CS 635 Advanced Object-Oriented Design & Programming
Fall Semester, 2020
Doc 1 Introduction
Aug 25, 2020

Reading

Aug 27 - Big Ball of Mud,
http://www.laputan.org/mud/mud.html
What Compsi textbooks don't tell you: Real world code sucks,
http://www.theregister.co.uk/2012/12/21/financial_software_disasters/print.html

Wait List

Last Day to Drop or Add Class

Last day to file for graduation

Sept 4

Course Web Site

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

CS 635 Fall 20

Course Details

Canvas
Zoom
Pandemic
Email
Exams

Lecture Notes
Assignments
Wiki
Course Portal
Syllabus
Reading Assignments

Pandemic

We are stuck with being remote

Personal interaction important part of education

In class & individual meetings

Feedback on how students are doing

Zoom

Use your real names

Keep video on

Audio muted

Practice
Raising Hand

Canvas

Was going to switch to doing everything in Canvas

Issues with assignments

Decided against after the Masters exams

Email Addresses

SDSU is now only sending email to your SDSU email address

Exams

Due to being remote seriously considered not having exams

But

Fewer graded events - bigger impact each assignment has on grade

As a student exams forced me to review - really helped in learning material

Oct 29 Midterm

Dec 17 Final

Languages

Java, C++, Swift, Kotlin, Python 3, Dart

C++ is STRONGLY Discouraged

I have not used C++ in over 10 years

I don't like the language

It is very difficult to grade
  Each additional language make grading harder

It is extremely hard to deal with GUI assignments in C++

Assignments are often harder in C++

What about C#?

I don't know the C# libraries

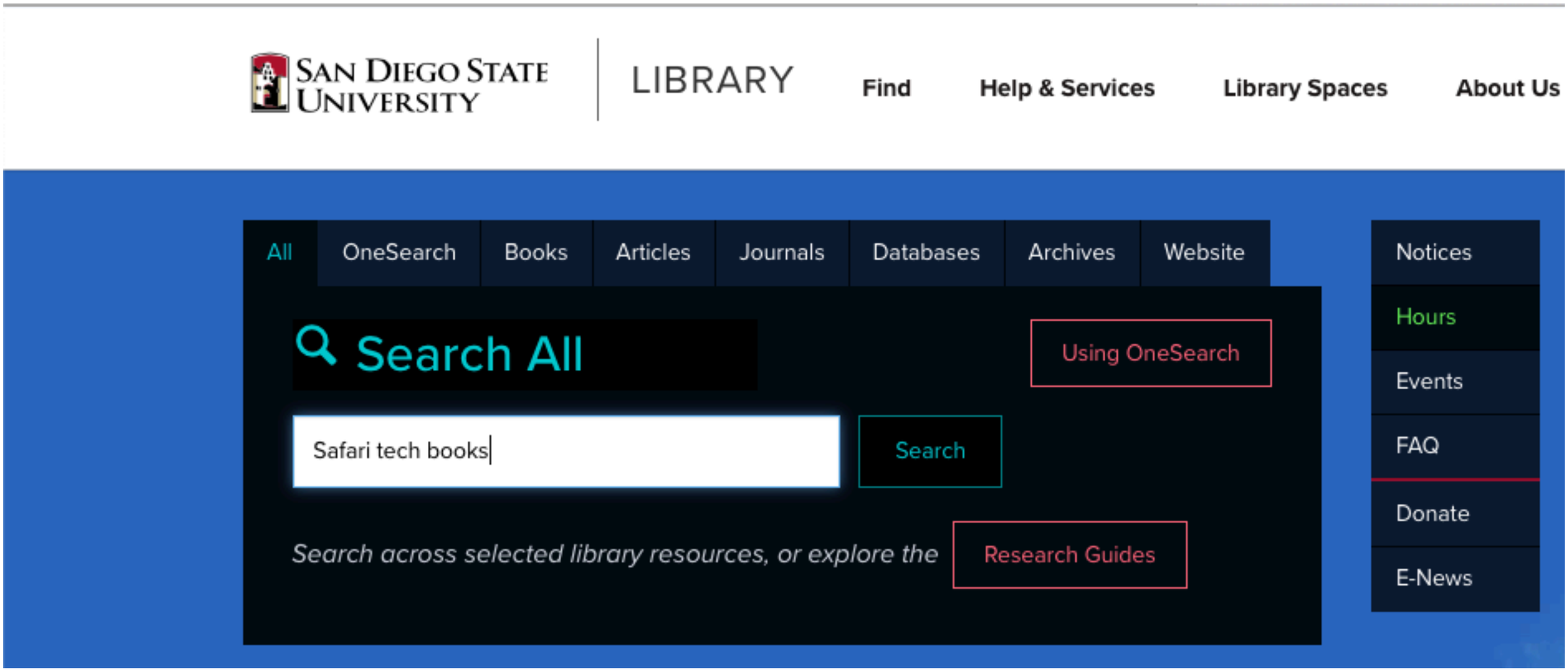
Which makes it too hard to grade assignments

Textbooks

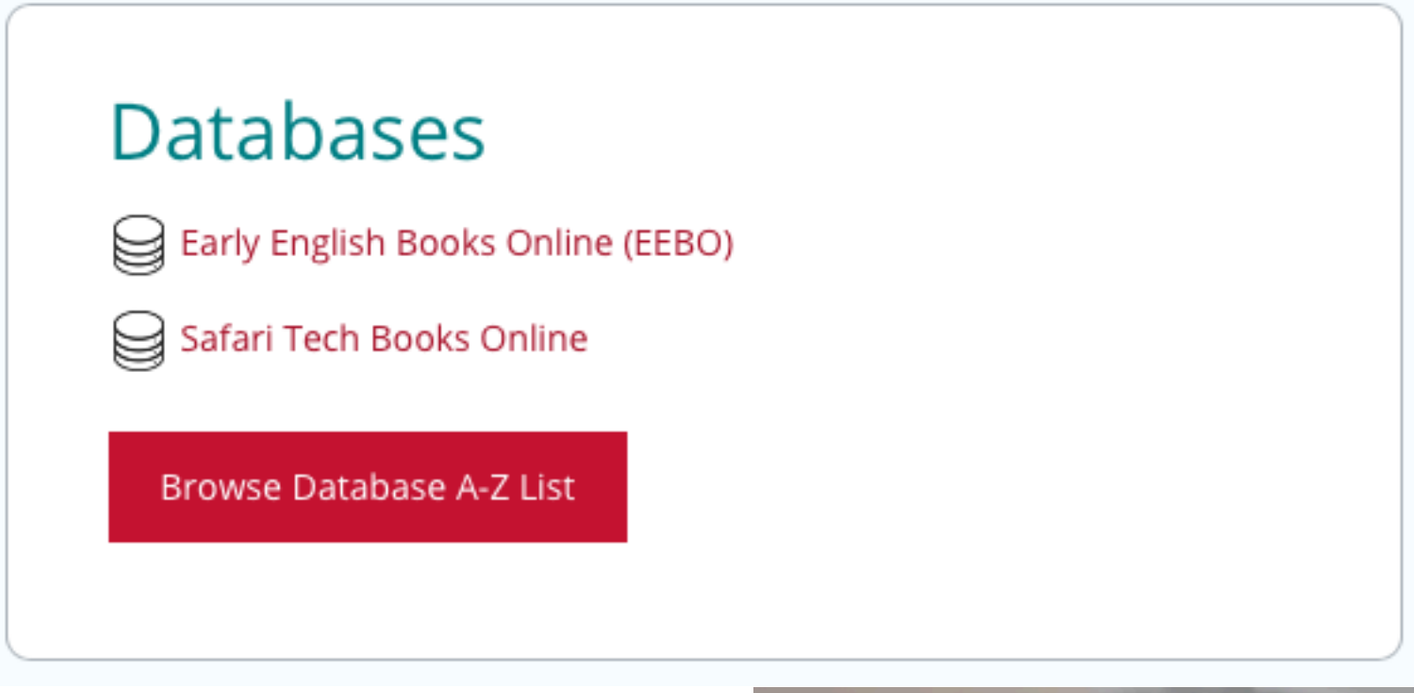
Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, Helm, Johnson, Vlissides, Addison-Wesley, 1995. Required

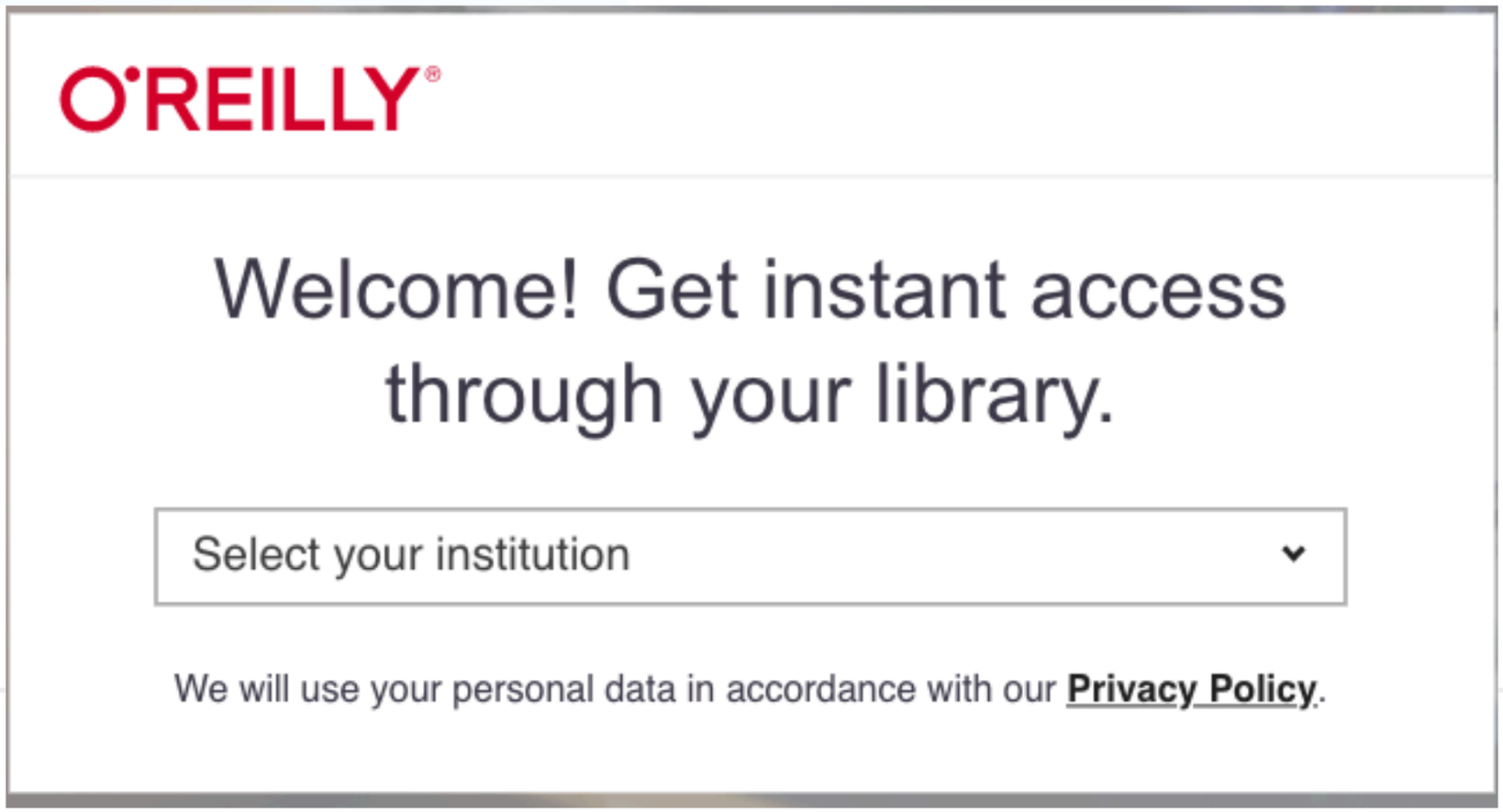
Refactoring: Improving the Design of Existing Code On-line access via SDSU Library, Required

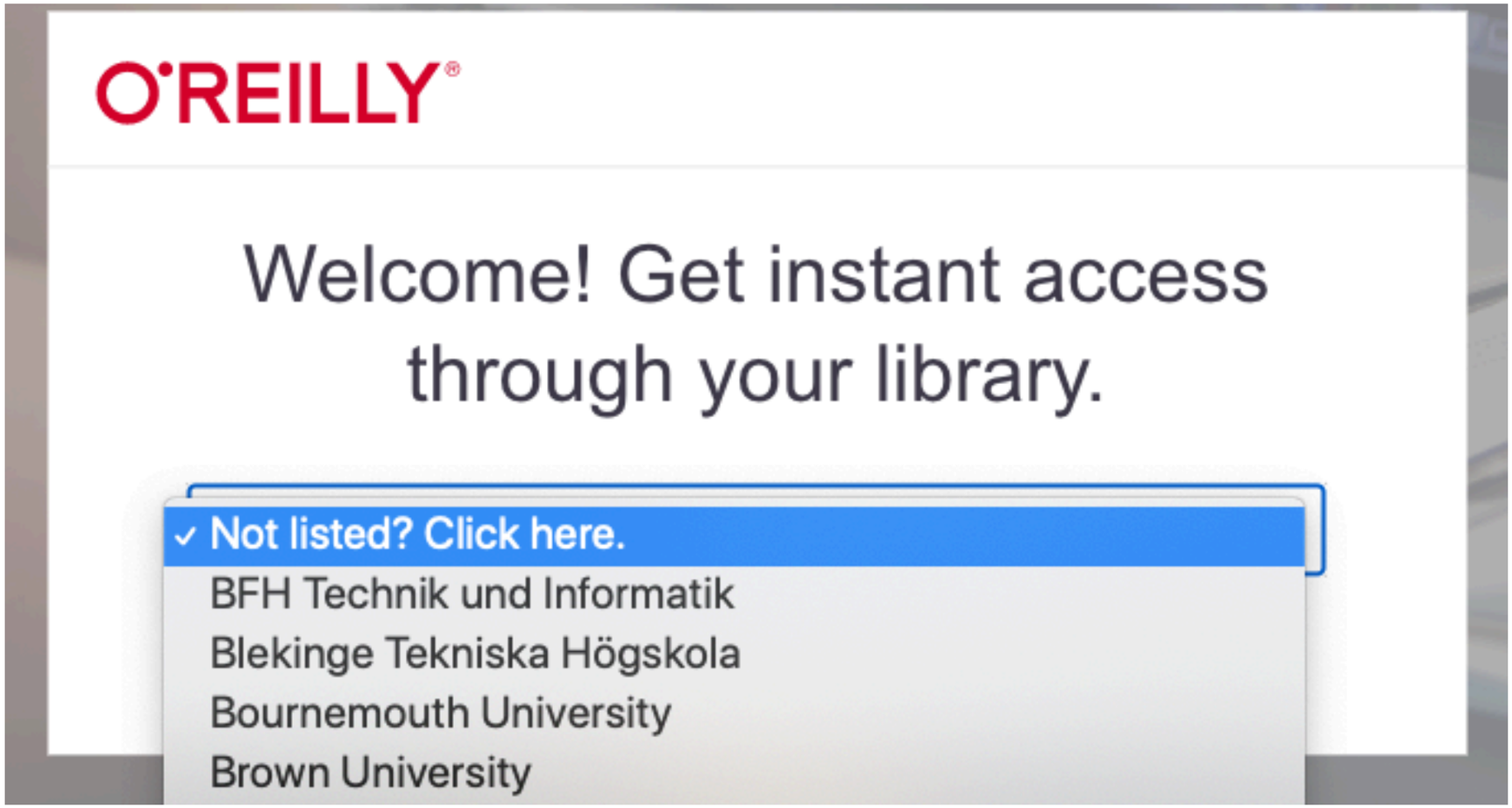
On-line Textbooks - https://library.sdsu.edu/



On the resulting page

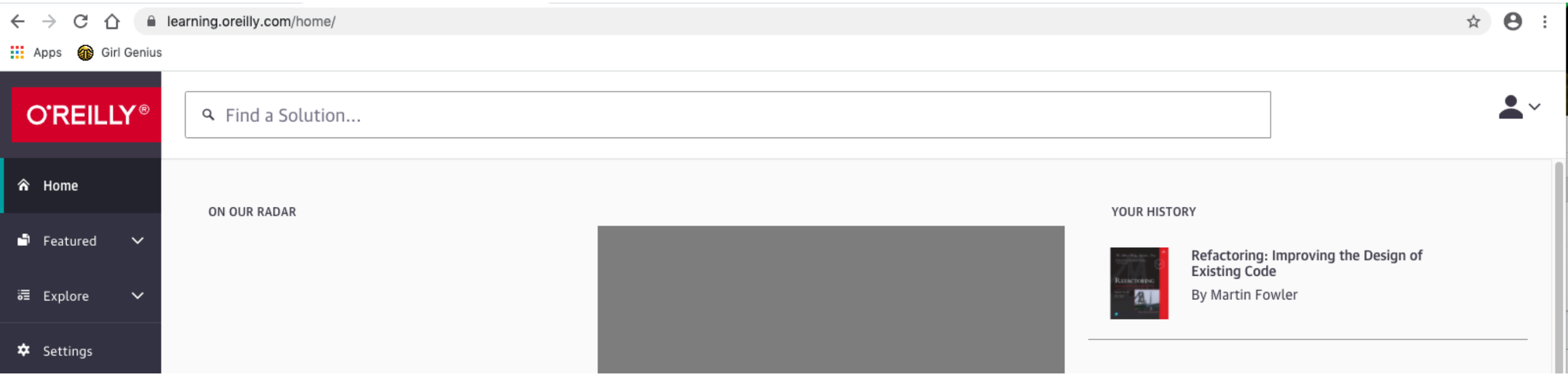






You then enter your SDSU email address

How you can search



In-Class verses On-line version

Same course but

In-Class verses On-line version

Same course but

In class provides visible social context

In-Class verses On-line version

Same course but

In class give access to other students

In-Class verses On-line version

Same course but

In class makes it easy to ask questions

In-Class verses On-line version

Same course but

on-line course requires more discipline and work

What this course is about

Writing quality OO code
Design Patterns
Coupling & Cohesion

Unit Testing
Refactoring

Scale Changes Everything

Norris Number

Average amount of code an untrained programmer can write before they hit a wall

~1,500 lines

Beyond that the code becomes so tangled they cannot debug or modify it without herculean effort

http://www.teamten.com/lawrence/writings/norris-numbers.html

Remedial Material

Formatting

Format your code

- Uniformly
- Consistently
- Show the block structure of your code

```
public void commandAction(Command c, Displayable d) {
   if (c == restartCmd) {
theGame.restart();
   } else if (c == levelCmd) {
   Item[] levelItem = {
 new Gauge(“level”, true, 9 theGame.getLevel())};
Form f = new Form(“Change Level”, levelItem);
        f.addCommand(OkCmd);
    f.addCommand(cancelCmd);
   f.setCommandListener(this);
     Display.getDisplay(this).setCurrent(f);
    } else if (c == exitCmd) {
  destroyApp(false);
  notifyDestroyed();
   }
 }
```

```
public void commandAction(Command c, Displayable d) {
if (c == restartCmd) {
theGame.restart();
} else if (c == levelCmd) {
Item[] levelItem = {
new Gauge(“level”, true, 9 theGame.getLevel())};
Form f = new Form(“Change Level”, levelItem);
f.addCommand(OkCmd);
f.addCommand(cancelCmd);
f.setCommandListener(this);
Display.getDisplay(this).setCurrent(f);
} else if (c == exitCmd) {
destroyApp(false);
notifyDestroyed();
}
}
```

Grading Policy - Formatting

Each method with such poor formatting loses 1 point per method

Name Structure - Language Conventions

	Java	Smalltalk	C#	Ruby
Class	PascalCase	PascalCase	PascalCase	PascalCase
Method	camelCase	camelCase	PascalCase	foo_bar
Field	camelCase	camelCase	camelCase	@foo_bar
Parameter	camelCase	camelCase	camelCase	foo_bar
Local Variable	camelCase	camelCase	camelCase	foo_bar

PascalCase ArrayList

camelCase courseSize

Grading Policy - Names

Each name that does not following your languages naming structure

- Loses 1 point/name
- Up to 20 points/assignment

Reading Verses Writing Programs

Code
- Written once
- Read many times

Use names that help the reader understand the code

Avd brvtns

brvtns r hrd t rd

n brvtn cn stnd fr dffrnt thngs

tmp - tmprr r tmprtr

Dffrnt ppl wll brvt dffrntl

Ds tcmplt s dn't hv t typ lng nms

Avoid Abbreviations

Abbreviations are hard to read

An abbreviation can stand for different things

tmp - temporary or temperature

Different people will abbreviate differently

IDEs autocomplete so don't have to type long names

Describe What "flag" is Used For

```
if (flag) {
    ...
}
```

```
if (foundDuplicate) {
    ...
}
```

flag
flagStatus
computeFlag

Do not help understand code

Variables 1 through N

String s1;
String s2;

String fileContents;
String pattern;

Who can remember the difference between s1 and s2?

Avoid Names With No Meaning

MyLinkedList

Who are you?

What makes your LinkedList different?

temp

All variables are temporary

```
swap = a;
a = b;
b = swap
```

```
(a, b) = (b, a)
```

Guidelines - Class Names

Use nouns

No abbreviations

Reader

Superclass

Single word to convey its purpose

StringReader

Subclass

Prepend adjective to superclass name

List

ArrayList

LinkedList

Guidelines - Method/Function/Procedure Names

Describe what method does

Use verb to describe an action

```
add(int index, E element)
clear()
```

If returns a value name what it returns

```
iterator()
subList(int fromIndex, int toIndex)
```

If returns boolean value make it true/false statement

```
isEmpty()
contains(Object o)
```

Guidelines - Variables, Fields, Parameters

Use names that indicate role variable is playing

If declare variable types don't use type as name

Use plurals to indicate collections

Make boolean variable names true/false statement

isVisible, hasMultipleParts,

```
public void execute(Vector vector) {
    Stack s;
}
```

```
public void execute(Vector commands) {
    Stack commandsExecuted;
}
```

Summary

Use names to help the reader understand the code

Follow language conventions

Avoid abbreviations

Use names that indicate role item is playing

Remedial Points

Each assignment I indicate points for different things

Remedial points (names, formatting) are in addition to those

Review

Define

Object

Class

What are the Benefits of OO

A verses B

```
struct A {
    int x;
    int y;
    int z;
}
```

```
public class B {
    public int x;
    public int y;
    public int z;
}
```

A verses B

```
public class A {
    public int x;
    public int y;
    public int z;
}
```

```
public class B {
    private int x;
    private int y;
    private int z;

    public int getX() { return x;}
    public int getY() { return y;}
    public int getZ() { return z;}
    public void setX(int value) {x = value;}
    public void setY(int value) {y = value;}
    public void setZ(int value) {z = value;}
}
```

A verses B

```
public class Stack {
   ArrayList elements;

   public void push(Object item) {
      elements.add(item)
   }

   public Object pop() {
   if elements.isEmpty()
      throw new NoSuchElement;
   return element.remove(elements.length -1)
}
```

```
struct Stack {
   ArrayList elements;
}


void push(Stack a, Object item) {
   add(a, item);
}

public Object pop(Stack a) {
   if isEmpty(a)
      throw new NoSuchElement;
   return remove(a, length(a) -1);
}
```

Heuristics

Keep related data and behavior in one place

A class should capture one and only one key abstraction

Heuristics

Beware of classes that have many accessor methods defined in their public interface

Do not create god classes/objects in your system

Beware of classes that have too much noncommunicating behavior

Learning, Understanding & Memorization

One disease long life

No disease short life