Recitation 14

Principles of Software

Brandon Rozek
`rozekb@rpi.edu`

Rensselaer Polytechnic Institute, Troy, NY, USA

April 2022

Outline

Two Things:

- Final Thoughts
- Final Exam Review

Final Exam

- The final exam is Thursday May 5th from 3-6pm.
- The exam will be *cumulative*.
- No office hours after today.
- Test 2 should be graded by Monday.

What is this class about?

Principles of Software is mainly about writing *correct* and *maintainable* software.

How do we accomplish this?

- Reasoning about code
- Specifications
- Testing
- OO Features (Polymorphism)
- Design Patterns
- Refactoring

Trust between Developers

But more importantly, this is about **trust**.

Trust that:

- Coding to the specification won't introduce unexpected changes when an update occurs.
- *Formal Verification* of software in order to have a strong guarantee that the software meets its specification.
- Informal verification when it isn't worth the time to formally verify it in Dafny or Coq.
- That we can collect technical debt now, but know we can refactor to pay it back later.

Business Builds on Trust

Clients need to trust your software as well:

- This feeds into usability, your software needs to enable users to accomplish their goals.
- Software interacts with the real world, don't break it!
    - This leads into *safety-critical* design.
    - Or even plain reliability.

Failing to gain trust will likely mean that your clients would go to a competitor instead...

Societal Advancements

Large societal advancements may be hindered by trust.

- Currently my inbox is being bombarded with email thread about the use of electronic voting systems.
- Would you trust these systems over paper ballots?

More Bugs

- We talked about the Mazda's forced to listen to NPR.
- How about the 2018 Chevy Malibu software bug that caused the fuel injectors of the engine to be disabled?

It's important to assess potential risk in your software project and take *appropriate* measures to balance trust and new features.

Testing $\rightarrow$ Specifications $\rightarrow$ Informal Proofs $\rightarrow$ Formal Proofs

Final Exam Review

Reasoning

Make sure you review:

- Forward Reasoning
- Backwards Reasoning
- Reasoning over **Loops**
  - Use induction!
  - Showing *partial correctness*
  - Proving *termination*

Specifications

- What are all specification tags that this class requires?
- What makes one specification *stronger* than another?
- What does specification strength have to do with substitutability?

Generics

- What are they?
- Type erasure
- Bounded Types
  - `<Type extends SomeType>`
  - `<Type super SomeType>`
- Wildcards

ADT

- What makes something an ADT?
- Representation Invariant
- Representation Exposure
- Abstraction Functions
- Difference between immutable and mutable and how are immutable data structures helpful?

Testing Strategies

- White Box Testing
- Black Box Testing
- Equivalence Partitioning
- Boundary Value Analysis
- Be able to quickly draw a CFG and produce minimal test cases for them!
- Coverage: Statement vs Branch

Design Patterns

- Factory
- Interning
- Observer
- Visitor
- Singleton
- Wrapper
- Composite

More to Know

- Equality and Hash codes
    - What is a valid hashcode?
- Function Subtypes
- Usability
- Software Lifecycle
- GUI Development

Lets go over some questions in the sample exam...

Any Questions?