CS 596 Functional Programming and Design
Fall Semester, 2014
Doc 1 Introduction
Aug 26, 2014

Course Overview

Course Issues

http://www.eli.sdsu.edu/courses/index.html

Crashing

Course Web Site

Wiki

Screencasts

Prerequisites

Grading

Clojure

Crash Policy

Class limited to 70

By seniority

- Measured by SDSU credits on transcript
- Alternate undergrad and graduate students
- 2 undegrad students for each graduate student

Provide unofficial transcript

- Hardcopy or softcopy via email
- Available for free in SDSU portal
- Need at least two hours before start of class

Crash Policy

SDSU students have priority over Open University students

CS majors have priority over non-cs majors

Crash Policy

Start adding students in class Thursday

So don't ask for add code an end of class today

No transcript by Thursday assume you have 0 SDSU units

Must be present to get add code

If miss a class then I drop you from crash list

Crash List FAQ

Why not get a bigger room and admit everyone?

No first hard assignment to scare people

No Grader

Do you really want a 500 level class of 120 people?

Crash List FAQ

Sept 8

Last day for regular students to add/drop classes

Open University students have lower priority than SDSU students

Will this be a difficult course?

You need to change how you solve problems in code

Yes very hard

```
publc float fizzle()
    x = y.foo();
    z = x.bar();
    if (z.cat() > 0)
        bat = x.index();
    else
        bat = 5;
    return bat;
}
```

f(g(x,y, h(z k(w))))

Data structures are immutable

For Thursday

Watch "Simplicity Matters" by Rich Hickey

http://www.youtube.com/watch?v=rI8tNMsozo0

ClojureScript Koans

Download Light Table

Goal

Understand the common features of functional programming

Know how to use features of functional programming

Become comfortable using functional programming

Build well designed functional programs

Growing Interest in Functional Programming

Blogs

Books

Banks & Start-up using Functional Programming

Languages adding functional features

- Java
- C++
- Swift

John Carmack

No matter what language you work in, programming in a functional style provides benefits.

You should do it whenever it is convenient, and you should think hard about the decision when it isn't convenient.

http://www.altdev.co/2012/04/26/functional-programming-in-c/

Facebook

Backend of chat service - Erlang

React.js

Immutable.js

Whats App

Bought by Facebook

Uses 1/3 number of engineers per user base

Uses Erlang for backend

Erlang

Functional programming language

Developed by Ericsson

Used in phone switches

Ericsson achieved 99.9999999% reliability with Erlang

- Machine started when your parents were born
- Would have 1 second of downtime

You can't do that with Java

Some Other Users of Erlang

Amazon's	SimpleDB
Yahoo's	Delicious
T-moble	SMS

Some Users of Clojure

Netflix
Citigroup & Other Banks
Akamai
Walmart

Lot of smaller companies

- Appvise
- Berico Technologies
- Compass Labs
- Dynamic Animations Systems
- Factual, Inc
- FlightCaster
- Geni
- Infinitely Beta
- KamaGames

http://dev.clojure.org/display/community/Clojure+Success+Stories

Why the Interest in Functional Programming

Concurrency

One style of programming does not fit all situations

Object-Oriented programming has matured

Problems with Object-oriented programming

Why Clojure

Practical language

Wanted pure functional language
- No imperative/oo programming

Clojure has access to Java
- But controlled access

Clojure, Java & Assignments

Clojure has special syntax to call Java code

You can

- Create Java objects
- Call methods on objects

So you can avoid Functional programming

Point of course is to learn Functional programming

You are not allowed to use Java code in assignments

Problems that use Java receive zero points

What is Functional Programming

Elements of Functional Programming

Pure Functions

First Class Functions

Higher-Order Functions

Immutability

Lazy Evaluation

Recursion

Currying

Memoization

Destructuring

Collection Pipelines

List Compressions

Pure Functions

Functions with no side-effects

Only depend on arguments

Don't change state

```
class Foo {
    int bar

    public int notPure(int y) {
        return bar + y
    }

    public void alsoNotPure(int y) {
        bar = y
    }
```

Why important

Easier to
- debug
- test
- understand program

First Class Functions

Functions can be

- Assigned to variables
- Passed as arguments
- Returned from functions

Anonymous functions

- Lambdas
- Closures

Why important

- Flexibility
- Generality

Higher-Order Functions

Functions that operate on functions

Why important

Fewer details/
higher level logic

Concurrency

Immutability

Data structures can not be modified

Like Java's Strings

Why important

Concurrency

Easier to
- debug
- test
- understand program

Lazy Evaluation

Operations & functions evaluated

- When used
- Not when called

Why important

Simplifies logic

Recursion

```
function factorial(n)
    if n = 1 return 1
    return n * factorial(n-1)
```

Why important

Powerful tool

Tail recursion/Tail Call Optimization

When last statement is just the recursion

Compiler can convert recursion into loop

Currying

```
function add(int x, int y) {
    return x + y;
}

addTen = add(10);

addTen(3)  //returns 13
```

Why important

Memoization

Cache value of functions

memoize(factorial)

```
factorial(1000)    //1000 recursive calls
factorial(1001)    // 1 recursive call
```

Why important

Performance

Collection Pipelines

```
String[] words = {"a", "ab", "abc", "abcd", "bat"};
List<String> wordList = Arrays.asList(words);
List<String> longWords;
longWords = wordList.stream()
                        .filter( s -> s.length() > 2)
                        .filter( s -> s.charAt(0) == 'a')
                        .map( s -> s.toUpperCase())
                        .collect(Collectors.toList());
```

Why important

Higher level logic

Concurrency

For Thursday

Watch "Simplicity Matters" by Rich Hickey

http://www.youtube.com/watch?v=rI8tNMsozo0

ClojureScript Koans

Download Light Table