

Core Concepts

School of Computing
Clemson University
Clemson, SC 29634 USA

CPSC 2150: Wednesday, Jan. 18, 2023

Slide Sources:

- 1 Previous/Current CPSC 2150 Instructors
- 2 The lecture slides have also benefitted from instructors at Ohio State: Paolo Bucci, Wayne Heym, Paul Sivilotti, and Bruce Weide.

Introduction

A Simple Programming Exercise

Objective: Given an array, print out the result in reverse order.

- ▶ Do we have all the information we need?

Introduction

A Simple Programming Exercise

Objective: Given an array, print out the result in reverse order.

- ▶ Do we have all the information we need?
 - ▶ Array size?
 - ▶ Output formatting?
 - ▶ Where are we printing to?

Introduction

A Simple Programming Exercise

Objective: Given an array, print out the result in reverse order.

- ▶ Do we have all the information we need?
 - ▶ Array size?
 - ▶ Output formatting?
 - ▶ Where are we printing to?
- ▶ How many ways can you think of implementing this?

Intentional Design

- ▶ We need to be careful with our design
- ▶ So far, you've had simple assignments
 - ▶ Worked on your own
 - ▶ A few hours
 - ▶ Bad habits developed
- ▶ As a professional, your projects will be much more complex
 - ▶ Will work on a large team
 - ▶ Projects can span years, with multiple releases
 - ▶ You may have to build on code written 5 years ago by someone else!
- ▶ Major projects don't rewrite the system with new versions
 - ▶ They just change parts of the system

Bad Habits

- ▶ Since your programming assignments have been simple, you have gotten away with some bad habits
 - ▶ Lack of comments
 - ▶ Poor design
 - ▶ Code first, ask questions later
 - ▶ No testing
 - ▶ Assumptions made
 - ▶ Print to the screen (console) from anywhere
- ▶ As programs get larger and more complex, that doesn't work
 - ▶ Need to collaborate with others
 - ▶ Design is far more important
 - ▶ Less time spent actually writing code

Reacting to Change

- ▶ Hard to predict *what* changes will come
 - ▶ But we can predict that things will change with almost 100% certainty
- ▶ **In this class:**
 - ▶ Changes won't be made to the requirements once the assignment is published
 - ▶ **But**, the next assignment will build on and require changes to the previous assignment

Design for Change

- ▶ We have several practices and concepts that help us design our software and anticipate change
- ▶ This is what makes the difference between a *programmer* and a *software engineer*

Encapsulation

- ▶ Grouping related data and operations together
- ▶ We do this with **classes** and **objects**
 - ▶ State
 - ▶ Methods
- ▶ We can keep data in sync

Without Objects...

For Example: Suppose we want to keep track of student information...

- ▶ We need parallel arrays to keep track of:
 - ▶ Name
 - ▶ Email
 - ▶ Major
 - ▶ GPA
- ▶ Each array has part of the information for the same student
 - ▶ Index **X** from each array holds the information for that student
- ▶ Possibility for errors
 - ▶ Sorting the arrays
 - ▶ Accidental changes to wrong index
- ▶ A student object keeps this data together

Information Hiding

- ▶ **Note:** *Information Hiding* and *Encapsulation* are often thought to be two terms for the same thing
- ▶ They're really two distinct, but related concepts
- ▶ **Information Hiding** is how we control access to *encapsulated* data and operations
- ▶ We do this by using our visibility settings
 - ▶ `public`
 - ▶ `private`
 - ▶ Package `private`
 - ▶ `protected`

Access Levels

Modifier	Class	Package	Subclass	World
<code>public</code>	Y	Y	Y	Y
<code>protected</code>	Y	Y	Y	N
<code>no modifier</code>	Y	Y	N	N
<code>private</code>	Y	N	N	N

Why Hide Information?

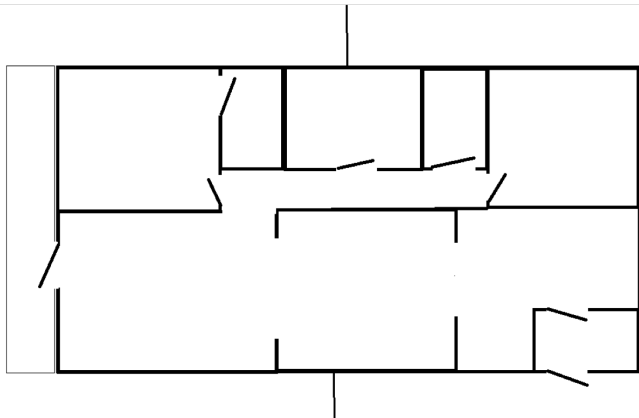
Why do we hide information?

- ▶ To keep data in sync
- ▶ To keep data accurate
- ▶ To keep data secure
- ▶ To not overwhelm other programmers with implementation details
 - ▶ How do computers solve for the square root of a number?
 - ▶ Do you really need to know?
or do you just need to know that `Math.sqrt()` returns the square root?

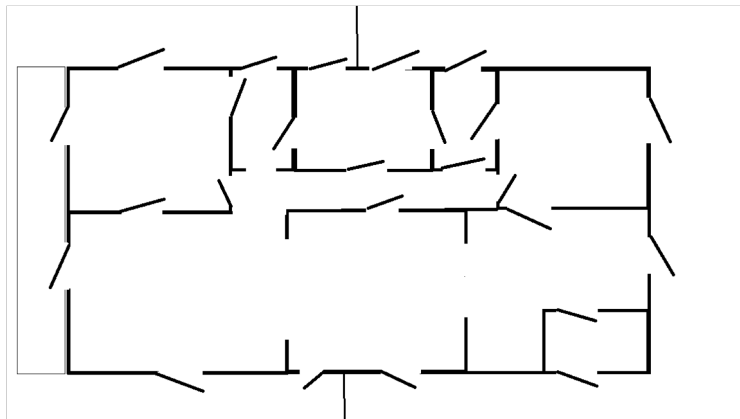
Consider building a house. . .

- ▶ Do we need to know how each of the components are built?
- ▶ Or do we just need to know how to use/install them?

Building a House Example



Building a House Example



One Simple Example

- ▶ Java has a `Date` class
 - ▶ However, the `Date` class example in the next slide has nothing to do with the Java `Date` class.
- ▶ Not necessarily a great example, but it's a convenient and easy example!

One Simple Example

Date class

- ▶ One possible Date class representation
 - ▶ `int day`
 - ▶ `int month`
 - ▶ `int year`
- ▶ Representation invariant for valid Date
 - ▶ day, month, and year should be in “sync”
- ▶ If day, month, and year can be modified arbitrarily, invalid dates could be represented
 - ▶ **For Example:** 13/35/2017

One Simple Example

Date class

- ▶ Through **public** methods provided by a class, access to the **private** representation can be controlled
- ▶ **Date** class methods either may not let you set the day, month, and year independently or have **preconditions**
- ▶ Unless there is related data to be “hidden” so that they can be kept in “sync”, grouped, and managed, classes may not be needed

Separation of Concerns

- ▶ AKA Modular design
- ▶ Every module has it's own specific role and concerns
- ▶ Modules then work together to solve a more complex problem
- ▶ Can have a high initial overhead
 - ▶ Have to identify the roles before writing code
 - ▶ Then design the modules
 - ▶ Then design how they work together
 - ▶ Then, start writing code
- ▶ **Objects** and **Classes** are modules

A good design helps with...

- ▶ Reuse

A good design helps with...

- ▶ Reuse
- ▶ Debugging

A good design helps with...

- ▶ Reuse
- ▶ Debugging
- ▶ Adapting to Change

Design Questions to Ask

What if _____ changes?

- ▶ Any of your numbers
 - ▶ Avoid “magic” numbers
 - ▶ use named constants instead
- ▶ User interface
 - ▶ Have all input and output be in one module
 - ▶ Boundary objects
- ▶ Order of events
 - ▶ What needs to be in what order, and what's flexible
 - ▶ Clear documentation
- ▶ Avoid repeated code
 - ▶ Separate into a module
 - ▶ Only have one place to make a change

Design Questions to Ask

Does <object> know _____ ? Should it?

- ▶ Hide information
- ▶ If an object **A** doesn't need to know info about object **B**, then **B** can change without affecting **A**.

Are these related?

- ▶ To encapsulate **OR** not to encapsulate
- ▶ Encapsulating related things makes it easier to keep related data in sync
- ▶ Encapsulating unrelated things makes it harder to reuse

Design Questions to Ask

Can someone use _____ 5 years from now without having to ask me?

- ▶ Do they need to know the programming language?
- ▶ Do I have clear documentation and comments?
- ▶ Did I hide implementation details, and use general abstract concepts

The Importance of Comments

What does this code do?

```
float foo( float number ) {  
    long i;  
    float x2, y;  
    const float threehalfs = 1.5F;  
  
    x2 = number * 0.5F;  
    y  = number;  
    i  = * ( long * ) &y;  
    i  = 0x5f3759df - ( i >> 1 );  
  
    y  = * ( float * ) &i;  
    y  = y * ( threehalfs - ( x2 * y * y ) );  
  
    // y  = y * ( threehalfs - ( x2 * y * y ) );  
  
    return y;  
}
```

With Comments and Name

What does this code do?

```
float Q_rsqrt( float number ) {  
    long i;  
    float x2, y;  
    const float threehalfs = 1.5F;  
  
    x2 = number * 0.5F;  
    y = number;  
    i = * ( long * ) &y;  
  
    // evil floating point bit level hacking  
    i = 0x5f3759df - ( i >> 1 );  
  
    // what is going on here?  
    y = * ( float * ) &i;  
    y = y * ( threehalfs - ( x2 * y * y ) );  
    // 1st iteration  
  
    // y = y * ( threehalfs - ( x2 * y * y ) );  
    // 2nd iteration, this can be removed  
  
    return y;  
}
```

What Does It Do?

Fast Inverse Square Root:

$$\frac{1}{\sqrt{x}}$$

- ▶ Needed to be faster for computer graphics
- ▶ https://en.wikipedia.org/wiki/Fast_inverse_square_root

Don't Worry If You Are Confused!

- ▶ This is what this course is about
- ▶ It takes time and practice to make these design decisions come naturally

Homework Assignment

- 1 Watch the various videos posted on your lab Canvas page.

Summary

- 1 Intentional (Bad) Design
- 2 Reacting, Designing and Adapting for Change
- 3 Encapsulation
- 4 Information Hiding
- 5 Separation of Concerns
- 6 Design Questions to Ask