Hello App Inventor!: The Homework Excuse Generator

By Paula Beer and Carl Simmons, authors of [Hello App Inventor!]

Even if you've never programmed before, this article will show you how to create an Android app for generating a unique excuse if you forget to do your homework.

Purpose of this App: This app generates a unique excuse if you forget to do your homework. It uses four hidden lists each with 10 elements. There are 10,000 possible combinations of excuses some are sensible – but most are very silly!

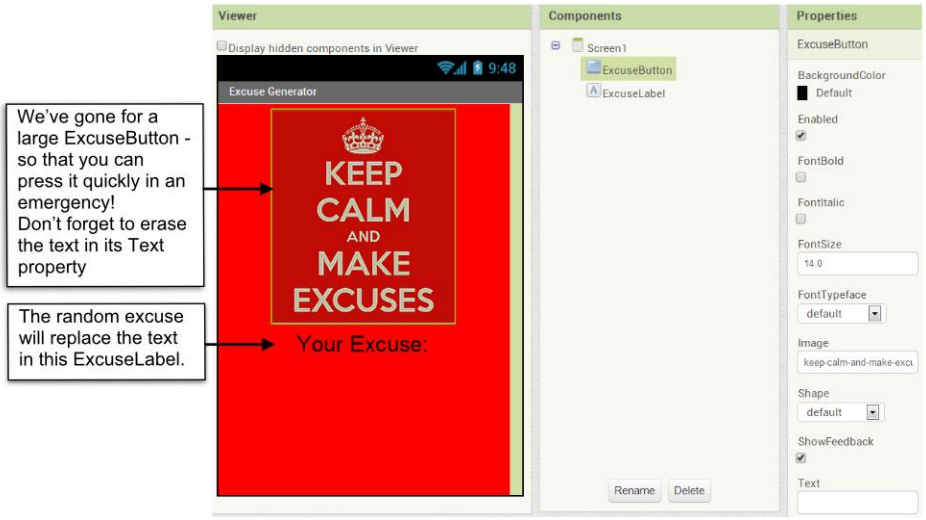
Assets you'll need: keep-calm-and-make-excuses.png (The png file is available from manning.com/beer - downloads section Chapter 6.)

Start a new project - "ExcuseGenerator".

Excuse Generator App			
Screen1 Properties:	Title: Excuse Generator AlignHorizontal: Center	BackgroundColor: Red Screen Orientation: Portrait	
Components	**What do I rename it?**	**What does it do?**	**What Properties do I set?**
Button From Palette: Basic	ExcuseButton	Produces a new excuse each time it is clicked.	Image:"keep-calm-and-make-excuses.png" Text: none
Label From Palette: Basic	ExcuseLabel	This is where the excuse will appear.	BackgroundColor: Red FontSize: 28 Text: "Your Excuse:"

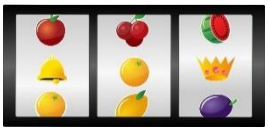
Setting up the Screen

Here's the screen layout; every time you press the button a new excuse will appear at the bottom of the screen:



Coding the Blocks – A Slot Machine Algorithm.

The rules (or steps) that programmers think about, and then turn into code are called an "algorithm." In this case we are going to use an algorithm that works like a slot machine - symbols spin around on three wheels and if three match, then you get a payout.



In this case, we'll have four wheels (lists) and, instead of symbols, we'll write parts of a sentence that make up an excuse. As long as we write the sentence parts so that they follow the same structure and fit together when we read them, then we can mix and match any combination of the four lists. The sentence structure must always be the same, it starts with an apology then a person doing some kind of action in a location like so:

Sentence part List:	**Apology**	**Person**	**Action**	**Location**
Example:	I'm sorry,	my sister	set fire to my homework	at the swimming pool.
Example:	A thousand apologies,	my dog	ate my homework	in the Principal's office.

If we write 10 elements for each list (Apology, Person, Action and Location), then the number of possible combinations is 10 X 10 X 10 X 10 = 10,000 sentences.

Learning Point – Algorithms

An "Algorithm" is a precise set of instructions that solve a problem. Algorithms exist outside of the world of computer programs, you'll find them everywhere - in math, science, engineering, english, art and even in your day to day life. Here are some examples of some algorithms that you have probably **come** across:

1. A recipe to bake a cake

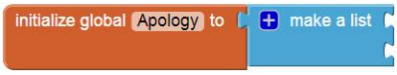
2. Working out the area of a shape
3. Folding a paper model like a bird or aeroplane
4. Following the rules of a game

These algorithms might appear in lots of different ways – a recipe is usually written down, to work out a math problem you might have written instructions or an example to follow; to fold a paper model, you probably followed a diagram or copied someone else; and to learn a game, you may have listened to instructions. Computer scientists do all of these things too when they want to figure out and share an algorithm – they might imagine it and talk it through, write down the steps or draw a diagram like a flowchart.

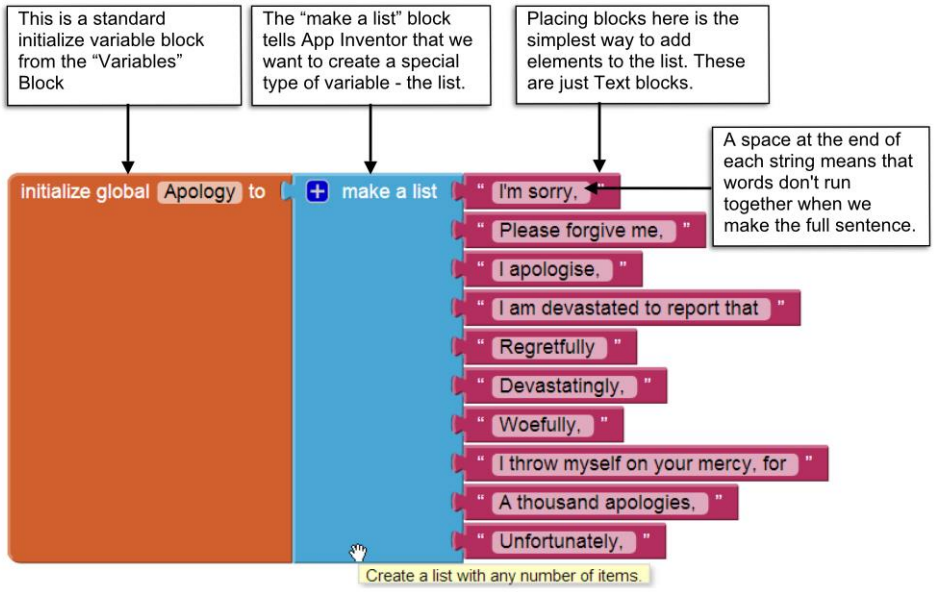
One thing that is often very important in computer algorithms is that programmers want to build an **efficient** algorithm. This is because computers often do millions of operations every second, so even a small change in the time to complete some instructions can make a big difference in how quickly the overall program or app runs. Think about it like this – saving 5 seconds when you bake a cake is no big deal, but make a million cakes and the overall saving is about 58 days!

Setting up the Lists

First set up the Apology list with 10 elements; this is almost the same as setting up any other variable. Just like with any other variable you start with an "initialize global" block from the "Variables" section of the blocks. Change the name of the variable to something sensible, like "Apology." Now instead of a number or text string we use a "make a list block" from the "Lists" section. Your blocks will look like this:



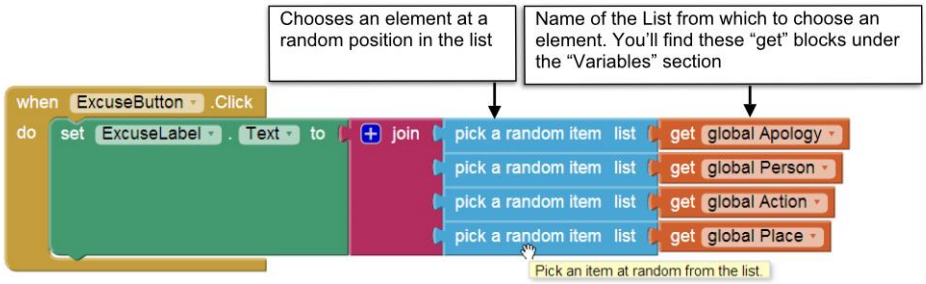
In some apps you might not know what the list will contain in future – for example if you were setting up lists to store users' contacts or high scores you can only fill up the list elements once the user is running the app. In that case, we would just stop there, leave this variable definition block as it is (with empty sockets after "make a list") and move on to programming the rest of the app. However in this app we are going to build the sentence elements right into the program – so we can add the 10 apologies as text blocks to the list like so:



Now we'll add the other three lists to the blocks – the fastest way is to copy and paste the block above three times change the variable names (to Person, Action and Location), and then change the text blocks. Feel free to change these or add additional text blocks to any of the lists. Here's the way ours looked when finished:

Building the Sentence

So now all of the hard work is done. When the ExcuseButton is clicked we need the app to choose a random element from each of the four lists and display them the sentence altogether in our ExcuseLabel. In some programming languages, we would do this by picking a random number and then choosing the element at that position (or **"index"**) in the list. App Inventor makes things even simpler by using a block called "pick a random item." You just tell it which list to use and it randomly selects an item from that list. So if we do that for the four lists and join the responses our blocks look like this:-



Try it out – each time the button is pressed you should see one of 10,000 possible excuses. You could use this as the basis for a game with friends where each player presses the button and then has to explain what happened to their homework starting with that sentence. The more plausible, creative and imaginative they are, the more points you should give them.

You could enhance the app by having it play some sad music as you give your excuse – maybe it will sway your teachers. You could also have the text-to-speech component read out the excuse (we guarantee it will sound more plausible than you – because it won't snicker or laugh).

You can find instructions for creating a bunch of other fun apps in the book Hello App Inventor!, by Paula Beer and Carl Simmons.