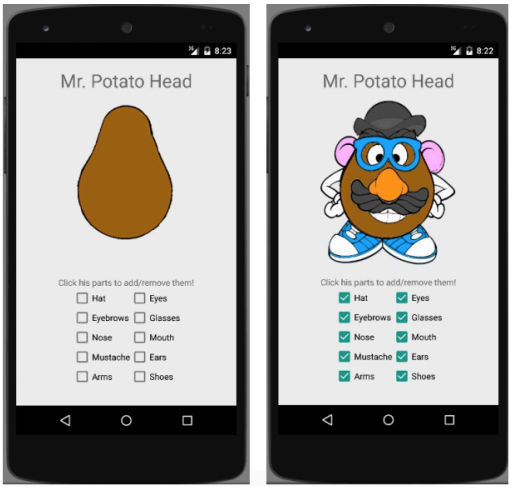
Homework 1

Layout The purpose of this assignment is to practice more advanced graphical layouts than you did in Homework 1, as well as using more widgets/views in a graphical user interface. We'll suggest some ideas for programs you could write to practice layout. We will list several suggestions, but you only need to write one app to submit.

Suggestion 1 (small): Mr. Potato Head

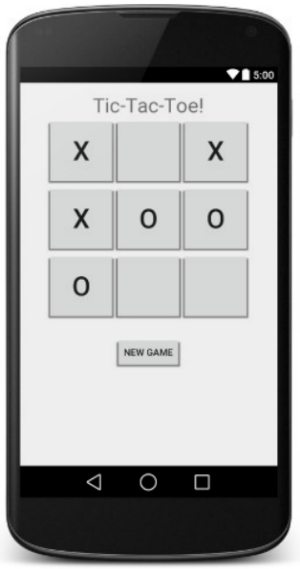
Write an app that displays a "Mr. Potato Head" toy on the screen as an ImageView. The toy has several accessories and body parts that can be placed on it, such as eyes, nose, mouth, ears, hat, shoes, and so on. We will provide you with image files for each body part and accessory, such as body.png, ears.png, hat.png, and so on. Initially your image view should display only the toy's body, but if the user checks/unchecks any of the check boxes below the toy, the corresponding body part or accessory should appear/disappear. The way to display the various body parts is to create a separate ImageView for each part, and lay them out in the XML so that they are superimposed on top of each other. You can achieve this with a RelativeLayout in which you give every image the same position, though you should probably nest it in some other overall layout for the screen. The check boxes should align themselves into a grid of rows and columns. You can set whether or not an image (or any other widget) is visible on the screen by setting its android:visibility property in the XML, and/or by calling its setVisibility method in your Java code. The setVisibility method accepts a parameter such as View.VISIBLE or View.INVISIBLE. There is also a getVisibility method if you need to check whether a widget is currently visible.



Suggestion 2 (medium): Tic-Tac-Toe

Write a basic game of tic-tac-toe, where two players take turns pressing buttons in a 3x3 grid to mark "X" or "O" characters on them respectively. If any player can place three of their letter in a row horizontally, vertically, or diagonally, that player wins the game. Setting up the buttons for the game is a good opportunity to practice using GridLayout. You'll probably want to set the buttons to have a large size so that they fill a large portion of the screen, as well as giving them a large font to make them easier to read and click. If you want a simpler

implementation, you can write your code as though two human players were playing it on the same screen; the first tap is an X move, the second is an O move, and so on. If you want more challenge, you could have the computer play as the second player. A simple strategy would be to just randomly move on any open square, but a more complex computer player would try to "block" the human player if the human has any two-in-a-rows and is one move away from winning the game.



Suggestion 3: Make Up Your Own

If you don't like our suggested assignment ideas or prefer to do something unique of your own, please feel free to do so. Whether you do our suggestions or your own, we'd prefer to see an app that has the following qualities:

- Your app should be set up as an **Android Studio** project, so it can easily be opened/run/graded by others.
- Your project should not always use the default names. (For example, rather than calling your project the default name of MyApplication and your activity the default name of MainActivity, call your project something like Hangman and your activity something like HangmanMainActivity, etc.)
- Your app should use at least **2 different layouts**. (Examples: a LinearLayout with another LinearLayout inside it; or a RelativeLayout with a GridLayout inside it; etc.)
- Your layout XML should adjust the **box model** properties of at least one widget or view. (Example: setting the gravity, weight, padding, margins, etc. of a widget to adjust its appearance or spacing.)
- Your app should respond to at least one event. (Example: clicks on a button.)
- Along with your app, please turn in a file named **README.tx**t that contains your name and email address along with the name of your app and a very brief description of it, along with any special instructions that the user might need to know in order to use it properly (if there are any). For example:

  Joe Student NumberGame 2.05 - This app shows two numbers on the screen and asks the user to pick the larger number.