Application Patterns and Structures

July 21, 2017

Review Quiz

What is the most important thing to beware of w/ event loops?

A. Blocking the thread
(since it prevents other queued up events from firing)

B. Allocating too much memory
(since it can kick other events out of memory)

C. Allocating too many event handlers
(since it can overwhelm the event queue)

D. Modifying global variables
(since it can cause instability between events)

Which problem **does not** lend itself well to an event loop?

A. IO intensive applications (e.g. web servers)

B. UI intensive applications (e.g. mobile apps)

C. Computation intensive applications (e.g. data processing)

D. Timer intensive applications (e.g. cron-like applications)

Which **is not** an effective method to prevent blocking the event loop?

A. Moving the blocking operation to a helper method

B. Breaking long-running task into many smaller, quicker parts

C. Moving the blocking operation to a subprocess

D. Moving the blocking operation to a server, and calling it with a network request

Which **would not** be a plausible event in an event loop system?

A. "onFileRead"

B. "onTimerComplete"

C. "onNetworkRequestFailed"

D. "onFunctionReturned"

When would a subprocess be a better tool than a thread?

A. When you're composing functionality by combining existing programs

B. When IO thru-put is more important than speed

C. When execution time is more important than correctness

D. When you're working in an interpreted language, instead a compiled one

Done!

Final Project

Part 1: Planning

- 20 points total
  - 5 points: initial proposal
  - 5 points: first progress report – 7/19/2017
  - 5 points: second progress report – 7/24/2017
  - 5 points: final progress report – 7/28/2017

Part 2: Code Contribution

- 40 points total
  - Correctness
  - Good coding standards
  - Documentation
  - Unit tests
  - Problem difficulty

Part 3: Presentations

- 10-15 minutes
- 20 points
  - Describing the project / application
  - Demo your contribution
  - Application of class topics
  - Discussing difficulties

Progress?

A. I have not decided on a project yet

B. I have started planning my contribution, but haven't written any code yet

C. I have started writing code, but its not complete

D. I have completed the code part of the project

Possible Next Steps

- Submitting the issue?
- Submitting the application to Maven / Google Play / etc
- I will work with you after the class is done, if desired

Patterns in Software Development

What Are Patterns

- Idioms / common practices / templates for problems
- Application structure (today)
- Solution structure (Monday)
- System structure (Wednesday)

Why Patterns

- Any organization is better than none
- Common organization patterns reducing "learning" time
- Some patterns steer you away from problems

scratch/NoOrg.java –>

NoOrg Example

- What if we want to change the output type?
- What if we want to use the same output code, but describe URLs?
- What if we wanted to write tests?

Model-View-Controller

Model-View-Controller

- Common application pattern
- Split up applications into three parts
  - **Controller**: handle and receive the request
  - **Model**: represent the data type(s) worked with
  - **View**: Control the presentation to the user

Flow diagram of MVC project –>

Common MVC Problem Domains

- Web applications
  Present data as HTML, JSON

- Mobile apps
  High need to reuse UI code

- Command line applications
  Single interface for many data types

Controller

- Each controller receives a type of request
- (Possibly) preprocess the data
- Interact with **model** classes to modify / build data
- Interact with **view** classes to build output
- Smallest part of the system

Model

- Represents a type of data
- Hides data implementation from **controller** classes
- One model for each type of data interacted with
- Granularity varies
  - One class for the entire database?
  - One class for each thing stored in the database?

View

- Represents a way of presenting data
- Receives data to present from **controller** class
- Avoid tying to **model** classes
- Wide range of possibilities
  - HTML, CSV, gz, bz, JSON, XML
  - (ie everything)

scratch/NoOrg.java –>

hw4 –>

Overview

- MVC is a very common application pattern
- Emphasizes splitting a project into code to deal with
  - requests (controller)
  - data interaction (model)
  - presentation (view)