Name: ______________________________

ROSE-HULMAN INSTITUTE OF TECHNOLOGY **CSSE490 Android Application Development**

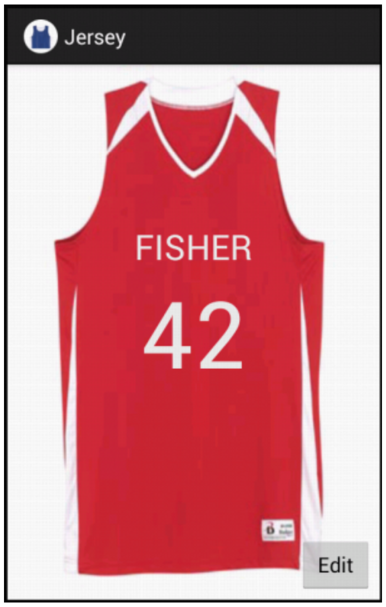
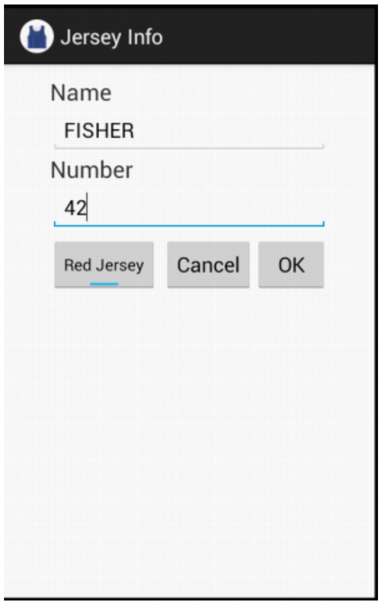
Lab 3: Passing Data in Intents

Our goal is to make an app with multiple Activities that passes data via Intents. "Passing" data got me thinking about basketball so this is a basketball jersey themed app.

To get this lab checked off, show the finished app to your instructor or assistant.

Introduction

This app will contain two activities:

JerseyDisplayActivity	JerseyInfoActivity
	 Name FISHER Number 42

You will be passing three pieces of information back and forth.

Type	Description
String	Player name
int	Player number
boolean	Is jersey red

As the names suggest, JerseyDisplayActivity will display the information and JerseyInfoActivity will let you edit the information.

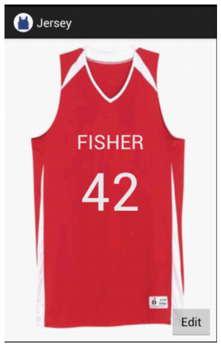
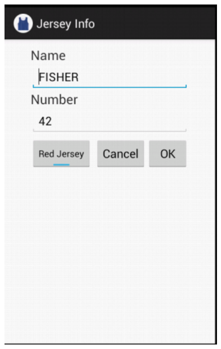
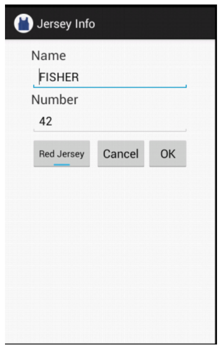
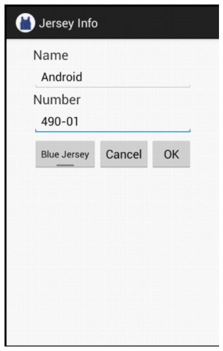
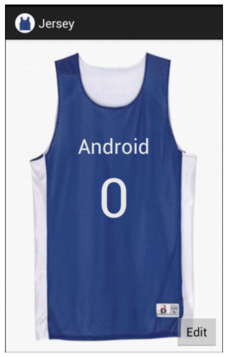
Requirements

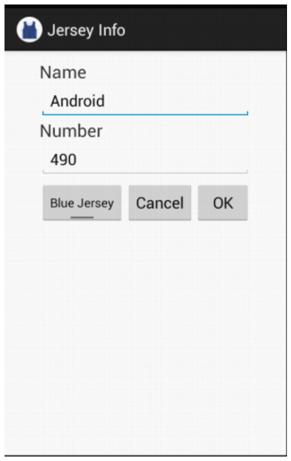
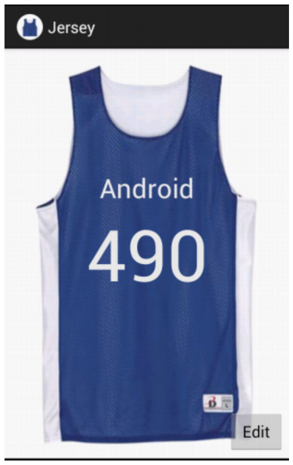
- You must pass the String, int, and boolean between the activities via Intent extras.
- Clicking the edit button on the display activity launches the info activity. Clicking OK or Cancel on the info activity goes back to the display activity. OK will pass the entered data back, but cancel won't.
- The jersey number can only display a number. There should be no way for a user to put any other text in there. Numbers only. If they try to input non numbers you may convert their non-number String to just 0, but the program should not crash.
- JerseyInfoActivity should display the current values - the same info as the current jersey (passed via the Intent). They should not be blank.
- Your app must look exactly like the screen shots. Pay careful attention: the color button is a toggle button :)

You will need to download these files now, since you will need the icon right away:

Jersey Images Files

Here was my basic test :

Launch	Edit Button	Click EditText	Change to invalid	OK Button doesn't crash
	 Jersey Info Name FISHER Number 42 Red Jersey Cancel OK	 Jersey Info Name FISHER Number 42 Red Jersey Cancel OK	 Jersey Info Name Android Number 490-01 Blue Jersey Cancel OK	

Change to Valid	OK Button!
 Jersey Info Name Android Number 490 Blue Jersey Cancel OK	

I also checked to make sure the cancel button actually ignored the info typed.

The best way to test your knowledge is to stop reading this lab now and just start making the app. The rest of the document is just hints.

If you are still reading, you think you need some hints. If you listened in lecture, you probably don't need that many. Really... try to do as much on your own as possible.
It's good practice for exams.

Part A: Jersey Display XML View

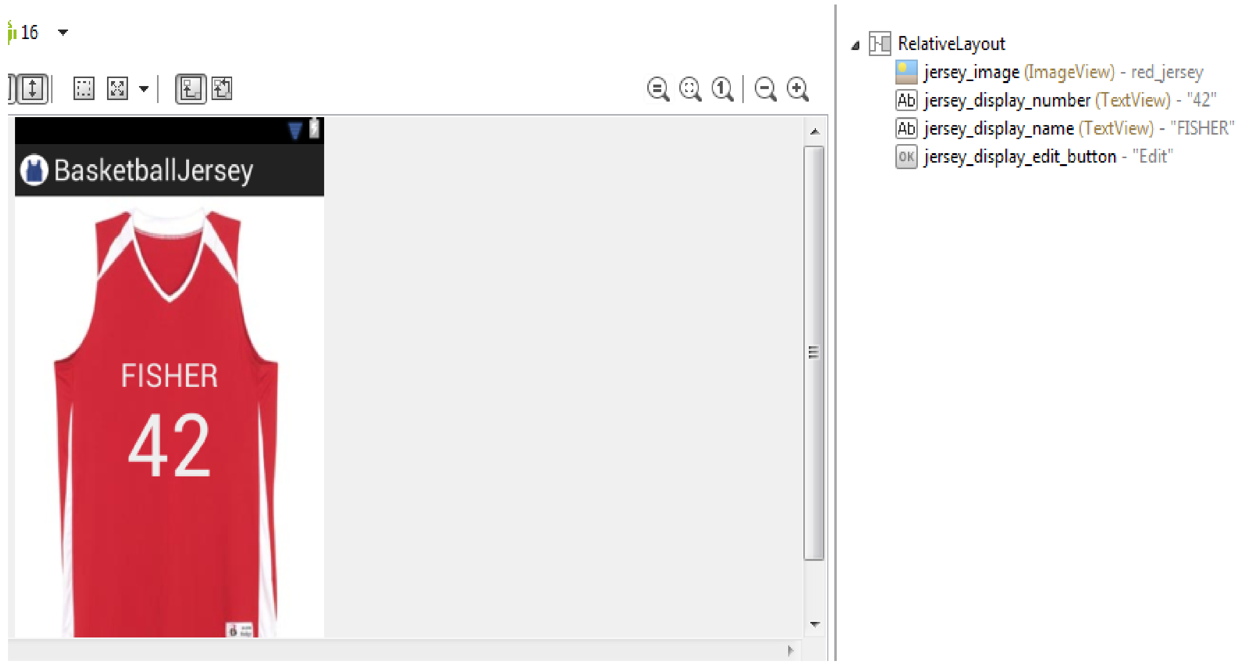
Start by making a new project. I called mine BasketballJersey. Use the icon you downloaded. The new activity name will be JerseyDisplayActivity, and the title will be Jersey. Use any SDK you like.

Next grab the images for the app and move them to the drawable folder. (If you have drawable folders with multiple resolutions, then you can just put them in the drawable-ldpi folder.) Change the app icon to icon_jersey in the manifest if you forgot when you were making the project.

Next I added a few handy string resources:

```
<string name="start_name">FISHER</string>
<string name="start_number">42</string>
<string name="edit">Edit</string>
<string name="name">Name</string>
<string name="number">Number</string>
<string name="red_jersey">Red Jersey</string>
<string name="blue_jersey">Blue Jersey</string>
<string name="ok">OK</string>
<string name="cancel">Cancel</string>
```

Then I started editing activity_jersey_display.xml. Mine looked like this:



```
RelativeLayout
    jersey_image (ImageView) - red_jersey
    jersey_display_number (TextView) - "42"
    jersey_display_name (TextView) - "FISHER"
    jersey_display_edit_button - "Edit"
```

For the ImageView to fill the parent in both height and width, use the fixXY setting for the Scale Type. The number is centered in the parent and the name is placed above it. The text size was selected to make it look more like a jersey. Technically the image is the front of a jersey, but we won't worry about that. Feel free to go grab your own images if you like. :) The button is then bottom and right in the parent. Feel free to add a little margin to move it out of the corner. Hint: if you have overlapping views, do you recall which one is displayed on top? (Review Lab 2 if not.)

Make sure everything has an id and move on to the controller.

Part B: Jersey Display Activity

To start, I wanted to make sure I could programmatically change the name string, int number, and boolean jersey color. So I made a few member variables to hold the string, int, and boolean, and a tag for Log messages, and the captured ImageView, TextView, and TextView.

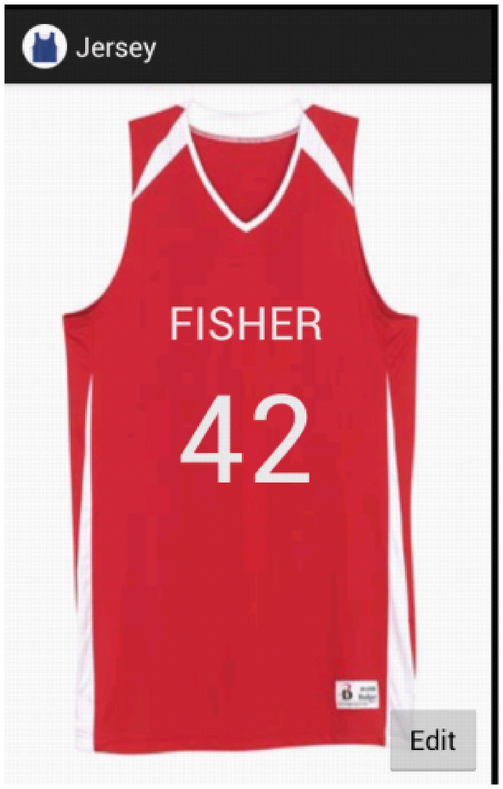
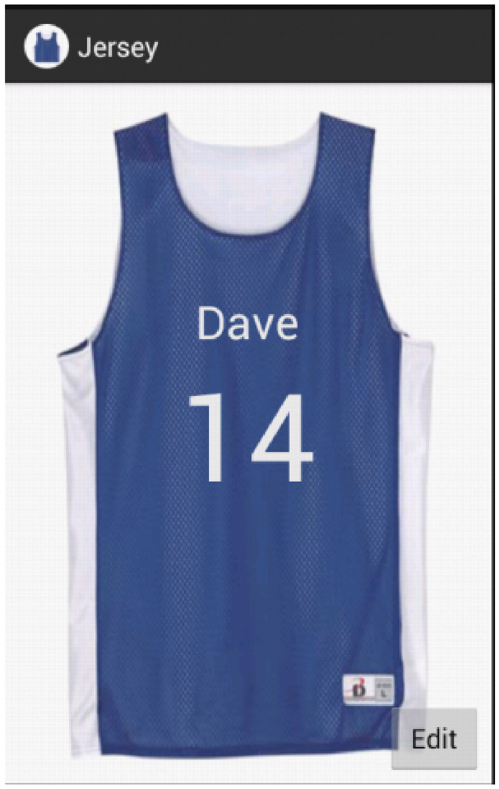
```
public static final String JDA ="JDA";
private ImageView mJerseyImageView;
private TextView mNameTextView;
private TextView mNumberTextView;
private String mPlayerName = "Fisher";
private int mPlayerNumber = 42;
private boolean mJerseyIsRed = false;
```

I captured the findViewById objects and created a helper method called updateJerseyInfo(). Pseudocode for that function:

```
updateJerseyInfo
if mJerseyIsRed
    mJerseyImageView setImageResource to R.drawable.red_jersey
otherwise
    mJerseyImageView setImageResource to R.drawable.blue_jersey
mNameTextView setText to mPlayerName
mNumberTextViewsetText to “” + mPlayerNumber
```

The only new item there is the setImageResource function on the ImageView. The rest is pretty simple. There are a few times in this app where you need to convert an int to a String or a String to an int. Int to String is easy. Going the other way is a bit more trouble since a user may type in a String that doesn’t convert to an int.

If I comment out my updateJerseyInfo method I get the XML layout file directly. With the updateJerseyInfo function I get the String, int, and boolean member variables. You can see I changed all the values to be sure my updateJerseyInfo method is working.

xml	**updateJerseyInfo**
	

Next I made an onClickListener for my button. You can either use an anonymous inner class or have the Activity implement the OnClickListener interface; I chose the inner class.

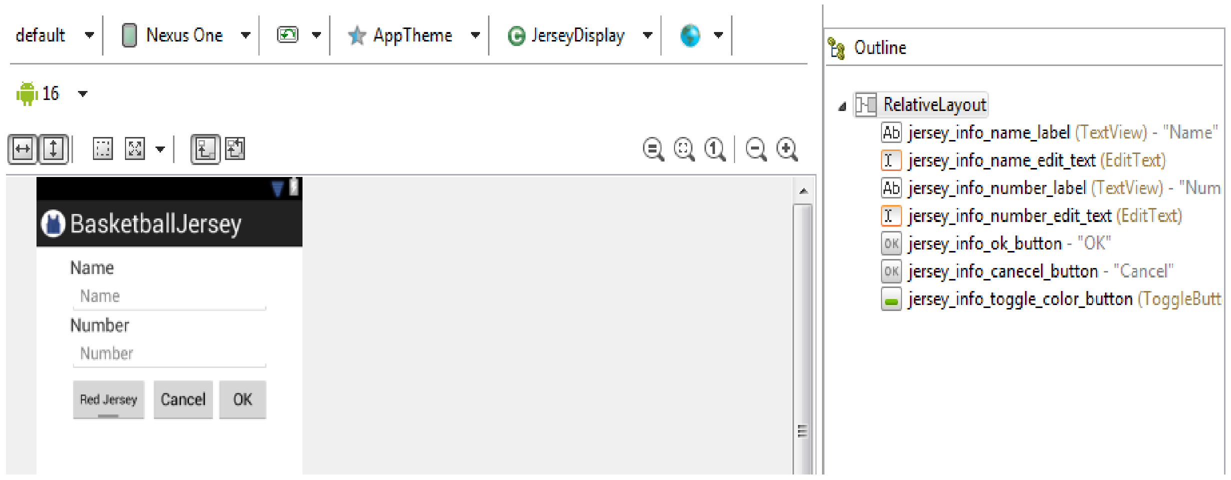
Within the onClick method I put extras into the Intent and used the startActivityForResult. Note, I needed a few constants. I made these:

```
public static final String KEY_PLAYER_NAME = "KEY_PLAYER_NAME";
public static final String KEY_PLAYER_NUMBER = "KEY_PLAYER_NUMBER";
public static final String KEY_JERSEY_IS_RED = "KEY_JERSEY_IS_RED";
private static final int REQUEST_CODE_JERSEY_INFO = 1;
```

Once you've got the code ready to launch the other Activity we can stop on JerseyDisplayActivity.java for now. Later we will need to write the method onActivityResult to handle the update when the data comes back.

Part C: Jersey Info XML View

I made the JerseyInfo.java class (superclass Activity) and setup the onCreate method to setContentView(R.layout.activity_jersey_info). Next I registered this activity in the manifest because I was afraid I'd forget to do that later. I then made the activity_jersey_info.xml file, with the root element of RelativeLayout. My jersey_info.xml layout looked like this:



RelativeLayout
- jersey_info_name_label (TextView) - "Name"
- jersey_info_name_edit_text (EditText)
- jersey_info_number_label (TextView) - "Num
- jersey_info_number_edit_text (EditText)
- jersey_info_ok_button - "OK"
- jersey_info_canecel_button - "Cancel"
- jersey_info_toggle_color_button (ToggleButt

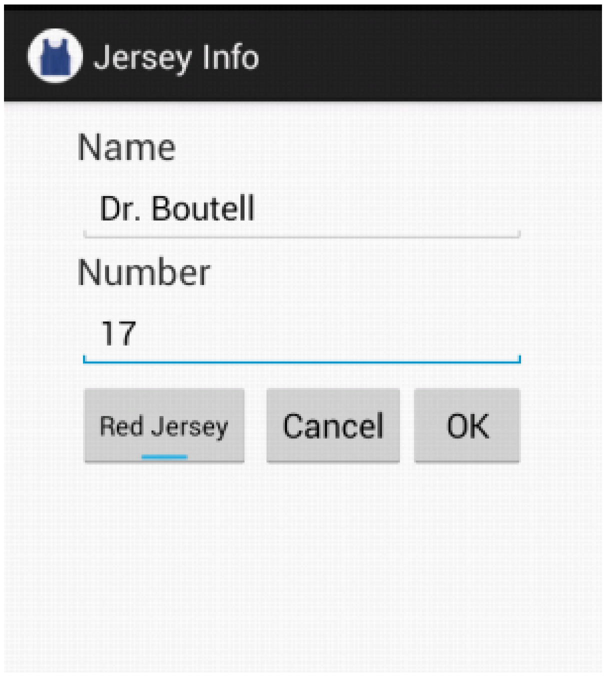
Here is a rough breakdown of my layout.

Relative Layout with a left and right padding of 40dp, top 10dp

- TextView align parent top
- EditText below the TextView
- TextView below the EditText
- EditText below the last Textview
- ToggleButton below the last EditText, 5dp top margin
  - Text off = @string/blue_jersey
  - Text on = @string/red_jersey
- Button for OK align top with ToggleButton align right in parent
- Button for Cancel align top with OK button left of OK button

Note, I intentionally didn't center this in the view to make room for the soft keyboard if it's there. It's easy in this case to allow the space so I just left the bottom open.

Next it's time to see if your XML is working. Run it and see if your new XML appears. The boxes will be blank but you can type into them. (I didn't have a soft keyboard, since my emulator has keyboard support.)



You should be able to type in the EditText boxes and the ToggleButton should change states. However the OK and cancel buttons don't do anything yet.

Next, we'll write code to handle the button presses.

Part D: Jersey Info Class

The JerseyInfoActivity needs to have references to the EditText objects and the ToggleButton so I made a few member variables:

```
private EditText mNameEditText;
private EditText mNumberEditText;
private ToggleButton mJerseyIsRedToggleButton;
```

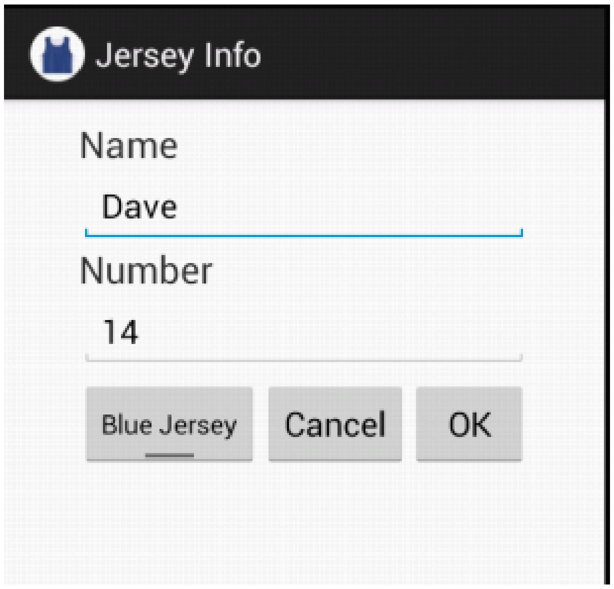
I captured those via findViewById and added listeners for the two buttons. To auto-fill Next I wanted to auto fill those elements using the passed values. Here is the summary of that process.

<u>Auto fill with the passed values</u>

- getIntent
- getStringExtra for JerseyDisplayActivity.KEY_PLAYER_NAME
- mNameEditText setText to the passed name
- getIntExtra for JerseyDisplayActivity.KEY_PLAYER_NUMBER (default any #)
- mNumberEditText setText for the "" + passed number

- getBooleanExtra for JerseyDisplayActivity.KEY_JERSEY_IS_RED (default false)
- mJerseyIsRedToggleButton setChecked to the passed “jersey is red” value

Check to make sure the values autofill using the passed values. If you used my values form the prior part, it should look like this when the activity launches:



Now implement the onClick method for the buttons. The Cancel button onClick is very simple. Just set the result to RESULT_CANCELED and call finish().

The OK button is probably the trickiest part of this app. Just because the input we allowed to the user is a String, but we want an int and we want to avoid app crashes. Thankfully, the Integer.parseInt method throws a NumberFormatException if reads a bad value, so we just need to catch it.

- Get the mNameEditText text and convert to a String via toString()
  - Note getText() returns something that is not a String
- Get the mNumberEditText text and convert to a String via toString()
- Convert that string to an int. This is not meant to be the focus of this assignment so I’ll just show you my code. Here is the solution that I used:

```
String numberString =
mPlayerNumberEditText.getText().toString();
int number = 0;
try {
     number = Integer.parseInt(numberString);
} catch (NumberFormatException e) {
     Log.e(JerseyDisplayActivity.JDA, "Invalid number
format");
}                  }
```

- Get the mJerseyIsRedToggleButton isChecked value
- Create an intent
  - Load the String, int, and boolean extras
    - putExtra BasketballJersey.KEY_PLAYER_NAME
    - putExtra BasketballJersey.KEY_PLAYER_NUMBER
    - putExtra BasketballJersey.KEY_JERSEY_COLOR
- setResult to RESULT_OK with the intent
- finish

Once you handle loading up the return intent the final step will need to happen with the onActivityResult function of the JerseyDisplayActivity.java activity.

Not much to test until we implement the final part of reading these returned values.

Part E: Basketball Jersey's onActivityResult

The final step is to read the values from JerseyInfoActivity. Since we used the startActivityForResult method to launch that activity, we will get a call to onActivityResult when if finishes. Your onActivityResult method should check the request code sent back and the result code.

```
Check the request code for REQUEST_CODE_JERSEY_INFO
    Check the result code for RESULT_OK
        Get value from data via getStringExtra for KEY_PLAYER_NAME
        Get value from data via getIntExtra for KEY_PLAYER_NUMBER
        Get value from data via getBooleanExtra for KEY_JERSEY_COLOR
        call updateJerseyInfo()
```

When you get this method working, test, test, test. (At least pass the tests at the start of the lab!)

Congrats! It's a bit longer than it looks. :)

For your demo, show that it passes tests like the ones at the start of the lab. Invalid #; valid color, name, and # change; ignore cancel.