
LATE ONE NIGHT...
OOP
DESIGN PATTERNS
2 WEEKS LATER...
WHAT IS THAT CODE DOING? WHO WROTE THIS CODE?
I HAVE NO IDEA.

Introduction to Good Code

CSCI 2134: Software Development

Agenda

- Lecture Contents
  - Brightspace Quiz
  - Software Quality from the Developer’s Perspective
  - Characteristics of Good Code
- Readings:
  - This Lecture: Chapter 20, 31
  - Next Lecture: Chapter 20, 31

Software Quality in 1st Year

- Functionality: Does it work? (Does it pass the tests)
- Solution Quality: Is the implementation / algorithm appropriate?
- Solution Clarity: Is the code properly formatted and commented?

- These criteria are reasonable for small programs

- Are they reasonable for large software systems?

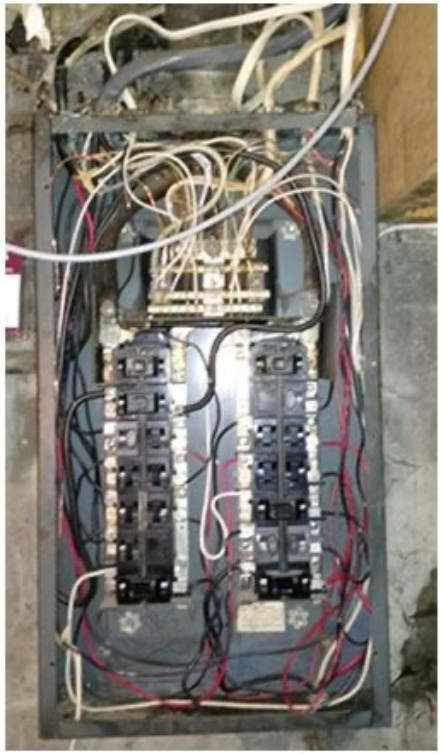
What Do We Mean by Software Quality?

User Perspective
(External, Functionality)

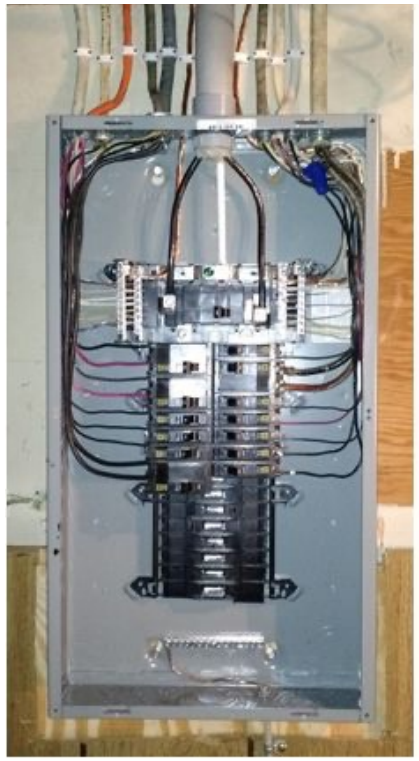
Bad

Good

Developer Perspective
(Internal, Implementation)



Bad



Good

User Perspective Software Quality Criteria

(McConnel, "Code Complete 2nd ed", 2004)

Criteria	Description
Correctness	The degree to which a system is free from faults in its specification, design, and implementation.
Usability	The ease with which users can learn and use a system.
Efficiency	Minimal use of system resources, including memory and execution time.
Reliability	The ability of a system to perform its required functions under stated conditions whenever required—having a long mean time between failures.
Integrity	The degree to which a system prevents unauthorized or improper access to its programs and its data.
Adaptability	The extent to which a system can be used, without modification, in applications or environments other than those for which it was specifically designed.
Accuracy	The degree to which a system, as built, is free from error, especially with respect to quantitative outputs. (How well the system performs its task.)
Robustness	The degree to which a system continues to function in the presence of invalid inputs or stressful environmental conditions.

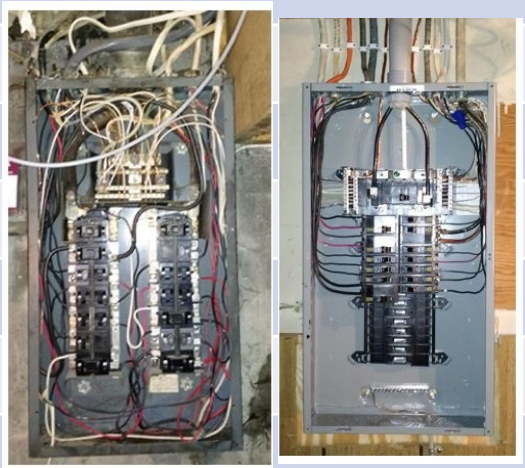
Developer Perspective Software Quality Criteria

(McConnel, "Code Complete 2nd ed", 2004)

Criteria	Description
Readability	The ease with which you can read and understand the source code of a system, especially at the detailed-statement level.
Testability	The degree to which you can unit-test and system-test a system; the degree to which you can verify that the system meets its requirements.
Understandability	The ease with which you can comprehend a system at both the system-organizational and detailed-statement levels. (How easy is it to understand the entire system.)
Maintainability	The ease with which you can modify a software system to change or add capabilities, improve performance, or correct defects.
Reusability	The extent to which and the ease with which you can use parts of a system in other systems.
Flexibility	The extent to which you can modify a system for uses or environments other than those for which it was specifically designed.
Portability	The ease with which you can modify a system to operate in an environment different from that for which it was specifically designed.

Criteria for Software Quality and Our Focus

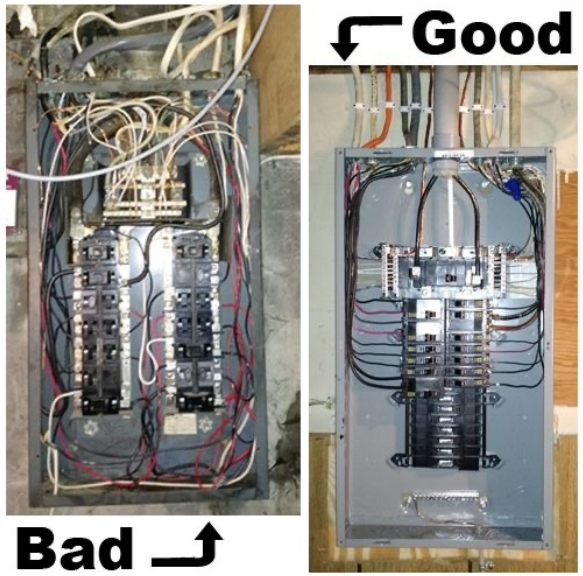
User Perspective (External, Functionality)	Developer Perspective (Internal, Implementation)
Correctness	Readability
Usability	Testability
Efficiency	Understandability
Reliability	Maintainability
Integrity	Reusability
Adaptability	Flexibility
Accuracy	Portability
Robustness	



You may not be able to achieve all the characteristics.
Some characteristics may oppose one another.

Why Does Software Quality Matter

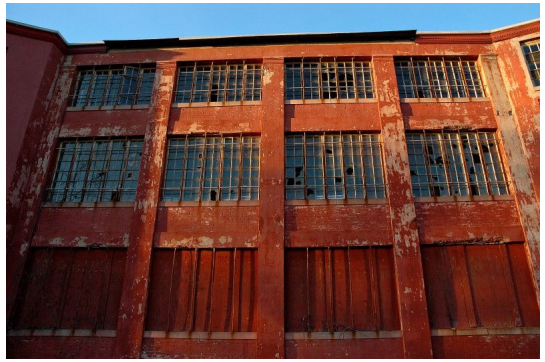
- User Perspective: It’s harder to sell bad software.



- Developer Perspective: Software needs to be
  - Developed
  - Maintained
  - Fixed
  - Updated
  - Extended
  - Etc.
- It is cheaper to maintain and produce good quality software
- This is known as the **General Principle of Software Quality**

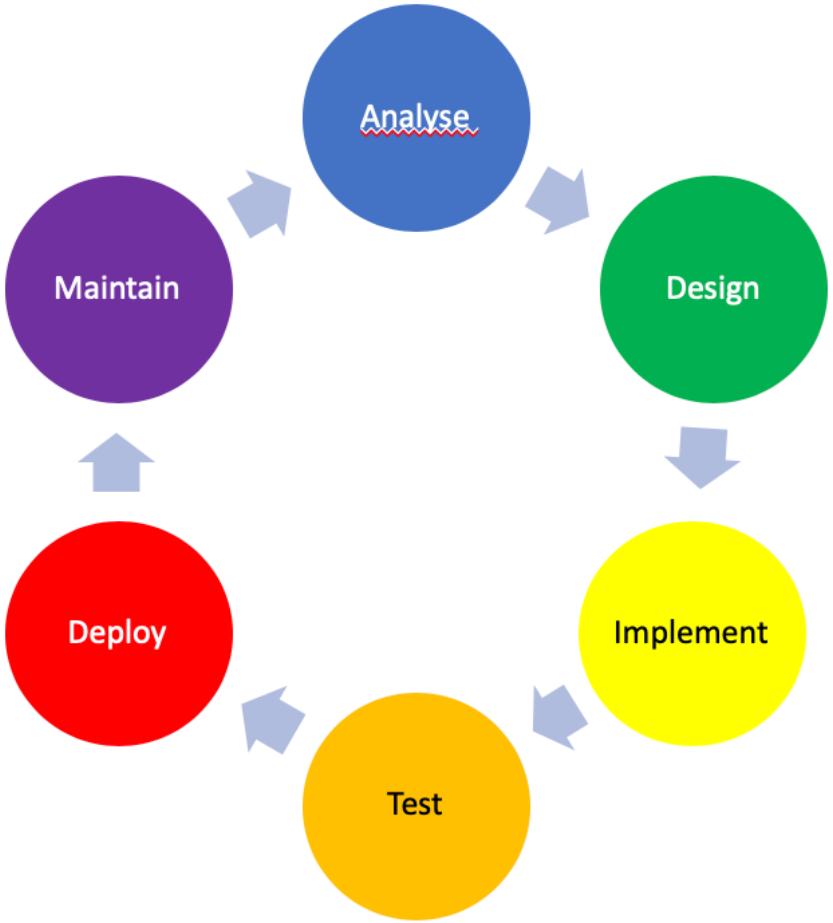
Software is a Liability

- Question: What is a good size for a software project?
- Answer: As small as possible
- Liabilities
  - **Software rot**: APIs change, systems change, standards change
  - **Software obsolescence**: Users want new features
  - **Software defects**: There will always be bugs to fix
- All software needs to be maintained:
  - More software means more maintenance means higher cost
  - Better quality software reduces the maintenance cost



Starting with Readability

Criteria	Description	Stage
Readability	Easy to understand each code unit	
Testability	Easy to verify that code is correct	
Understandability	Easy to understand entire system	
Maintainability	Easy to make small changes	
Reusability	Easy to use code in other projects	
Flexibility	Easy to make large changes	
Portability	Easy to adapt to a new platform	



- Suppose you are a new programmer, working on your first project
- What is the worst thing you can do? What is the least you should do?
  - Commit broken code! Produce readable code.

Characteristics of Readable Code

- Readable code is easy to comprehend
- Readable code is
  - Divided into small logical units that are easy to understand logically fit together
  - Well documented and explained
  - Consistently formatted using a common standard
- Which of these is more readable? A or B?

A

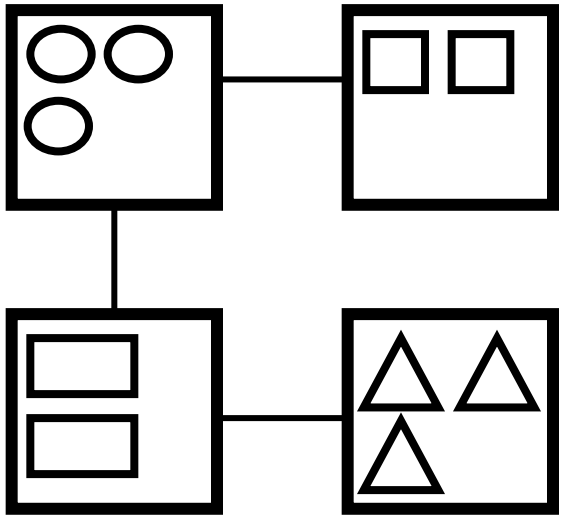
```
public class
LongestIncreasingSubsequence { static
void main(String [] a) {
Scanner in = new Scanner(System.in);
int index = 0; int curIndex = 0;int
maxCount = 1; int count = 1; int prev =
in.nextInt(); int seq = 1;
while(in.hasNextInt()) { int next =
in.nextInt(); if(next > prev) { count++;
if(count > maxCount) { maxCount = count;
index = curIndex; } } else { curIndex =
seq; count = 1; } prev = next; seq++; }
System.out.println(index); } }
```

B

```
/**
 * CSCI 1110
 * @author Alex Brodsky
 * @description: This program computes the start
 * of the longest increasing subsequence of a
 * sequence of numbers
 */

public class LongestIncreasingSubsequence {
  /* main method, where the program starts running
   * @params: Strings [] : command line parameters
   * @return: none
   */
  static void main(String [] a) {
    Scanner in = new Scanner(System.in);

    int index = 0;          // start of longest seq.
    int curIndex = 0;       // start of current seq.
    int maxCount = 1;       // length of longest seq
    int count = 1;          // length of current seq
    int prev = in.nextInt();  // prev num in seq.
    int seq = 1;            // seq # of cur read num

    /* assume integers are separated by spaces
     */
    while (in.hasNextInt()) {
      int next = in.nextInt();

      /* Next number is either part of current sequence
       * or start of next
       */
      if (next > prev) {
       . . .
```

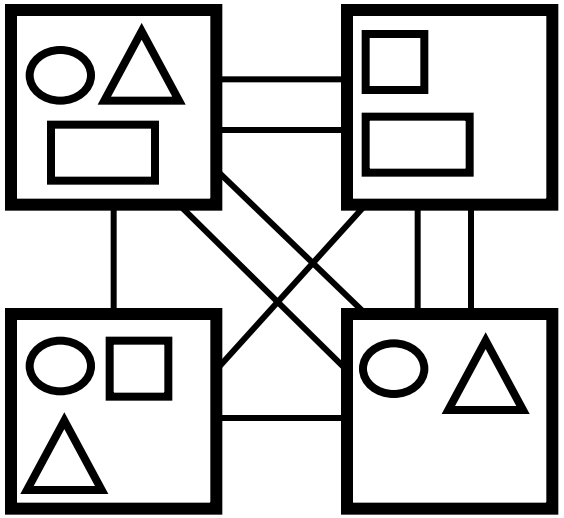
Characteristics of Testable Code

- Testable code can be easily be tested and checked for defects
- This means
  - Every unit can be tested and verified individually
  - Each component can be tested to ensure all units are working together
  - All the components of a system are working together
- We will spend the next couple weeks talking about testing

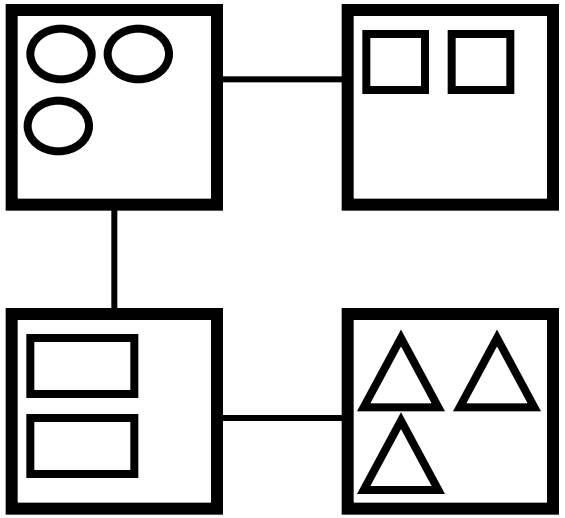


Characteristics of Understandable Code

- Highly understandable code is
  - Readable
  - Logically structured at the high level
  - Has low coupling
  - Has high cohesion
- Low coupling and high cohesion depend on good design
- We will talk about good design in the last month of the course



Low-cohesion, high coupling
Less understandable

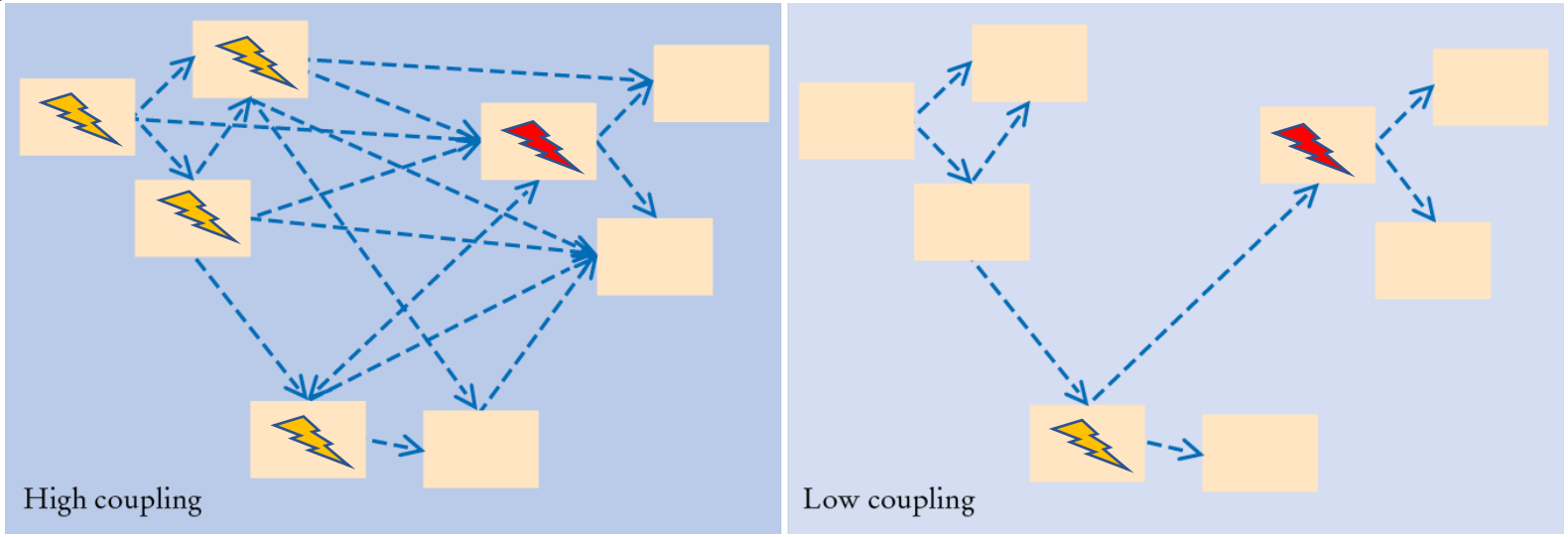


High cohesion, low coupling
More understandable

Coupling

© visual7/iStockphoto.

- Coupling is the density of dependencies among classes
  - High coupling means there are many dependencies
  - Low coupling means there are few dependencies
- Low coupling is preferred
- If a class changes and there is high coupling, many other classes will need to change as well



Cohesion

- Ideas:
  - Each class should represent a single concept
  - All methods of the class should be directly applicable to the concept
- Classes that are multi-concept or have unrelated methods reduce cohesion and indicate improvements are needed to the design.

Example:
Bad vs Good Cohesion

```
public class Car {
  int engineSize;
  int gasTankSize;
  boolean diesel;
  Boolean autoTransmission;
  …
}
```

```
public class Car {
  Engine engine;
  Transmission transmission;
  FuelTank fuelTank;
  …
}
```

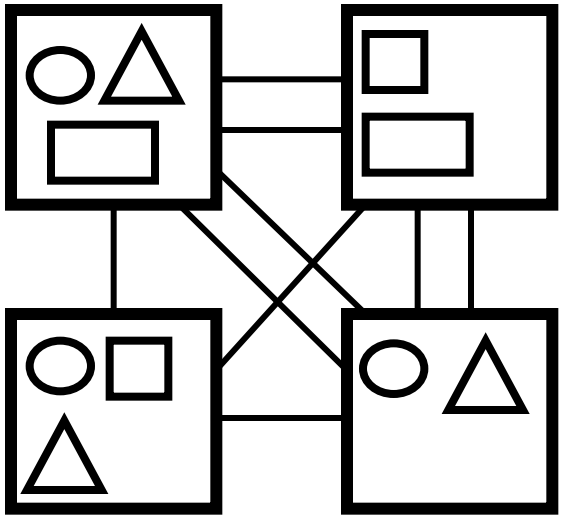
```
public class Engine {
  int size;
  boolean diesel;
  …
}
```

```
public class Transmission {
  Boolean automatic;
  …
}
```

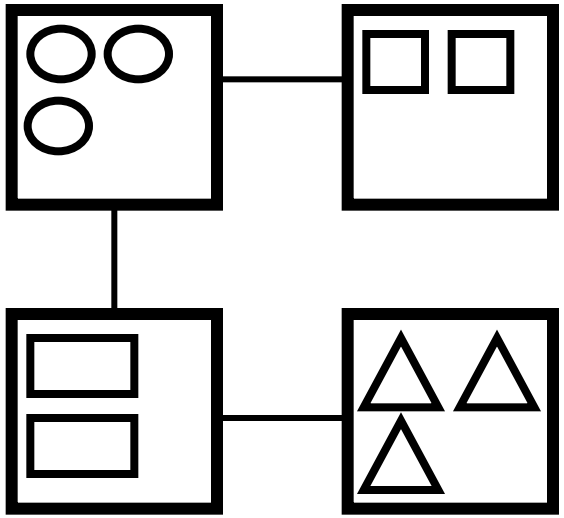
```
public class FuelTank {
  int size
  …
}
```

Characteristics of Understandable Code

- Highly understandable code is
  - Readable
  - Logically structured at the high level
  - Has low coupling
  - Has high cohesion
- Low coupling and high cohesion depend on good design
- We will talk about good design in the last month of the course



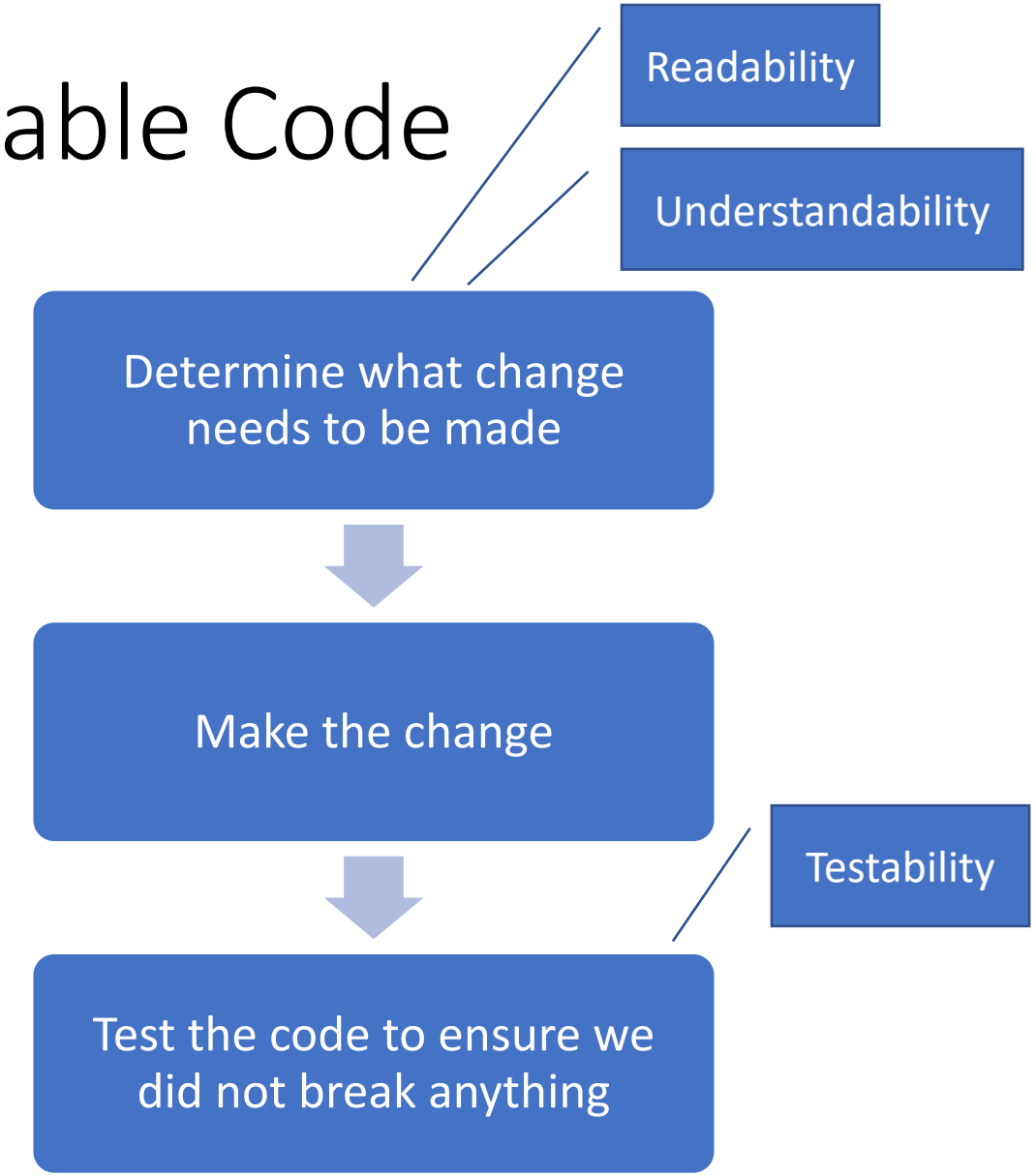
Low-cohesion, high coupling
Less understandable



High cohesion, low coupling
More understandable

Characteristics of Maintainable Code

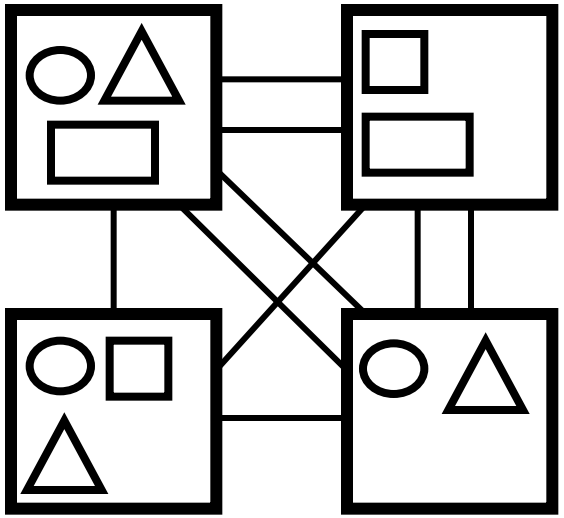
- It is easy to make small changes and fix bugs in maintainable code
- Highly maintainable code is
  - Readable
  - Testable
  - Logically structured at the high level
  - Understandable



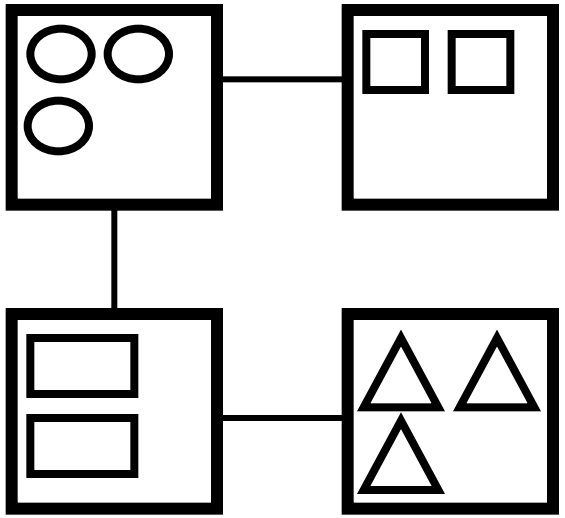
Characteristics of Maintainable Code

- It is easy to make small changes and fix bugs in maintainable code
- Highly maintainable code is
  - Readable
  - Testable
  - Logically structured at the high level
  - Understandable
- **Observation**

  If a small change to software requires multiple modifications, maintainability is not ideal



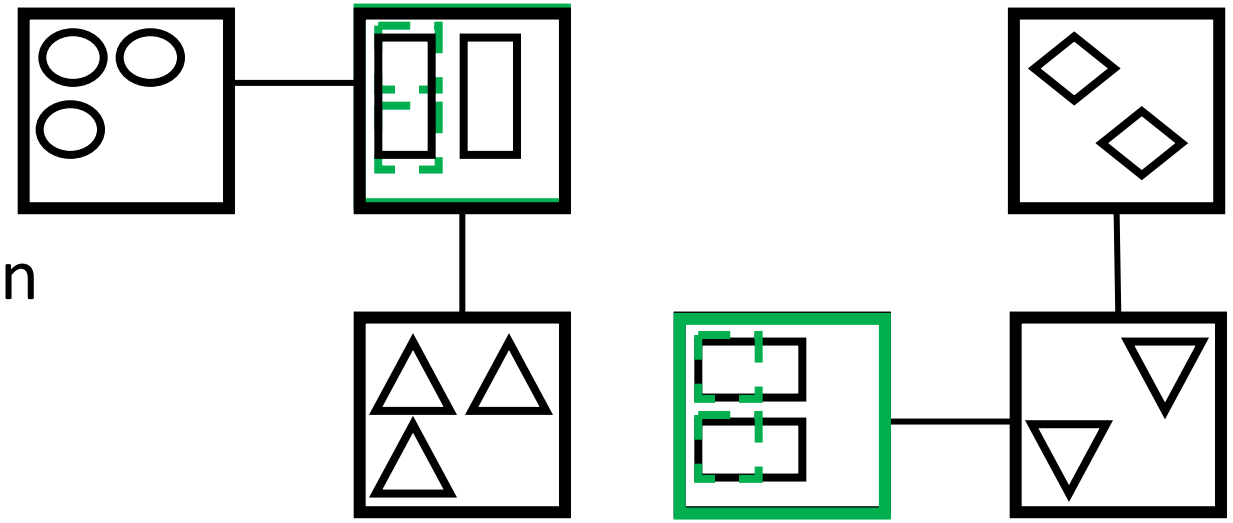
Less Maintainable



More Maintainable

Characteristics of Reusable Code

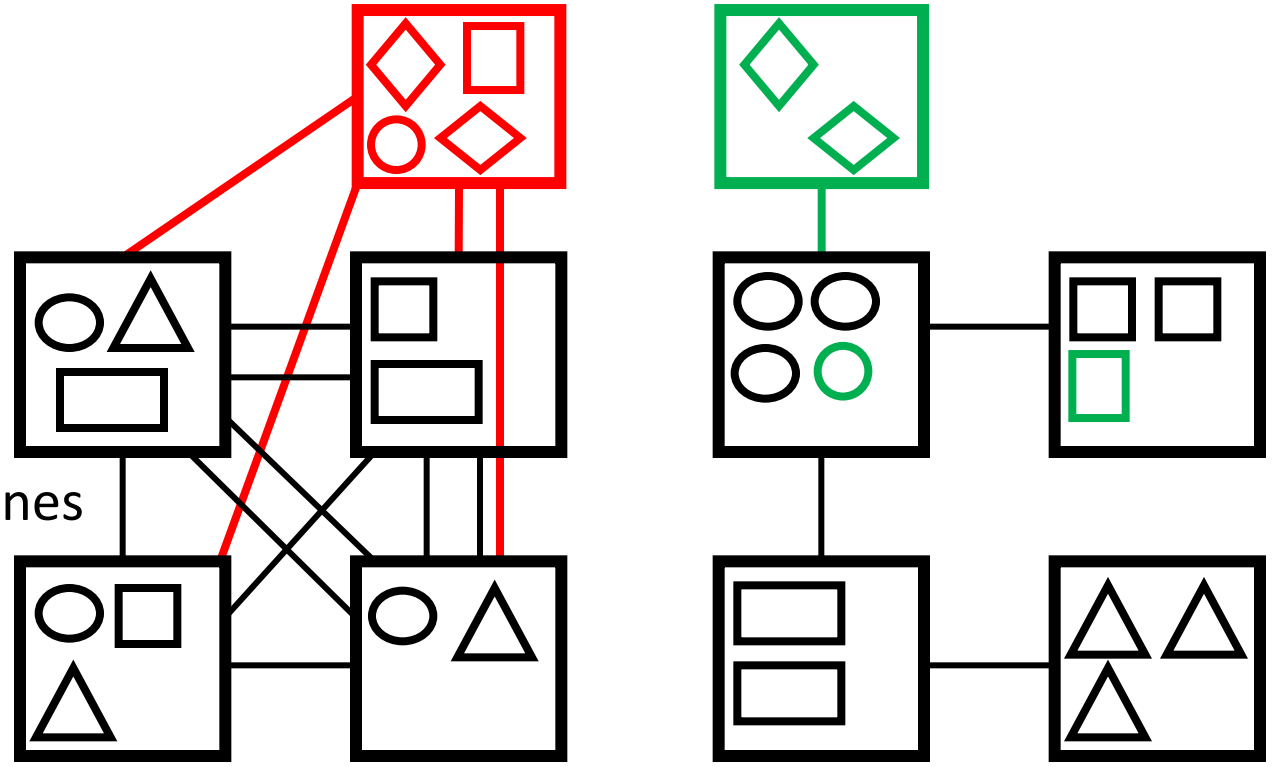
- Reusable code can be reused in other projects, making it much more economical (write-once, use-many)
- Reusable code:
  - Has well defined functionality (an interface)
  - Is as general as possible
  - Is specified by **what** it does, not **how** it does it
- Reusability depends on good design



Characteristics of Flexible Code

- Flexible code allows easy adaptation of code for
  - New purposes
  - New environments
- Flexible code must be maintainable, readable, understandable, and extensible
- Extensibility:
  - Easy to create new software components to interact with existing ones
  - Easy to rearrange how the existing components interact with each other
- Flexibility depends on
  - Good design (Design stage)
  - Good analysis (Analysis stage)

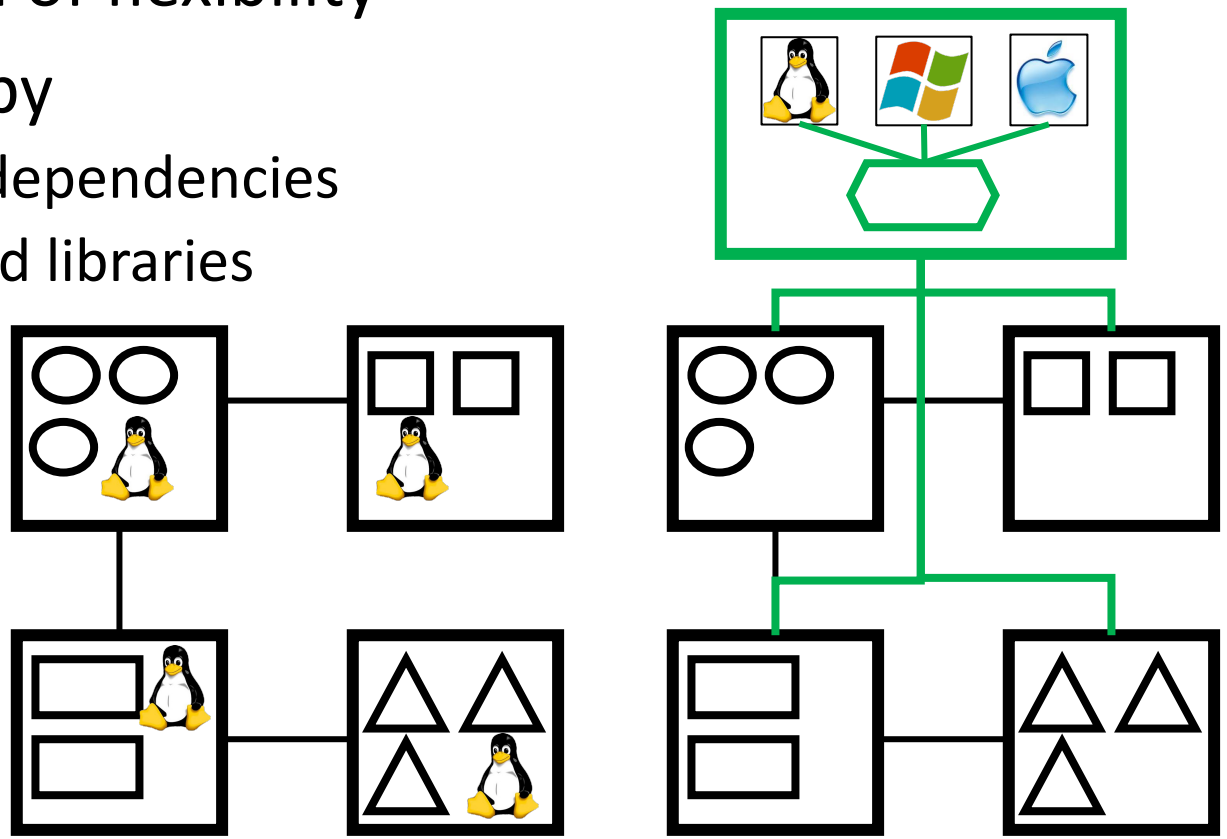
    A recognition that software needs evolve over time



Less Extensible **More Extensible**

Characteristics of Portable Code

- Portable code can be easily adapted to run on different platforms
- Portability is a special form of flexibility
- Portable code is achieved by
  - Encapsulating all platform dependencies
  - Avoiding use of nonstandard libraries



Avoid Use of Nonstandard Libraries

- Two common mistakes that developers make:
  - Recreating functionality found in the language's standard library
  - Using nonstandard libraries without considering the implications
- Problems
  - Some implementations are platform specific
  - Developer's version may be buggy
  - Nonstandard libraries are not available for all platforms
- Guidelines:
  - If the standard library provides the functionality, use it
  - Avoid using nonstandard libraries or writing your own version if you do not need to

Key 

Points

- Developer’s criteria for software quality focuses on the quality of the code itself rather than functionality
- Software quality criteria include: readability, testability, understandability, maintainability, reusability, flexibility, and portability
- As new developers, our biggest impact is in software readability and testability

Image References

Retrieved December 19, 2019

- http://pengetouristboard.co.uk/vote-best-takeaway-se20/
- https://media.istockphoto.com/vectors/sledgehammer-to-crack-a-nut-vector-id465746244?k=6&m=465746244&s=612x612&w=0&h=jMa4sJQsKl6DMgm6NCSr-WPog5j82pDY3I3QbxmETw4=
- https://previews.123rf.com/images/artnataliia/artnataliia1801/artnataliia180100009/93384948-big-set-of-house-repair-tools-including-hammer-sledgehammer-spatula-brush-nail-screw-nut-wrench-and-.jpg
- https://cdn0.iconfinder.com/data/icons/file-names-vol-8-1/512/03-2-512.png
- https://c7.uihere.com/files/893/429/157/bmp-file-format-bitmap-others-thumb.jpg
- https://upload.wikimedia.org/wikipedia/commons/thumb/a/af/Revision_controlled_project_visualization-2010-24-02.svg/800px-Revision_controlled_project_visualization-2010-24-02.svg.png
- https://images-na.ssl-images-amazon.com/images/I/61q0wrsXejL._AC_SY741_.jpg
- https://publicdomainvectors.org/en/free-clipart/Do-Not-sign-vector-clip-art/30078.html

Image References

Retrieved December 29 - 31, 2019

- http://pengetouristboard.co.uk/vote-best-takeaway-se20/
- https://www.twotwentyone.net/wp-content/uploads/2013/08/USB-receptacle.jpg
- https://i.pinimg.com/originals/b5/22/38/b52238fad11b0a3ecac36fa176041d98.jpg
- https://webstockreview.net/images/clipart-hospital-money-3.png
- https://s3-production.bobvila.com/articles/wp-content/uploads/2018/04/Reasons_Electrical_Outlet_Not_Working.jpg
- https://c7.uihere.com/files/109/173/249/software-quality-assurance-quality-control-quality-management-assurance.jpg
- https://i7.pngguru.com/preview/380/91/790/hourglass-time-clock-clip-art-vector-illustration-time.jpg
- https://1001freedownloads.s3.amazonaws.com/vector/thumb/133270/neoguiri_Barrier.png
- https://thumbs.dreamstime.com/z/six-components-project-charter-components-project-charter-159700743.jpg
- https://thumbs.dreamstime.com/b/compliance-rules-regulations-guidelines-arrow-signs-words-colorful-road-directing-you-to-comply-wih-important-laws-31478130.jpg
- https://www.researchgate.net/profile/Mehmet_Celepkolu/publication/329855173/figure/fig2/AS:706489106841600@1545451537633/Pair-programming-setting-Students-look-in-different-directions-during-the-session.png
- https://www.slideshare.net/ChihyangLi/object-oriented-programming-ch3-srp-dip-isp
- https://www.clipartmax.com/png/middle/146-1467994_windows-symbol-mark-start-menu-icon-windows-8.png
- https://library.kissclipart.com/20181207/wyw/kissclipart-linux-logo-png-clipart-linux-foundation-tux-e477406d44b4921f.jpg

Retrieved September 16, 2020

- https://codemestat.files.wordpress.com/2014/07/art_of_readable_code_cartoon.png