public presentation

{

Paradigms.of(Programming)

}

//@Authors: Adam Whittingham
// Anna Kennedy
//Original Session: Steve Freeman
// Michael Feathers
//Original Material: http://github.com/sf105/paradigms-of-programming

An Inconvenient Truth:

There is no perfect language.

(nope, not even Ruby)

The Truth:

We need to think less about the **languages**

and more about the **paradigms**

The Truth:

- **Languages** are tools for expressing **paradigms**
  - Procedural , OO, Functional, Rule based, constraint based…
  - Languages can support other paradigms (DSLs,)
- Knowing different **paradigms** helps us
  - Broadens our view or what the solution might look like
  - Lets us use our languages in more imaginative ways

Paradigms?

- Robert Floyd –Turing Award acceptance, 1978

  "If the advancement of the general art of programming requires the continuing invention and elaboration of paradigms, advancement of the art of the individual programmer requires that he expand his *repertory* of paradigms"

Paradigms?

- Robert Floyd –Turing Award acceptance, 1978

  If the advancement of the general art of programming requires the continuing invention and elaboration of paradigms, **advancement of the art of the individual programmer requires that he expand his repertoire of paradigms**

Today we advance!

Procedural!

Functional!

Rule Based!

Object Orientation!

How this will work?

- There will be 4 iterations of 15 minutes (agile!)
- There's a problem (isn't there always?)
- Each iteration extends the problem
- Each table has a set paradigm
- Each team solves this iterations problem in the paradigm of the table they are at

BUT WAIT! There's more!

- After each iteration everyone rotates round to the next table

EXCEPT FOR ONE PERSON

- This person is the maintainer
  - The maintainer explains the thoughts and designs of the previous developers to the new dev team

One last thing…

After the first round

THE MAINTAINER MUST MOVE TABLES

That's right- after the 2nd round none of the original dev team will be left and you'll be maintaining someone else's code.

(Sound familiar?)

The Problem

- We're going to make a list box for a GUI
  - Displays a **list of items** on screen
  - Has a **selected** item
  - Users can go **up** or **down** the list

- Work in Pseudo-code
  - We're not rendering things or using any frameworks!
  - Focus on the paradigm we're using

The Problem: Part 1

- The list box is initialized by a list of items. When initialized, the current selection is the first element in the list.

- Create 2 operations: **`arrowUp`** and **`arrowDown`**, which allow you to change the position of the current selection.
  - If you **`arrowUp`** when the first element of the list is selected, nothing should happen (no-op).
  - If you **`arrowDown`** when the last element of the list is selected, nothing should happen (no-op).

- If the controller is given an empty list, there is no currently selected item.

Remember, in later iterations we will expand this behaviour.

Time for a change!

- Select 1 person to be your maintainer

  (Someone who understands what you've just done might help!)

- Ensure the maintainer has **all** the source code!

- Everyone else, move clockwise to the next table

The Problem: Part 2

- Add the concept of the "`window`", the portion of the list box which is currently displayed on the GUI.
  - Imagine a list box with 100 elements. The window may be showing elements [10..19].

- The `arrowUp` and `arrowDown` operations have special behaviour at the top and bottom of the window- they move it.

  For example: If the window is showing elements [10..19] and 10 is selected:
  - `arrowUp` changes the current selection to 9 and shifts the window up to show [9..18].
  - `arrowDown` changes the current selection to 11.

Note: You **do not** have to deal with the cases where the **window** touches the top or bottom of the list. (We have to save some fun for next time!)

Change Places!

- Select 1 person to be your maintainer.

This MUST NOT be the person who was the maintainer this round!

- Ensure the maintainer has all of the source code!

- Everyone else, move clockwise to the next table

The Problem: Part 3

- Account for the window hitting top and bottom of the list
  - An `arrowUp` operation at the top of the window when the window is showing [0..x] is a no-op.
  - An `arrowDown` operation at the bottom of the window when the window is showing [x..last] is a no-op also.

- The windows size should be represented by `windowSize`.
  - The default `windowSize` is 10
  - If the list size is less than 10, then `windowSize` becomes the size of the list.

- **Refactor** the code to improve its structure.

One more time…

- Select 1 person to be your maintainer.

This MUST NOT be the person who was the maintainer this round!

- Ensure the maintainer has all of the source code!
- Everyone else, move clockwise to the next table
- Psych yourselves up for the big finale!

The Problem: Part 4

- Add the operations **pageUp** and **pageDown**:

- **pageUp** moves the window up **windowSize** elements so that it is just above it's previous first element.
  - When the operation is complete, no element which was visible in the window before is visible in the window afterward.
  - If **pageUp** can not go up **windowSize** elements because it would hit the top of the list, it stops at its last possible move up.
  - The current selection after a **pageUp** is the last element in the window.

- The **pageDown** operation has the exact opposite behaviour. It moves the window downward, and makes the current selection the top element of the window.

How did it go?

- **How did you find working with different paradigms?**
- How did you find maintenance?
- Which did you find easiest to extend?
- What did you get stuck on?
- Did you get to refactor? If so, how did it go?
- How easy was code reuse for pageUp & pageDown?

Almost done…

- Many thanks to Michael Feathers & Steve Freeman for creating the original session & materials!
- Thank you Anna for all your help preparing & running the session!
- Our version of this presentation will be shared-alike, URL will be posted to the DevCon3 wiki page after the event.

Thank you

for participating!