**Captcha**

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a type of security measure known as 'challenge-response authentication.' *(courtesy Wikipedia)* Captchas can be annoying but are necessary to prevent *botting*.

Unscrupulous programmers write scripts to (for example) buy highly sought-after tickets to concerts/sporting events/etc. faster than humans are able to for the purposes of re-selling them on the grey market. Captchas aim to stop botting by requiring a human response to the 'challenge.'

In this project you will make a simple captcha application using some supplied assets.

1. See my hastily coded version [here](https://www.loom.com/share/3abf6bfc417846589f24152d24c8f18b) for generally how this project should work (yours does not need to look like mine).
2. In a new Swift file, create a UIImageView sub-class called CaptchaImageView. Add the following function to it:

func shake() {

let animation = CABasicAnimation(keyPath: "position")

animation.duration = 0.05

animation.repeatCount = 3

animation.autoreverses = true

animation.fromValue = NSValue(cgPoint: CGPoint(x: self.center.x - 4.0, y: self.center.y))

animation.toValue = NSValue(cgPoint: CGPoint(x: self.center.x + 4.0, y: self.center.y))

self.layer.add(animation, forKey: "position")

}

This sub-class adds a reusable 'shaking' functionality to a plain UIImageView. Sub-classing allows you to add custom functionality (e.g. graphics or behaviors) to standard UI components, though at the moment all you are adding is the shake function (you may add other properties later).

1. Add four UIImageView objects attached to the CaptchaImageView class, laid out with appropriate stacks/constraints. Have the buttons display captcha images (chosen randomly from the provided files). Choose a 'correct' image at random, then prompt the user to select the image with the correct text (rather than entering the text of the image and checking what the user entered).
   1. If the user chooses correctly\*, transition to the next screen with *audio* captchas (more info to come).
      1. \*Note that UIImageView objects don't natively respond to user interaction, like e.g. a UIButton does. You will need to add a UITapGestureRecognizer to each image.
   2. If the user chooses *incorrectly\*\*,* shake the button and give them another attempt with four new images.
      1. \*\*If the user chooses incorrectly two times, transition to a very rude screen calling them out for totally being a bot.
2. Create a custom UIButton sub-class called CaptchaButton. Copy the shake function to this class.
3. On the second screen, allow the user to play an audio captcha (chosen at random), then require the user to click the UIButton containing the correct text. Shake the button if the user chooses incorrectly.
4. If the user answers both correctly, transition to another *super cool, super secret screen* of your choosing.
5. Copying/pasting the shake function two separate places is a [*code smell*](https://martinfowler.com/bliki/CodeSmell.html)- if you wanted to make changes to the way the function works, you'd have to do so in multiple locations; not bad now, but imagine you were working on a large project with hundreds of files including dozens of shakeable UI components. Further, there would be no indication of *which* files intend to shake - you would have to look through the entirety of each type's source.
6. Refactor the project to use a protocol called Shakeable (defined in its own Swift file).
   1. Shakeable should work for any UIView and should contain a single function called shake.
   2. **(Optional)** Add a default implementation to the Shakeable protocol (allowing sub-classes to override how it works if they want, or use the default implementation if they don't care) with a protocol *extension*. Protocol extensions won't be covered until later in the book.
   3. Make both sub-classes you created conform to the Shakeable protocol. This is better for a couple reasons:
      1. It's now more explicit and readable that your sub-classes intend to shake:

**class** CaptchaImageView : UIImageView, Shakeable {

Rather than having to scour their code for a random function, you can immediately tell from the class header that they are intended to be shakeable.

* + 1. If, in the future, you decide that these classes should *not* be able to shake, you can simply remove the conformance.

**Extension ideas**

* Programmatically generate your own captchas by adding some type of noise to text. Humans should be able to figure out what the text says easily, but the text should be hard to parse for computers.
* Try to beat the text/image captcha with OCR (optical character recognition) and possibly machine learning.
* Try to beat the audio captcha by parsing the sound in some way.
* Pull images from a web database for image captcha, e.g. 'choose all the pictures that show a boat.'