin' it look better



# PLAN! HOW TO BUILD THIS LAYOUT

STEP 1: Select Views (which views?)

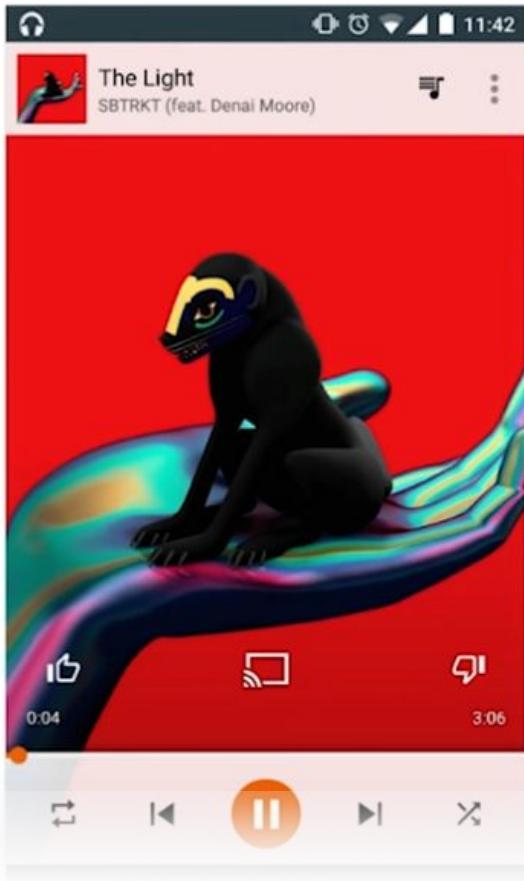


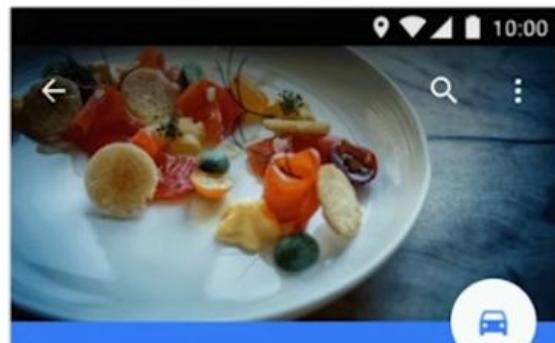
STEP 2: Position Views (Which ViewGroups are relevant?)

STEP 3: Style Views (Anything we need to do here?)

# NESTED VIEWGROUPS







## Namu Gaji

4.3 ★★★★☆ 48 reviews • \$\$

19 min

Asian Restaurant



CALL



SAVE



WEBSITE

An open kitchen turns out innovative dishes by way  
of local produce & Korean culinary traditions.

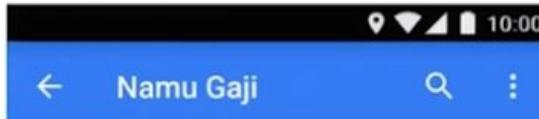


499 Dolores St, San Francisco, CA 94110



11:30 AM – 4:00 PM · 5:00 PM – 11:0... ▾

Menu



### Find a table

Date & Time

Fri, Oct 31, 7 PM

Party size

2 people

Select from the available tables

6:45 PM

7:00 PM

7:15 PM

powered by OpenTable



Add a photo

### Review summary

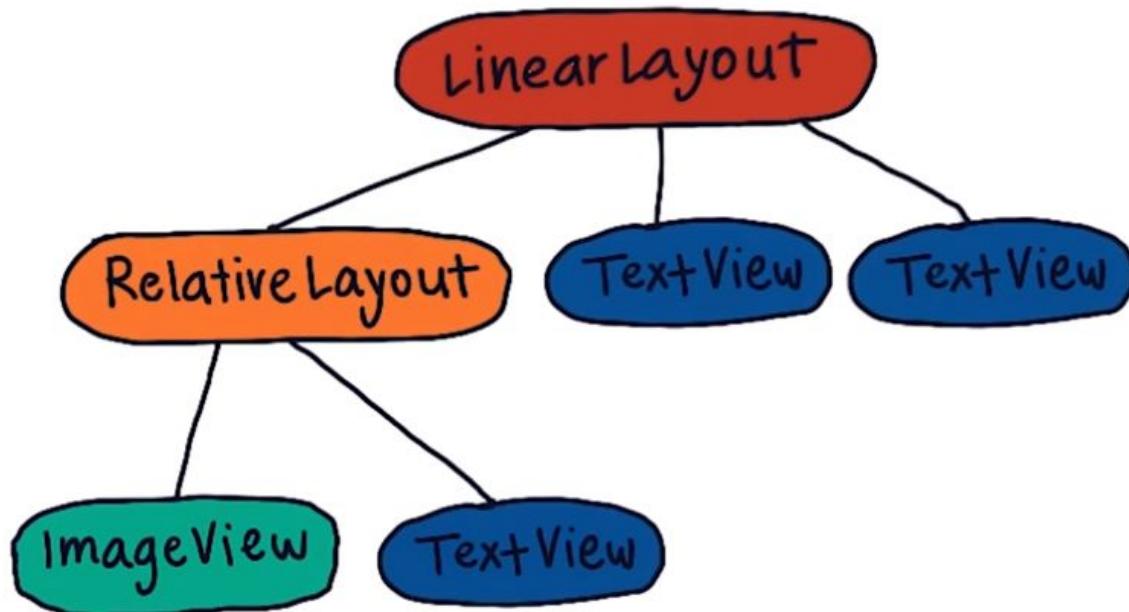
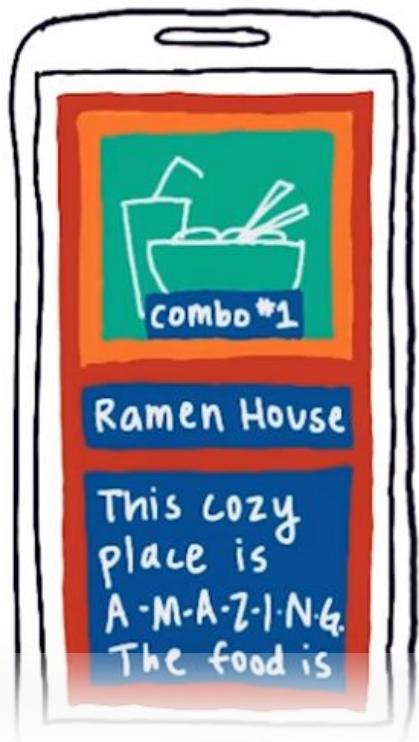
5 \*



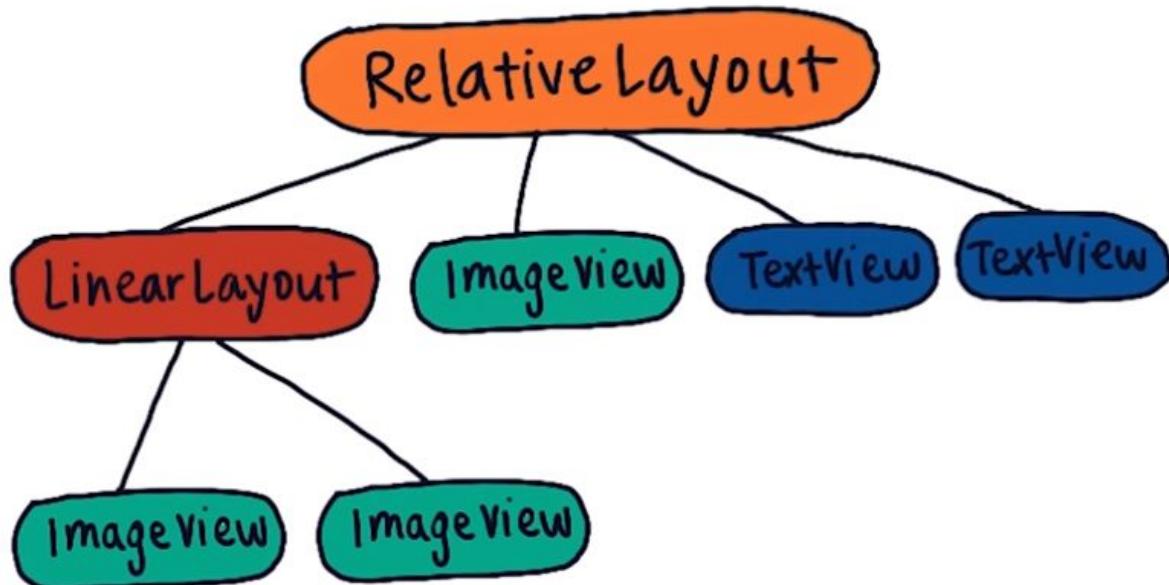
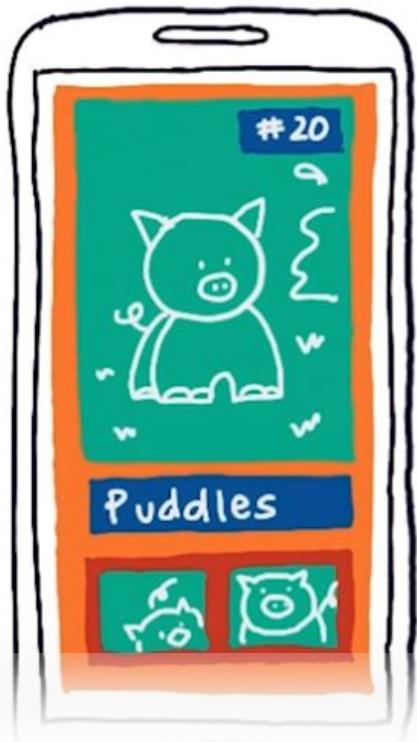
A 2

view hierarchy

# NESTED VIEWGROUPS

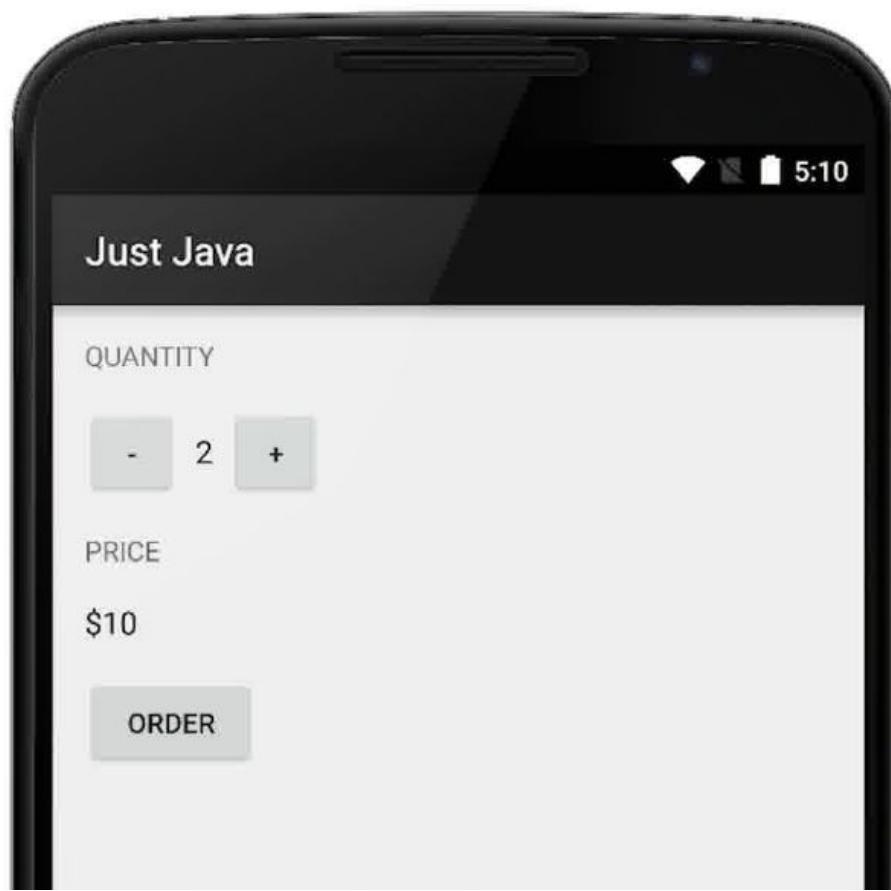


# NESTED VIEWGROUPS

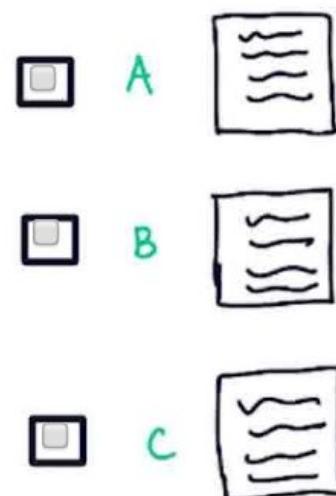


quiz time

# NESTED VIEW GROUPS



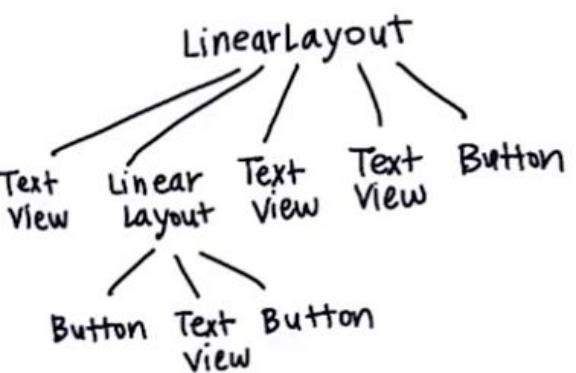
To determine correct XML layout, first draw out view hierarchy diagram for each option.



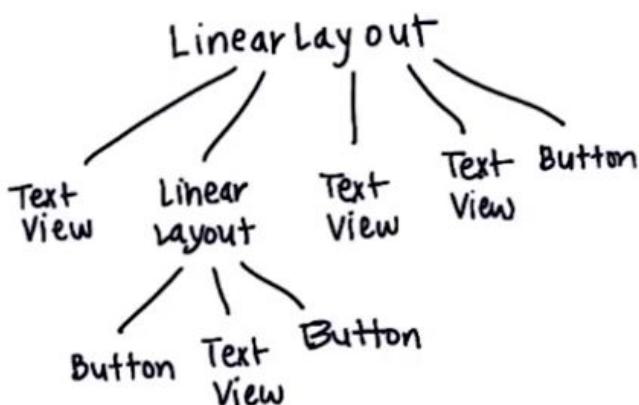
Link: [here](#)

# NESTED VIEWGROUPS

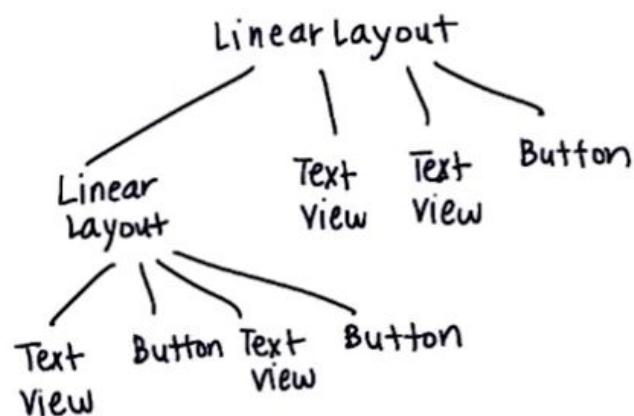
OPTION A



OPTION B



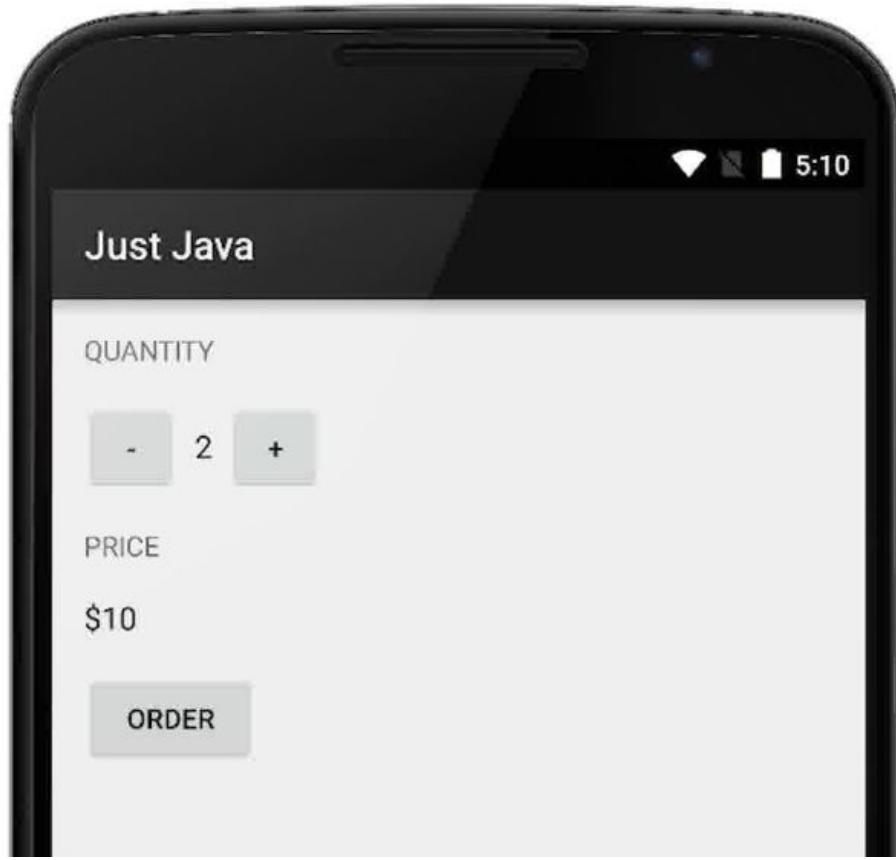
OPTION C



# NESTED VIEWGROUPS

Sketch out what each option would look like on a device.

Which XML file option gives the desired layout?



- A
- B
- C

# BUILD IMPROVED QUANTITY PICKER

Back to  
Android studio  
land



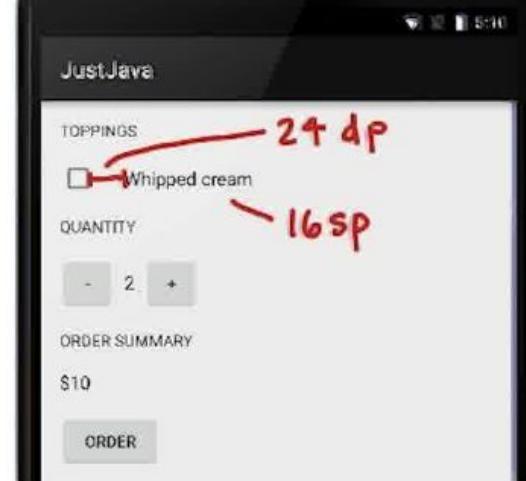
# CHECKBOX FOR WHIPPED CREAM

STEP 1: Select Views

STEP 2: Position Views

STEP 3: Style Views

Then add this to your app!



# NAME FIELD



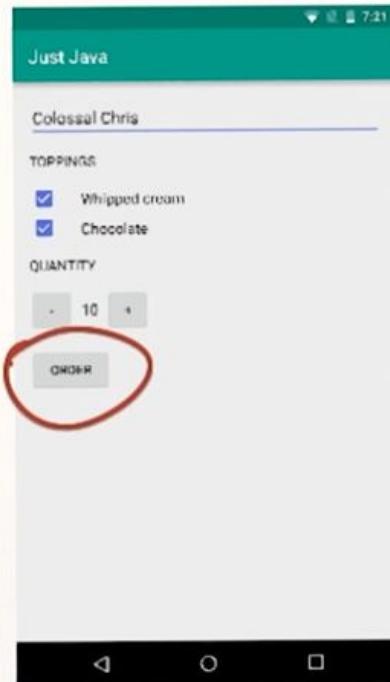
Allow the user to enter their name and update the order summary to include the name.

Plan out the steps you need to do:



\* use logging to verify values along the way

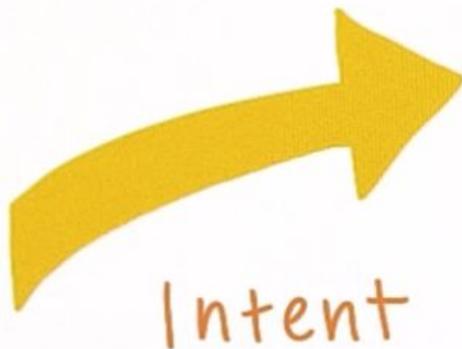
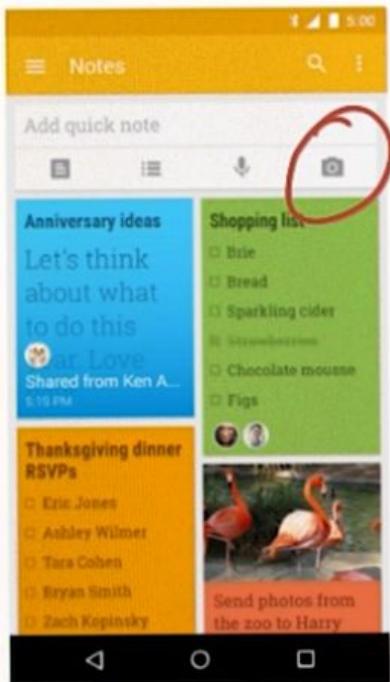
# SENDING INTENTS



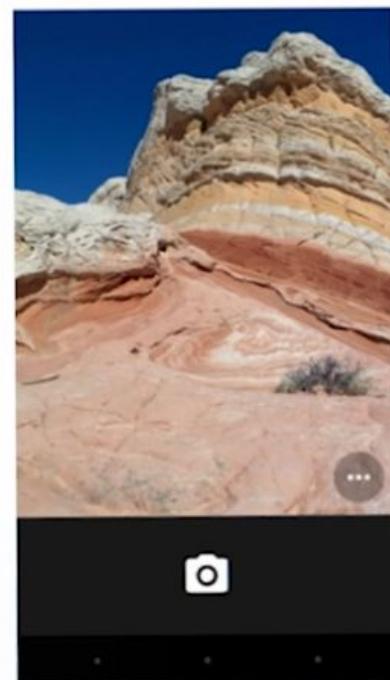
Just Java app

Email app

# SENDING INTENTS



Google keep app

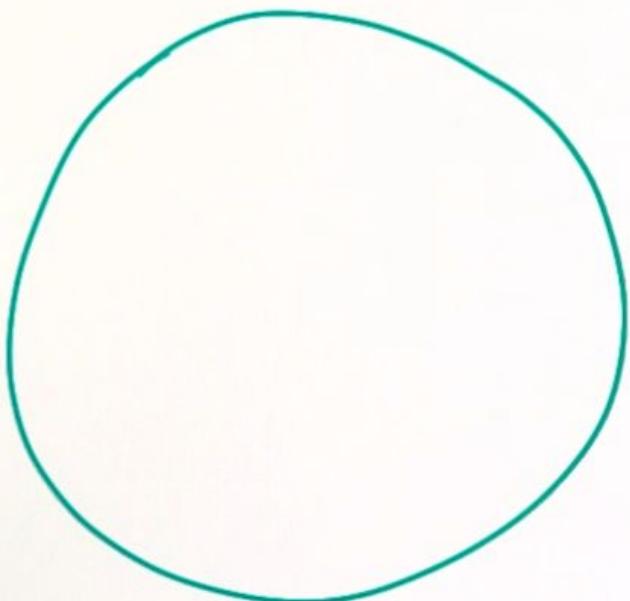


Camera app

# WHAT'S INSIDE AN INTENT?

Example dial intent

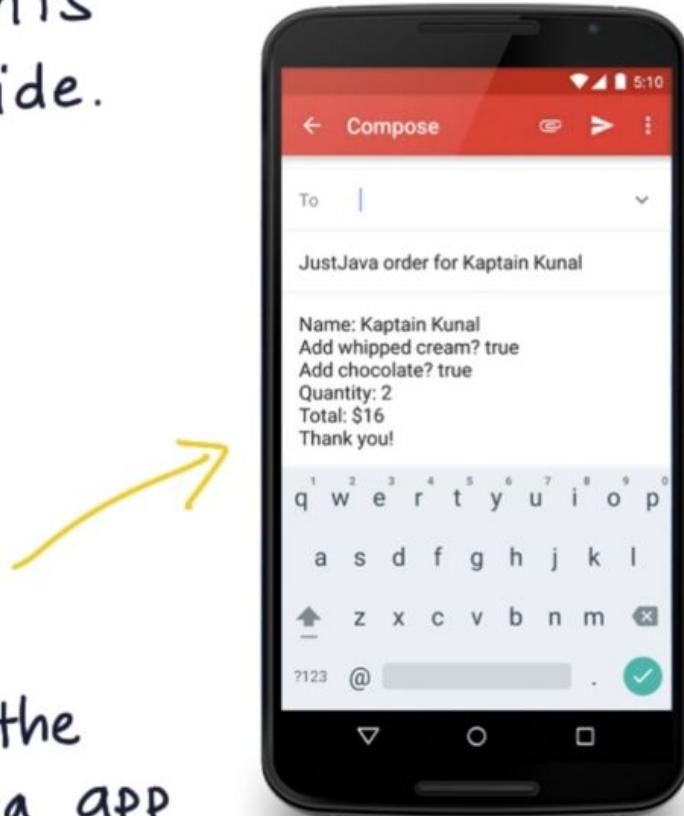
- Action ACTION\_DIAL
- Data tel:2125551212
- Category
- Component
- Extras



# EMAIL OUT THE ORDER SUMMARY

Experiment with sending intents from the **Common Intents** guide.

- Use an intent to send the order summary to an email app
- Populate email message with this text
- Remove code that displays the order summary in the Just Java app



Design your  
own App!

GO

Show it to  
everybody!

CRAZY

# More resources

- <https://blog.stylingandroid.com/>
- <https://android-developers.googleblog.com/>
- <https://www.youtube.com/user/GoogleDevelopers>
- <https://developer.android.com/design/index.html>

# homework #1

## vacation time

# LINEAR LAYOUT WEIGHT

Build this layout !



Start with XML provided below.

1. Determine width of each view first
2. Determine height of each view
3. Make sure the image  takes up any remaining height in parent Linear Layout

Link: <http://labs.udacity.com/android-visualizer/#/android/linear-layout-weight>

# PADDING & MARGIN



- Modify the provided XML to build this desired layout



Link:

<http://labs.udacity.com/android-visualizer/#/android/padding-and-margin>

# homework #2

# party

Kunal Ben Kagure

Natalie  
Lyla Me

Amy  
Omoju  
Jennie

# RELATIVE LAYOUT

Positioning children relative  
to other views

- Modify the provided XML layout to achieve this desired layout



Link:

<http://labs.udacity.com/android-visualizer/#/android/relative-layout-view-ids>

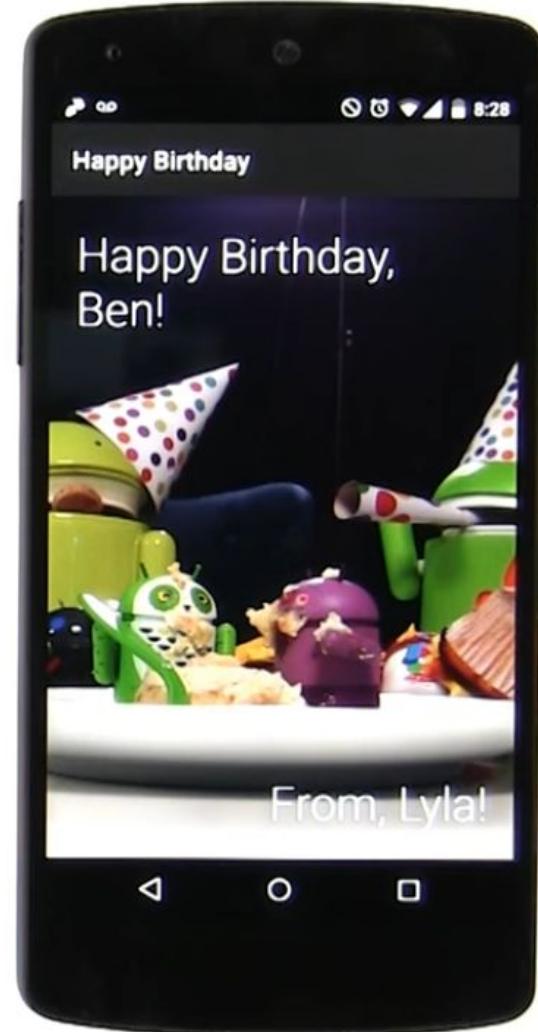
homework #3  
your birthday card app

# Drawing to APP

Step 1: Select the Views

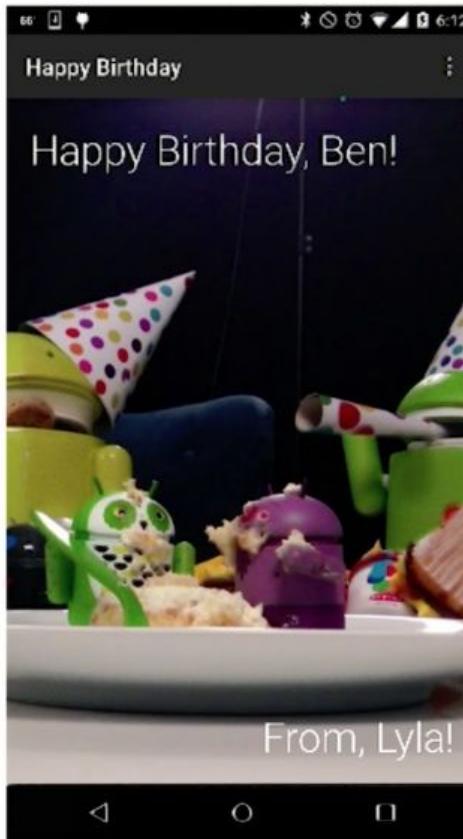
Step 2: Position the Views

Step 3: Style the Views



# Step 1: Select the Views

What Views are included  
in this layout?

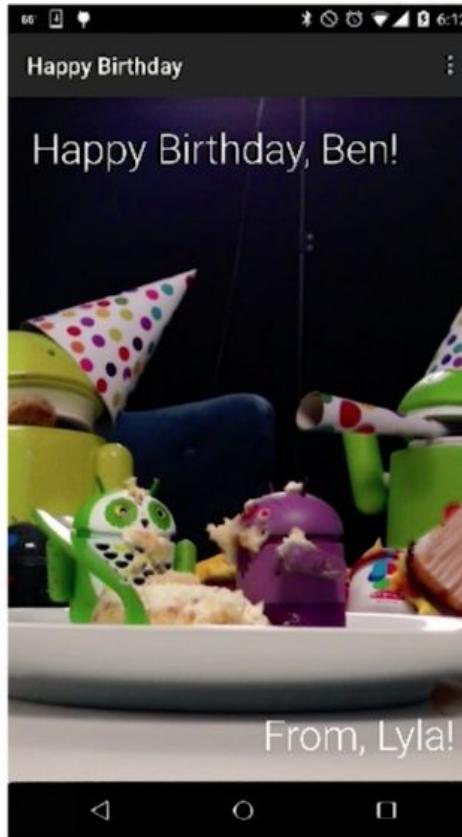


## Step 2: Position the Views

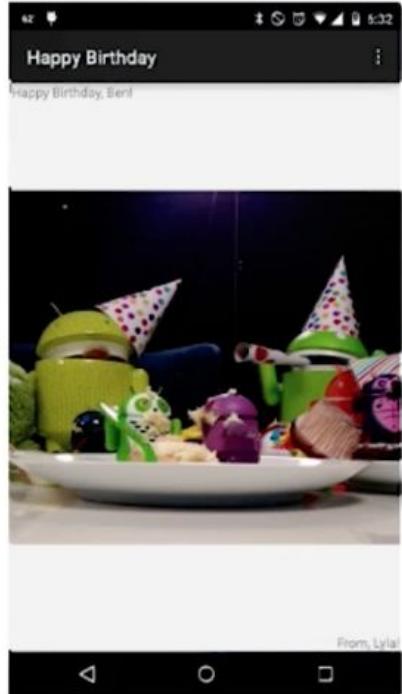
What View Group should you use?

- Linear Layout
- Relative Layout

Why?



# Step 3: Style the Views



Write down all the differences you see between what we have **NOW** and the end **GOAL**:

# homework #4

# some next level designs

Recreate the following design.

You must have a Toolbar and CardView and you must strive to be as accurate as possible.

The content (the card) may not be possible to be shown on the whole screen, so you should add the possibility of scrolling.

You will find resources for the design (icons) in the github of the course.

(<https://github.com/android-soft-uni/03-Views-Layouts> )

