CSCI-2600: Principles of Software

Spring 2024

Konstantin Kuzmin psoftstaff@cs.lists.rpi.edu or kmkuzmin@gmail.com

https://www.cs.rpi.edu/academics/courses/spring24/csci2600/

Thanks

- Some of the material in this course comes from
  - Prof. Milanova
  - Prof. Goldschmidt
  - Prof. Thompson
  - Prof. Varela
  - Prof. Michael Ernst - University of Washington
  - Many others
- Thanks to our TAs and mentors
- Thanks to Prof. Cutler and the Submitty team

Some Administrative Notes

- On SIS, CSCI 2600 is restricted to CSCI and ITWS majors.
  - I will sign add forms for ITWS majors who cannot add this class on SIS
- If a student is adding the dual major or switching to CSCI, they must do that first before registering for CSCI 2600.
  - Students can go to the SoS Advising Hub (Empire State Hall 1107 at Polymer Center) to declare the dual major

More Administrative Notes

- All communications regarding accommodations, absences, extensions, and other scheduling and logistic matters must be directed to Shianne Hulbert (hulbes@rpi.edu) with a CC to Dr. Kuzmin (kuzmik2@rpi.edu)
- Any questions on homework, technical issues, quizzes, etc. should be posted on Submitty Forum. If you do not want your question to be on the forum, you may send an email to psoftstaff@cs.lists.rpi.edu mailing list
- Emailing the instructor or a specific TA instead of using the Forum or the mailing list will cause delays in getting a response

- The class web site is https://www.cs.rpi.edu/academics/courses/spring24/csci2600/
- Please carefully read the syllabus and ask any questions you have on the Submitty forum
- If you are not able to log onto Submitty, please let us know immediately.
- All homework assignments unless otherwise stated must be completed individually!
  - **Cheating will not be tolerated**
- Discussion is encouraged on Submitty, but do not post code (whether correct or incorrect) from any gradeable assignment.
  - https://submitty.cs.rpi.edu/courses/s24/csci2600/forum

How to ask for help

Before asking for help

- Read the docs.
- Fix all compiler errors.
- Examine all compiler warnings. Fix them if possible.
- Read and try to understand the error messages.
- Try and fix the problem yourself.
- Test your code with relevant data.
- Read the forum/mailing list, the question may have been answered.

When reporting a problem, say what you were doing,
what you expected to happen and what actually happened.

- Give information.
- “My code doesn’t work. What’s wrong?” is useless to you and your audience.
  - If your code worked, you wouldn’t be asking.

Before posting a question, read it yourself and ask if you would understand it.
If not, add more info. Remember other people are not there with you so they can't see what's on your screen.

Be polite. Say "thank you". Expressing frustration may make you feel better, but it doesn’t
help solve the problem and may just annoy people who might help you.

Remember: It’s almost never the hardware or the server. It’s your code.

Java and Git Environments

- As part of Homework 0, you will install Eclipse (general-purpose IDE)
  - https://www.cs.rpi.edu/academics/courses/spring24/csci2600/Documents/eclipse_and_git.pdf
- Homework will be released and also turned in using Git on Submitty
- Git – Version Control System (VCS)
- To turn in your homework, commit into your repository, push the files to Submitty then push the files to the Homework Submission Server
- Commit early and often
- More details on Git, JUnit, etc. to follow

Homework Assignments

- Homework assignments must be submitted (via Submitty) by 11:59 PM EST/EDT on the given due date
- The Submitty URL is https://submitty.cs.rpi.edu/home
- You have seven late days (7) for you to use for the entire semester for any reasons, including sickness and other absences
- You have a maximum of five (5) late days per assignment
- Requests for extensions beyond the limits specified above will only be considered in cases of prolonged excused absence backed by a request from the Student Success Office or other school authority

Remember This!

New submission for: Homework 0 **Due: 09/12/2019 @ 11:59 PM**

Select submission mode:
◉ Normal Submission ○ Make Submission for a Student

To access your Repository:
Note: There may be a delay before your repository is prepared, please refer to assignment instructions.

```
git clone https://submitty.cs.rpi.edu/git/f19/csci2600/hw00/thompw4 SPECIFY_TARGET_DIRECTORY
```

Grade My Repository

You must click this button for your homework to be graded.

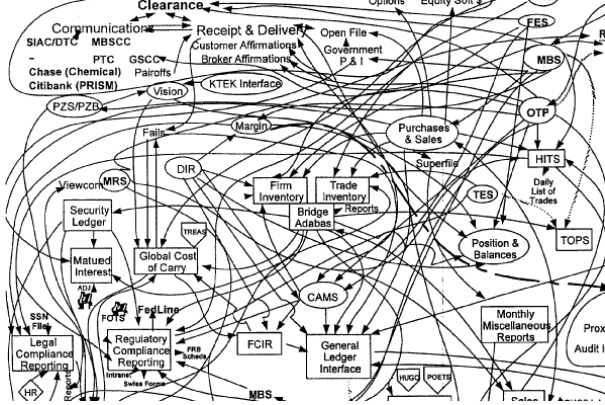
Just committing and pushing your repo is not enough.

What This Course is About

- Writing **correct code**
  - Error free code
  - Does what the programmer intended
- How can we describe what we intend?
- How can we guarantee the code matches our descriptions?
- We need more
  - The code must be maintainable

- Object-Oriented Programming

Why is software so hard?



- Complexity
  - Programming is about managing complexity
  - Complexity is anything related to the structure of a software system that makes it hard to understand and modify the system
    - John Ousterhout A Philosophy of Software Design
  - Common indications are *antipatterns* and *code smells*

- Symptoms
  - Change Amplification
    - Simple changes require changes in many different places
    - One of the goals of good design is to reduce the amount of code that is affected by each design decision, so design changes don’t require very many code modifications
  - Cognitive Load
    - How much does a developer need to know to complete a task
  - Unknown unknowns
    - Occurs when it is not obvious which pieces of code must be modified to complete a task, or what information a developer must have to carry out the task successfully

Causes of Complexity

- Dependencies
  - A dependency exists when a given piece of code cannot be understood and modified in isolation
  - There will always be dependencies, but they should be made obvious
- Obscurity
  - Obscurity occurs when important information is not obvious
    - Often a documentation problem
- Complexity is incremental
  - It creeps in

Goal – Manage Complexity

- Complexity comes from an accumulation of dependencies and obscurities.
- As complexity increases, it leads to change amplification, a high cognitive load, and unknown unknowns.
  - As a result, it takes more code modifications to implement each new feature.
  - Developers spend more time acquiring enough information to make the change safely.
  - In the worst case, they can’t even find all the information they need.
- The bottom line is that complexity makes it difficult and risky to modify an existing code base.

How Do We Manage Complexity?

- Modular design
  - Software is decomposed into a collection of *modules* that are relatively independent
    - Classes, subsystems, services
  - In an ideal world, modules would be independent of one another
  - In the real world, there are always dependencies
  - Our goal is to manage these dependencies to reduce complexities
    - Separate interface from implementation
      - The user of a modules interacts with the module through an interface
      - How the module performs the operation is hidden from the user
    - Write correct specifications
      - Programmer codes to the spec, not the code
    - Follow correct practices

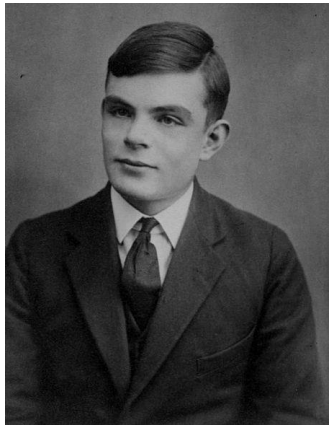
Correctness - why do we care?

- Programming errors cost the U.S. economy $60 billion yearly in lost efficiency and real harms.
  - http://outfresh.com/knowledge-base/6-famous-software-disasters-due-lack-testing/
- Crash of an Airbus A400M transport plane (2015)
  - Caused by error in software configuration
- Mars Climate Orbiter crash (1998)
  - Command software incorrectly used English units instead of metric
- Nissan Airbag Malfunction (2013)
  - System couldn’t determine if passenger seat was occupied
- Northeast blackout (2003)
  - Programming error caused alarm failure

Goals – Correctness

- We want to write correct and maintainable programs
- What does it mean for a program to be correct?
  - Matches specifications
- What are ways to achieve correctness?
  - Principled design and development
  - Abstraction and modularity
  - Documentation
- What are ways to verify correctness?
  - Reasoning about code
  - Testing
  - Automated verification

Writing Correct Code is Hard



- "Program testing can be used to show the presence of bugs, but never to show their absence!" Edsger Dijkstra
  - Halting problem
  - *For any non-trivial property of partial functions, no algorithm can decide whether or not a program (TM) M computes a partial function with that property.*
    - Rice's Theorem
    - TM = Turing Machine
  - Why?
    - See http://blog.paralleluniverse.co/2016/07/23/correctness-and-complexity/

Writing Correct Code is Hard

- That doesn’t mean we can’t do it in specific cases
- *The possibilities as to what one may do are immense. One of our difficulties will be the maintenance of an appropriate discipline, so that we do not lose track of what we are doing.*
  - Alan Turing

- *Almost all (92%) of the catastrophic system failures are the result of incorrect handling of non-fatal errors explicitly signaled in software… In 58% of the catastrophic failures, the underlying faults could easily have been detected through simple testing of error handling code*
  - Ding Yuan et al., Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-Intensive Systems, 2014

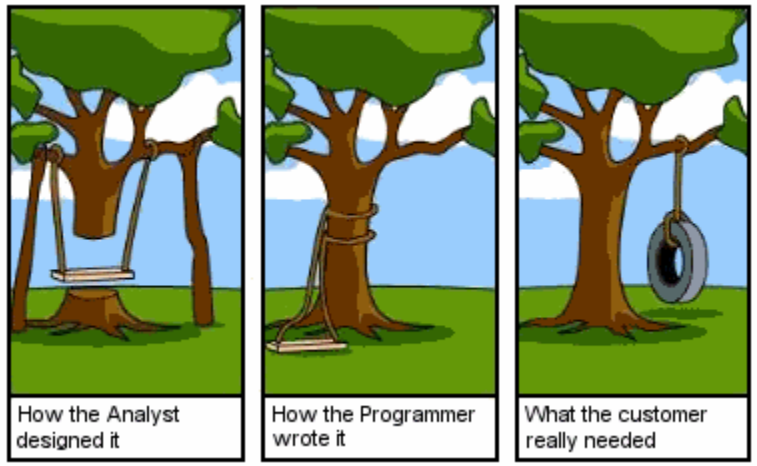
Goals – Maintainability

- What does it mean for a program to be maintainable?
  - Well-organized
  - Consistent
  - Well-documented and understandable
  - **Open for extension but closed for modification**
- Example:
  - Suppose we have an editor that manipulates two-dimensional shapes
  - We have coded Square and Circle objects and have tons of code that manipulates these objects
  - It should be “easy” to add Triangle (open for extension) without any changes to the existing code that manipulates shapes (closed for modification)

Goals – Maintainability

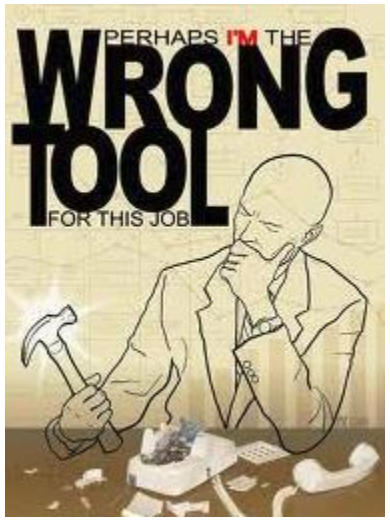
- What are ways to achieve maintainability?
  - Principled design and development
  - Abstraction and modularity
  - Correct object-oriented approach, especially polymorphism, facilitates achieving maintainability
  - Documentation!

Software Engineering



- Building “correct” software is difficult!
- Large software systems are enormously complex
  - Lots of moving parts
  - Ever-changing requirements, interfaces, devices, etc.
  - Often face problems scaling up
  - Software is often put to new uses, sometimes without relevant experience
- Software engineering focuses on:
  - Mitigating and managing complexity
  - Managing change
  - Handling software failures
  - Handling software project failures

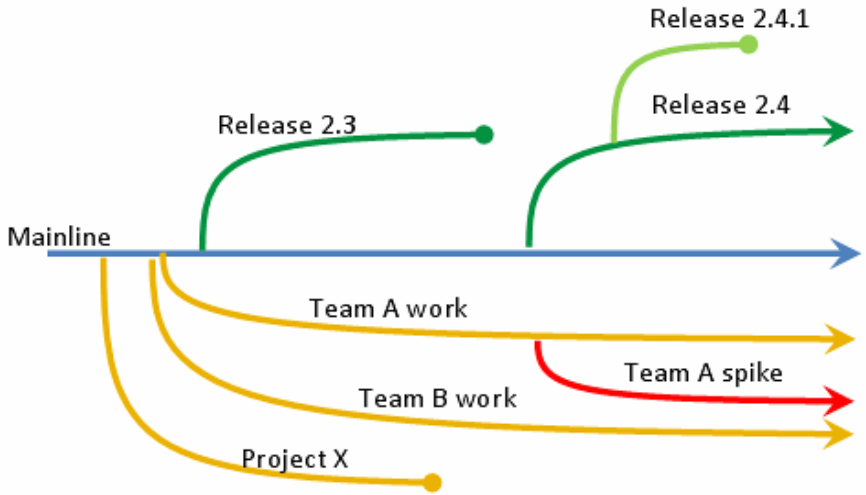
A Summary of Tools



- Principles are more important than tools!
- However, good tools are important

- Java – an industry standard object-oriented programming language
- Eclipse – powerful integrated development environment (IDE)
- Git – version control (more on this coming up...)
- JUnit 5 – testing framework for Java
- Submitty
  - Homework Submission Server
  - Developed by RPI and RCOS, a system for submitting assignments, auto-testing such submissions, assigning grades, etc.

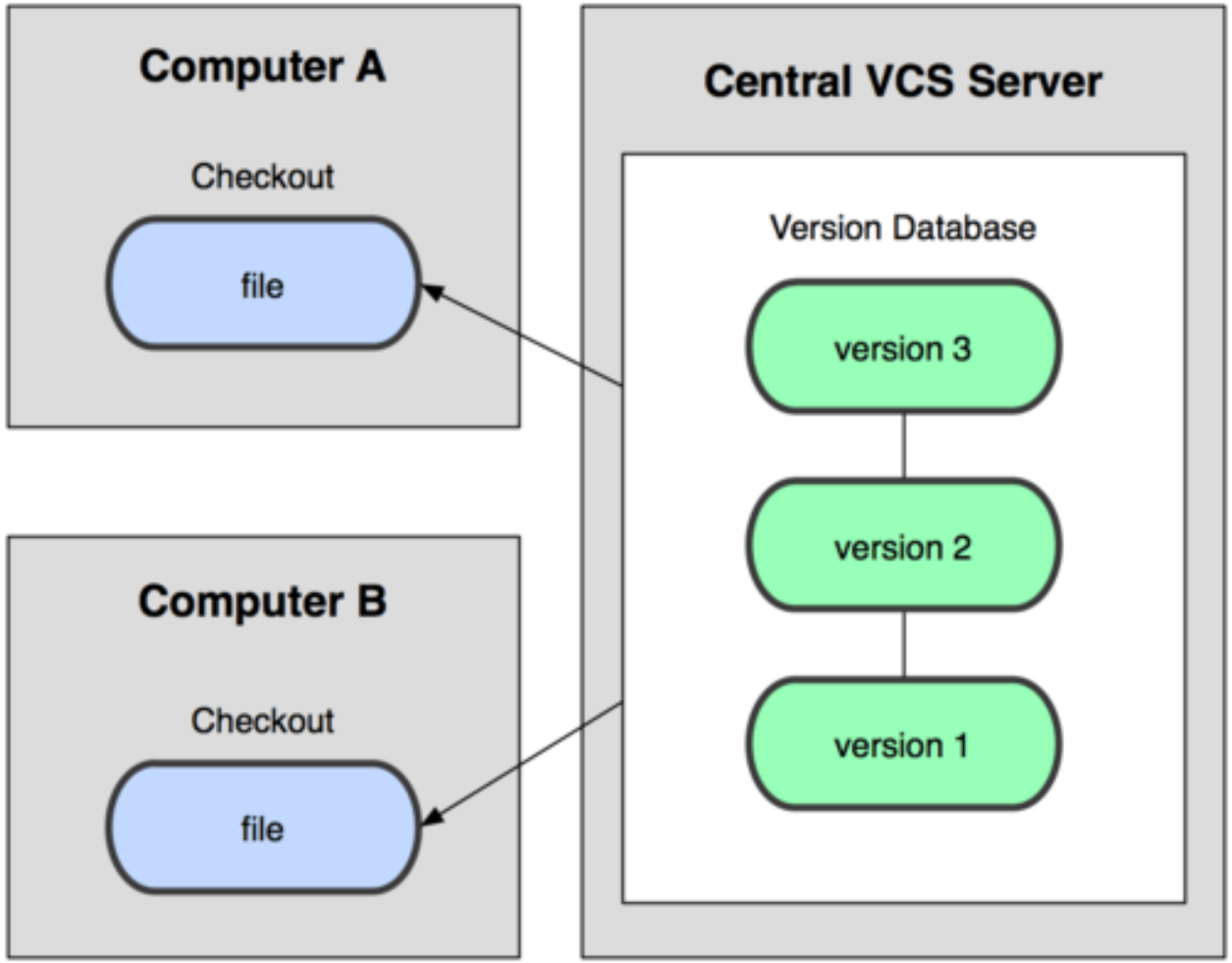
Version Control



- What does a version control system do?
- Records changes to a set of files over time
- Manages changes by a single user or multiple users
- Maintains a centralized repository of files, from which users may checkout all or some of the files on a local machine
- Reverts to older versions, reviews changes made over time, tracks all changes, reviews who introduced what issues, etc.

- We will use Git,
  - SVN is a popular alternative

Version Control – Basic Schematic



Version Control – Key Functions

- Checkout a set of files, from a centralized repository to a local machine
  - Typically just done once unless you wish to revert to an older version
- Add new files to a repository, i.e., to version control
  - Store newly created local file(s) in the centralized repository
  - Enables other users to see/obtain/modify files
- Commit files, i.e., push changes to the repository
  - In other words, commit your changes
- Update files, i.e., obtain changed file(s) from the repository
  - In other words, get your local machine or instance up to date

Version Control – Merges

- Merges occur when multiple users have modified the same file
  - Changes must be merged together into the file in the centralized repository
  - Hopefully, the changes do not overlap or conflict
  - If a conflict does occur, a merge must be performed manually
- In this class, we hope you will not need to worry about merges
  - Each of you will have your own repository on the Submitty server
- If you think you require a merge, let me know....

JUnit testing framework

- JUnit is a unit testing framework for Java that supports writing and running unit tests

- What is unit testing?
  - Testing in which the scope of the test is one unit (or module or component)
  - Example units include a subroutine, function, class, file, etc.
- In object-oriented programming, the smallest testable unit is the class
  - Unit testing is essentially class testing
- After unit testing, we have integration testing, then system testing

Junit 5.x

- JUnit uses annotations to specify test runs:
- @BeforeAll – static method to configure the test run
  - The JUnit framework runs this method before all tests
  - This method creates an instance of the class being tested
- @Test – this annotation marks a method as a test method
  - The JUnit framework runs this method as a test
- Test methods:
- assertEquals(expected result, actual expression, message)
- assertTrue(Boolean expression, message)

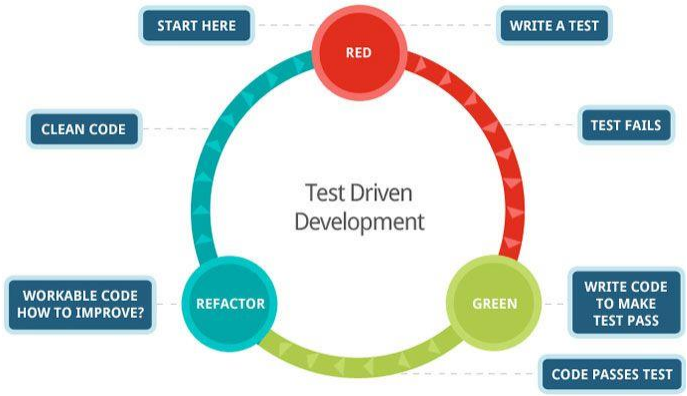
Junit 5.x Example


```
public class SaleTest {
  private static Sale sale = null;
  private static double ITEM_PRICE = 2.5;
  @BeforeAll
  public static void setupBeforeClass() { sale = new Sale(); }
  @Test
  public void testGetTotal() {
    sale.makeLineItem("item1", 1, ITEM_PRICE);
    sale.makeLineItem("item2", 2, ITEM_PRICE);
    assertEquals(7.5, sale.getTotal(), "sale.getTotal()");
  }
```

Unit Testing

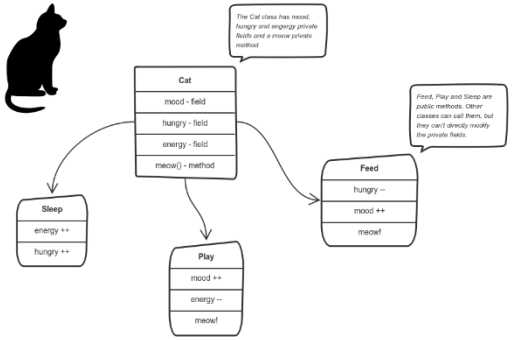
- Modern software development methodologies, such as Extreme Programming (XP) and the Unified Process (UP), place great emphasis on unit testing
- They advocate Test-Driven Development (TDD)
  - test-first development
- Key point is that developers first write the unit tests
  - before the unit (class) being tested is actually implemented!

Test-Driven Development (TDD)



- Write unit tests first! Why?
  - The unit tests actually get written
  - Programmer satisfaction that leads to more consistent test writing
  - Clarification of correct interface behavior
  - Forces one to think of necessary inputs and outputs, as well as “corner cases”
  - Repeatable, automated verification
  - More confidence in making changes to individual unit (i.e., “unit tests all passed”)

Object Oriented Programming (OOP)



- Object
  - Software *"thing"* that contains state and methods
    - State – maintains values
    - Methods – perform operations
- This class will mainly focus on object oriented programming concepts
  - Inheritance
  - Encapsulation
  - Abstraction
  - Polymorphism

Object Oriented Programming

- OOP requires discipline
  - Design Patterns
  - SOLID
    - **S**ingle responsibility principle
      - A class should only have a single responsibility
    - **O**pen–closed principle
      - Software entities should be open for extension but closed for modification.
    - **L**iskov substitution principle
      - Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.
    - **I**nterface segregation principle
      - Many client-specific interfaces are better than one general-purpose interface.
    - **D**ependency inversion principle
      - One should depend upon abstractions, [not] concrete implementations

It's not the only way to think about software

- *Object Oriented Programming puts the Nouns first and foremost. Why would you go to such lengths to put one part of speech on a pedestal? Why should one kind of concept take precedence over another? It's not as if OOP has suddenly made verbs less important in the way we actually think. It's a strangely skewed perspective.*
  - Steve Yegge
- Other approaches
  - imperative in which the programmer instructs the machine how to change its state
    - procedural which groups instructions into procedures,
  - declarative in which the programmer merely declares properties of the desired result, but not how to compute it
    - functional in which the desired result is declared as the value of a series of function applications,
    - logic in which the desired result is declared as the answer to a question about a system of facts and rules,
    - mathematical in which the desired result is declared as the solution of an optimization problem

What happens next?

- Review the course syllabus and schedule
- Be sure you are on Submitty
  - Log in with your RCS ID
    - The first part of your @rpi.edu email address
  - Check the Submitty forum regularly for announcements
  - Quizzes, homework, etc. will be announced there
- Make sure you are registered for the course on SIS
- Be sure you can receive email from kmkuzmin@gmail.com
  - Check your RPI e-mail at least once a day
- Start programming in Java