Lab 2

iOS: Evil Hangman Walkthrough

Evil Hangman

- Overview
- Equivalence Classes
- Setup
- Getting Input
- Property Lists
- Settings

Overview

Goal: Actively try to keep the player from winning the game.

Strategy: Select the longest list of possible words after each guess.

Equivalence Classes

- Start with a list of words:
  CAT
  COT
  COW
  DOG
  DOT
  TAG
  TOT

Equivalence Classes

- Check the list of words against player's guess:
- Guess: T

  CAT => --T
  COT => --T
  COW => ---
  DOG => ---
  DOT => --T
  TAG => T--
  TOT => T-T

- Possibilities: ---, --T, T--, or T-T

Equivalence Classes

- Pick the largest of the equivalence classes:

  --T => 3 (CAT, COT, DOT)

  --- => 2 (COW, DOG)

  T-- => 1 (TAG)

  T-T => 1 (TOT)

Equivalence Classes

- Repeat with the new list:
- Guess: C

  CAT => C-T

  COT => C-T

  DOT => --T

- Pick the largest equivalence class:

  C-T=> 2 (CAT, COT)

  --T => 1 (DOT)

Equivalence Classes

- Repeat with the new list:
- Guess: A

  CAT => CAT

  COT => C-T
- Pick the ***best*** equivalence class:

  COT => 1 (C-T)

  CAT => 1 (CAT)

Equivalence Classes

That’s great, but how do we actually implement this?

Equivalence Classes

http://en.wikipedia.org/wiki/Equivalence_class

The equivalence class of an element *a* is denoted [*a*] and may be defined as the set

$$[a] = \{x \in X \mid a \sim x\}$$

of elements that are related to *a* by ~.

Equivalence Classes

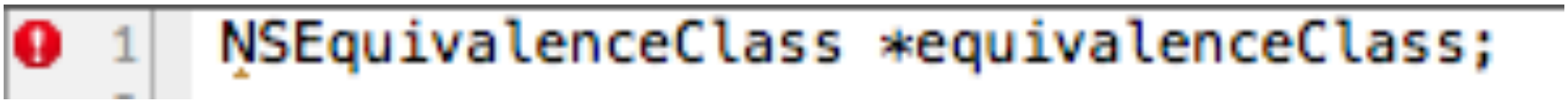
Roughly translated:

- Define a set of words sharing a given letter at a given location
- Order matters: - - T != T - -

Equivalence Classes

What about in code?

- Xcode has no idea what we're talking about:



Equivalence Classes

- What are our options?
  - `NSMutableDictionary`
  - `NSMutableArray`
  - `NSMutableSet`

Equivalence Classes

Pseudocode:

```
for each word in set:
    determine equivalence class for word
    add word to equivalence class
determine largest equivalence class
remove all other words
update UI
```

Equivalence Classes

- Each equivalence class contains a set of words
- Need to keep track of all equivalence classes
  - Always pick the largest class
  - Break ties in a pseudo-random manner

Equivalence Classes

Considerations:

- Time
  - Iterating over lists is slow
  - Indexing into arrays and dictionaries is fast
  - Loading plists from storage is slow
- Space
  - `words.plist` is pretty big
  - Keep your data structures as small as possible
  - Don’t keep things in memory longer than needed

Equivalence Classes

- “We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3%.” – Donald Knuth

(http://en.wikipedia.org/wiki/Donald_Knuth)

Equivalence Classes

Summary:

- Leave yourself the most number of options at each stage.
- If at all possible, do not let the player win
- If possible, optimize the algorithm further ☺

Setup

- Utility Application
  - Contains two controllers:
    - `MainViewController`
    - `FlipsideViewController`
  - Flipside is often used for configuration / settings
  - e.g. Apple Weather App

Setup – Delegates / Protocols

- Identifies an object to handle an action on your behalf, rather than handling it yourself
- `delegate` object implements a protocol
  - Guarantees that methods will be implemented

Setup - Protocol

```
@protocol SomeProtocol
- (void)someMethod;
- (int)calculateSomething:(int);
@end
```

Setup

DEMO: Utility Application

Getting Input

- Must come from a `UITextField`
- `UITextFieldDelegate` provides additional functionality
- Text field should probably be hidden

Getting Input - UITextFieldDelegate

- `textFieldShouldReturn:`
  - "Done" button pressed
- `textFieldShouldBeginEditing:`
  - User about to edit text
- `textFieldShouldEndEditing:`
  - Text field about to lose focus

Getting Input

DEMO: UITextField

Property Lists

- Key/value pairs
- Stored in XML
  - Can be edited manually, or with the plist editor
- Frequently used to store settings

Property Lists

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" http://www.apple.com/DTDs/PropertyList-1.0.dtd>
<plist version="1.0">
  <dict>
    <key>KeyName</key>
    <string>KeyValue</string>
  </dict>
</plist>
```

Property Lists

- `NSDictionary` has a method: `initWithContentsOfFile:`
  - Can also write plist files
- `NSBundle` has a method: `pathForResource:ofType:`
  - Provides access to the filesystem

Property Lists

DEMO: Property Lists

Settings

- `NSUserDefaults` has useful methods:
  - `registerDefaults:`
  - `integerForKey:`
  - `setInteger:forKey:`
  - `stringForKey:`
  - `setObject:forKey:`
  - …