Functional Programming

in Racket

Peter Campora

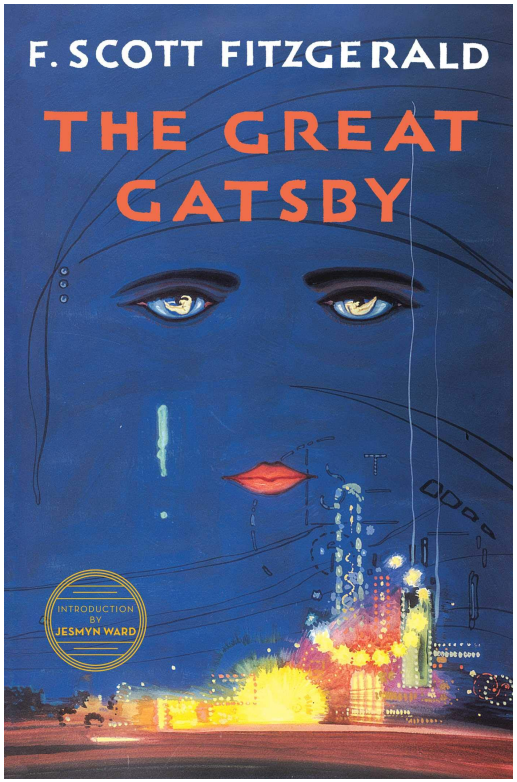
ULL

July 17, 2019

What Things are Beautiful?

What Things are Beautiful?

What Things are Beautiful?



Can Code be Beautiful?

Write a program for summing the squares of numbers from 1 to n.

Can Code be Beautiful?

Write a program for summing the squares of numbers from 1 to n.

```
public class SumSquares
{
   public static void main(final int[] nums)
   {
      int sum = 0;
      for (int i : nums)
        sum += i * i;
      System.out.println("The sum of the squares is: " +
↪   sum);
    }
}
```

An Equivalent Program

Let's get our first look at a functional program written in Racket.

An Equivalent Program

Let's get our first look at a functional program written in Racket.

```
;;Number -> Number
(define (square x) (* x x))

;;Number List -> Number List
(define (sum-of-squares lst)
  (sum (map square lst)))
```

An Equivalent Program

Let's get our first look at a functional program written in Racket.

```
;;Number -> Number
(define (square x) (* x x))

;;Number List -> Number List
(define (sum-of-squares lst)
  (sum (map square lst)))
```

Don't worry about the meaning and syntax of the program, yet. Given time, the parens will become beautiful!

What *is* Code Beauty?

Let's consider some philosophic views.

- Utilitarian Program Beauty (a more scientific view)

What *is* Code Beauty?

Let's consider some philosophic views.

- Utilitarian Program Beauty (a more scientific view)
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production

What *is* Code Beauty?

Let's consider some philosophic views.

- Utilitarian Program Beauty (a more scientific view)
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production
  - Critique: Hard to measure which features objectively help the process of writing programs.

What *is* Code Beauty?

Let's consider some philosophic views.

- Utilitarian Program Beauty (a more scientific view)
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production
  - Critique: Hard to measure which features objectively help the process of writing programs.
- Egoist Program Beauty (how most of us actually do it)

What *is* Code Beauty?

Let's consider some philosophic views.

- Utilitarian Program Beauty (a more scientific view)
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production
  - Critique: Hard to measure which features objectively help the process of writing programs.
- Egoist Program Beauty (how most of us actually do it)
  - The programming language that allows me to write clever, expressive programs is best. *The Haskell programmer viewpoint* ;p

What *is* Code Beauty?

Let's consider some philosophic views.

- Utilitarian Program Beauty (a more scientific view)
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production
  - Critique: Hard to measure which features objectively help the process of writing programs.
- Egoist Program Beauty (how most of us actually do it)
  - The programming language that allows me to write clever, expressive programs is best. *The Haskell programmer viewpoint* ;p
  - Critique: Engineering is largely a social endeavor. Other people need to be able to read your programs and understand abstractions.

What *is* Code Beauty?

Let's consider some philosophic views.

- Utilitarian Program Beauty (a more scientific view)
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production
  - Critique: Hard to measure which features objectively help the process of writing programs.
- Egoist Program Beauty (how most of us actually do it)
  - The programming language that allows me to write clever, expressive programs is best. *The Haskell programmer viewpoint* ;p
  - Critique: Engineering is largely a social endeavor. Other people need to be able to read your programs and understand abstractions.
- Aesthetic Program Beauty (an unfortunately ignored perspective)

What *is* Code Beauty?

Let's consider some philosophic views.

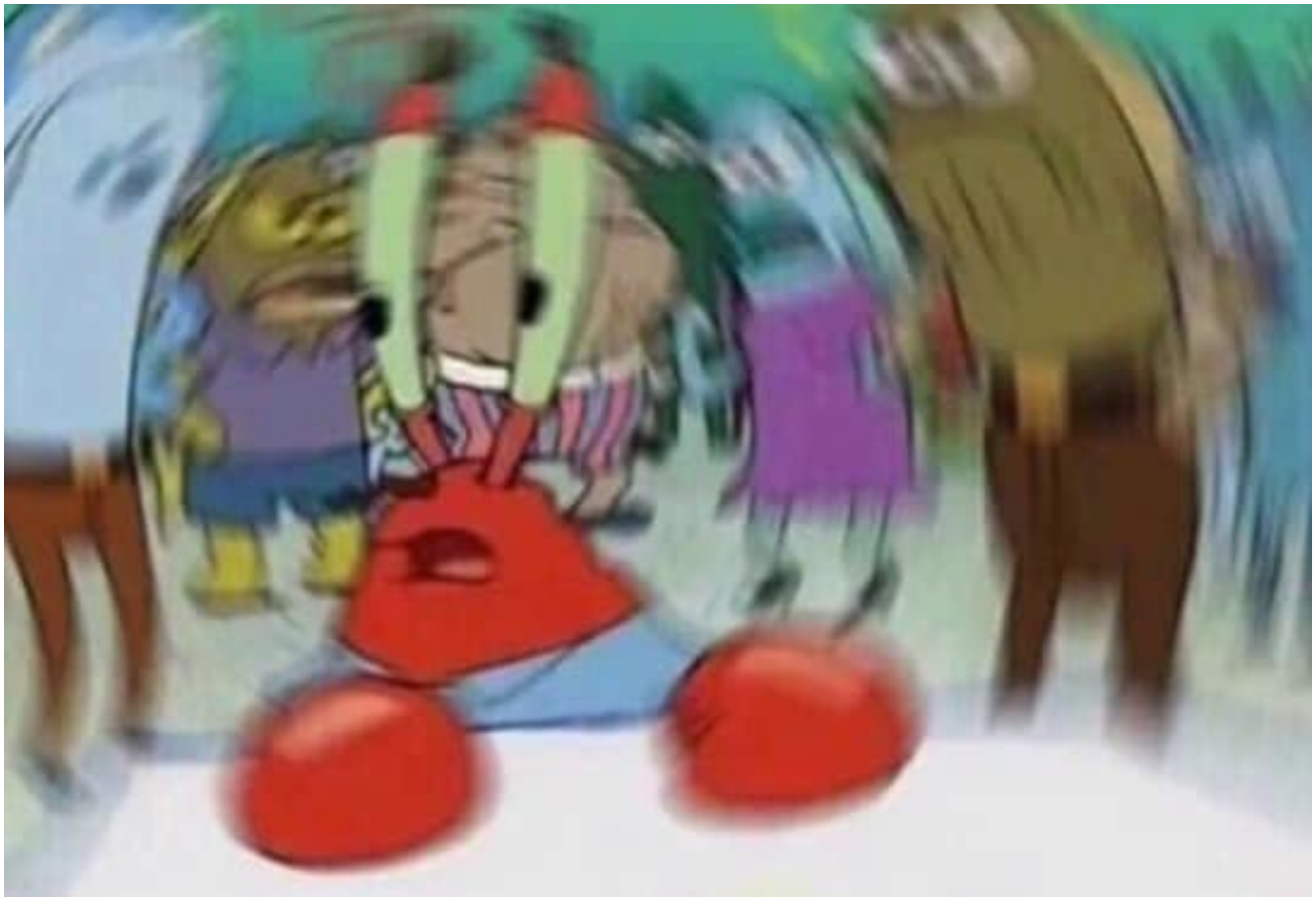
- Utilitarian Program Beauty (a more scientific view)
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production
  - Critique: Hard to measure which features objectively help the process of writing programs.
- Egoist Program Beauty (how most of us actually do it)
  - The programming language that allows me to write clever, expressive programs is best. *The Haskell programmer viewpoint* ;p
  - Critique: Engineering is largely a social endeavor. Other people need to be able to read your programs and understand abstractions.
- Aesthetic Program Beauty (an unfortunately ignored perspective)
  - Programming languages provide a framework for creating beautiful programs according to the idioms of some paradigm (i.e. cubism in art)

What *is* Code Beauty?

Let's consider some philosophic views.

- Utilitarian Program Beauty (a more scientific view)
  - The programming language that maximizes the utility of writing my program is the best. Minimize bugs, execution time, and time until production
  - Critique: Hard to measure which features objectively help the process of writing programs.
- Egoist Program Beauty (how most of us actually do it)
  - The programming language that allows me to write clever, expressive programs is best. *The Haskell programmer viewpoint* ;p
  - Critique: Engineering is largely a social endeavor. Other people need to be able to read your programs and understand abstractions.
- Aesthetic Program Beauty (an unfortunately ignored perspective)
  - Programming languages provide a framework for creating beautiful programs according to the idioms of some paradigm (i.e. cubism in art)
  - Critique: Disconnected from the empirical or mathematical viewpoints we typically employ to measure the value of scientific contributions.

Looks Like We Don't Know



So, Why Ask?

- Universities are a place for the creation and acquisition of knowledge.

So, Why Ask?

- Universities are a place for the creation and acquisition of knowledge.
- We should put our opinions under scrutiny.

So, Why Ask?

- Universities are a place for the creation and acquisition of knowledge.
- We should put our opinions under scrutiny.
- Computer Scientists are critical thinkers.

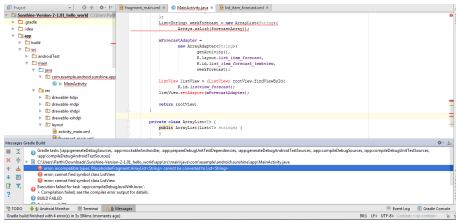
How do We Write Programs?

How do We Write Programs?

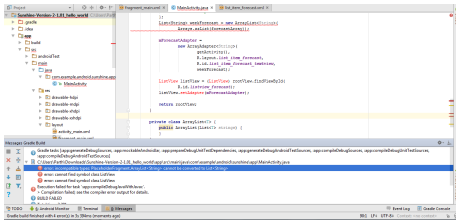
Let's hear from you.

Is Programming Hard?

Is Programming Hard?



Is Programming Hard?

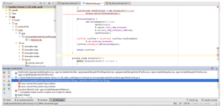


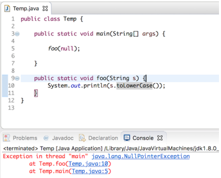
```
Temp.java
1  public class Temp {
2
3      public static void main(String[] args) {
4
5          foo(null);
6
7      }
8
9      public static void foo(String s) {
10         System.out.println(s.toLowerCase());
11     }
12 }
13
```

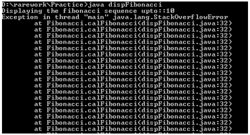
Problems Javadoc Declaration Console

```
<terminated> Temp [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_
Exception in thread "main" java.lang.NullPointerException
        at Temp.foo(Temp.java:10)
        at Temp.main(Temp.java:5)
```

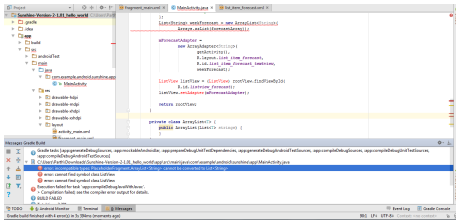
Is Programming Hard?

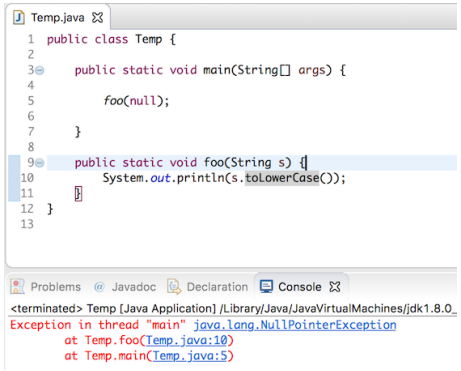




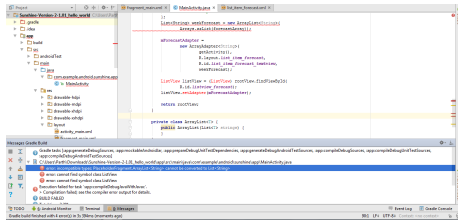


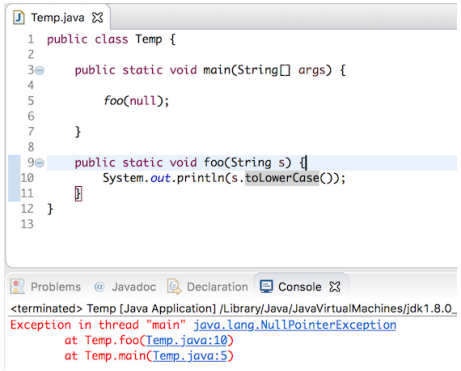
Is Programming Hard?





Is Programming Hard?





How Programming is Taught

- Programming courses typically start by motivating new features with concrete programs.

How Programming is Taught

- Programming courses typically start by motivating new features with concrete programs.
- Given enough examples, students begin to match problem descriptions to using features.

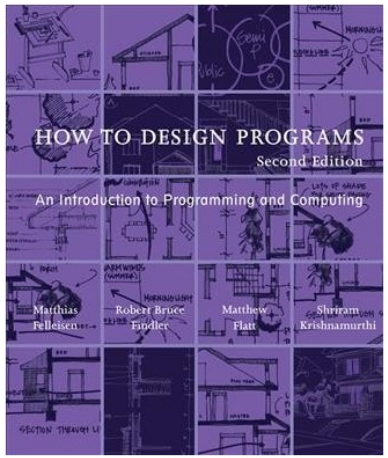
How Programming is Taught

- Programming courses typically start by motivating new features with concrete programs.
- Given enough examples, students begin to match problem descriptions to using features.
- From there, programming becomes gluing together recognized patterns.

How Programming is Taught

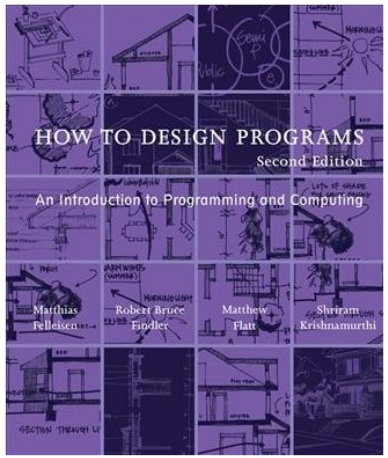
- Programming courses typically start by motivating new features with concrete programs.
- Given enough examples, students begin to match problem descriptions to using features.
- From there, programming becomes gluing together recognized patterns.
- **There has to be a better way!**

The Better Way



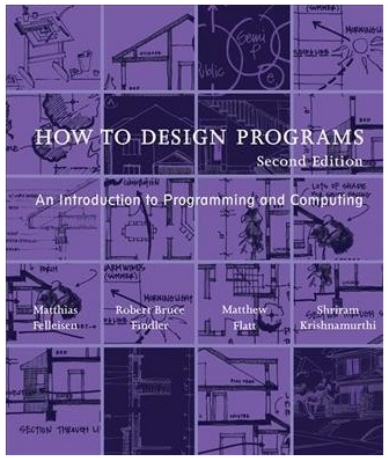
- This book (and course) will teach systematic program design.

The Better Way



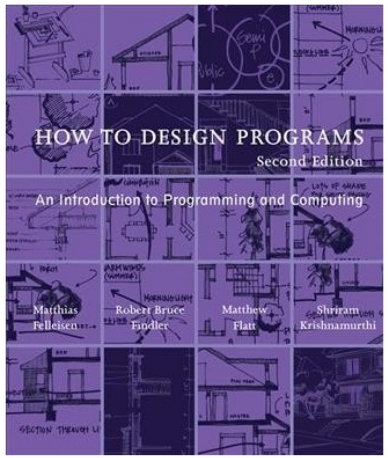
- This book (and course) will teach systematic program design.
- It presents recipes for designing two different types of programs.

The Better Way



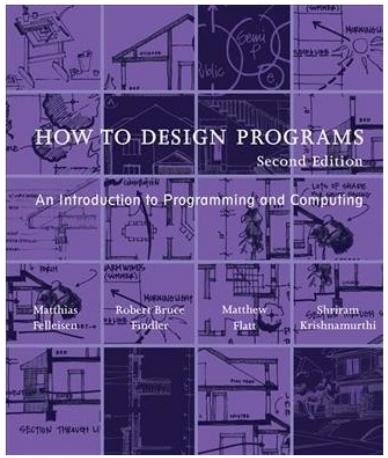
- This book (and course) will teach systematic program design.
- It presents recipes for designing two different types of programs.
  - Big bang (event driven) programs

The Better Way



- This book (and course) will teach systematic program design.
- It presents recipes for designing two different types of programs.
  - Big bang (event driven) programs
  - Batch programs (effectful scripts)

The Better Way



- This book (and course) will teach systematic program design.
- It presents recipes for designing two different types of programs.
  - Big bang (event driven) programs
  - Batch programs (effectful scripts)
- Isn't this a course on functional programming?

Learning Functional Programming from the Ground Up

- Why HDTP?

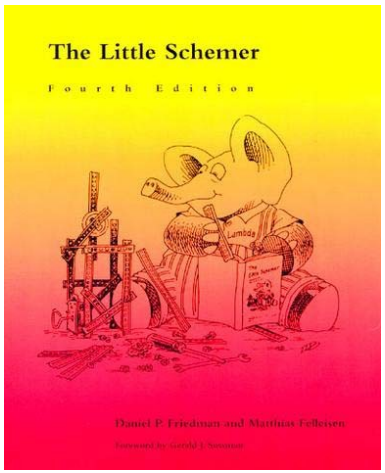
Learning Functional Programming from the Ground Up

- Why HDTP?
- When I first tried to learn functional programming, I failed.

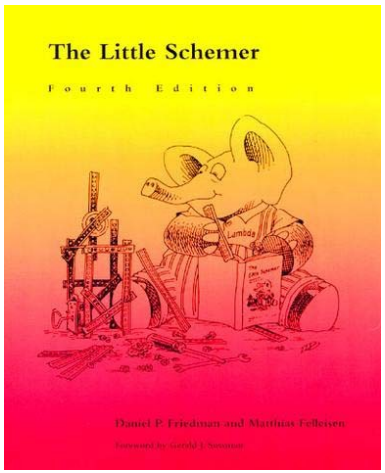
Learning Functional Programming from the Ground Up

- Why HDTP?
- When I first tried to learn functional programming, I failed.

Learning Functional Programming from the Ground Up

- Why HDTP?
- When I first tried to learn functional programming, I failed.
- 

Learning Functional Programming from the Ground Up

- Why HDTP?
- When I first tried to learn functional programming, I failed.
- 

- We will learn functional programming and systematic design

Why Racket and Not X?

- Teaching functional programming in an imperative-first language is awkward

Why Racket and Not X?

- Teaching functional programming in an imperative-first language is awkward
- Other functional languages have additional complexity

Why Racket and Not X?

- Teaching functional programming in an imperative-first language is awkward
- Other functional languages have additional complexity
- It's focused on being a good language for education

Course Structure

Grades will come from:

- Bi-weekly programming assignments

Course Structure

Grades will come from:

- Bi-weekly programming assignments
- 4-6 projects (1 or 2 as a team)

Course Structure

Grades will come from:

- Bi-weekly programming assignments
- 4-6 projects (1 or 2 as a team)
- An easy test and final

Course Structure

Grades will come from:

- Bi-weekly programming assignments
- 4-6 projects (1 or 2 as a team)
- An easy test and final
- *Try not to stress about grades*

The Main Thing I Want