CSC 217 Lab 05

Inspections & Debugging

Lab Overview

- Deadlines and Reminders
- Inspection Overview
- Debugging Reminders
- New Team Introductions
- Project Selection for starting Lab 05
- Lab Activities
  - Extract User superclass
  - Inspect RegistrationManager code
  - Test and debug RegistrationManager
- Lab Wrap-Up

Deadlines and Reminders

- Deadlines
  - Check lab write-up for deadlines!
- Reminders
  - When should you start the project?
  - Jenkins Console Output
  - Project **commit messages are graded**
  - New lab groups! Note pair programming in lab commit messages.
  - Don’t forget to breathe

Inspection (or Reviews)

Process where **software artifacts** are reviewed by developers, managers, and/or customers for comment or approval [IEEE 1990]

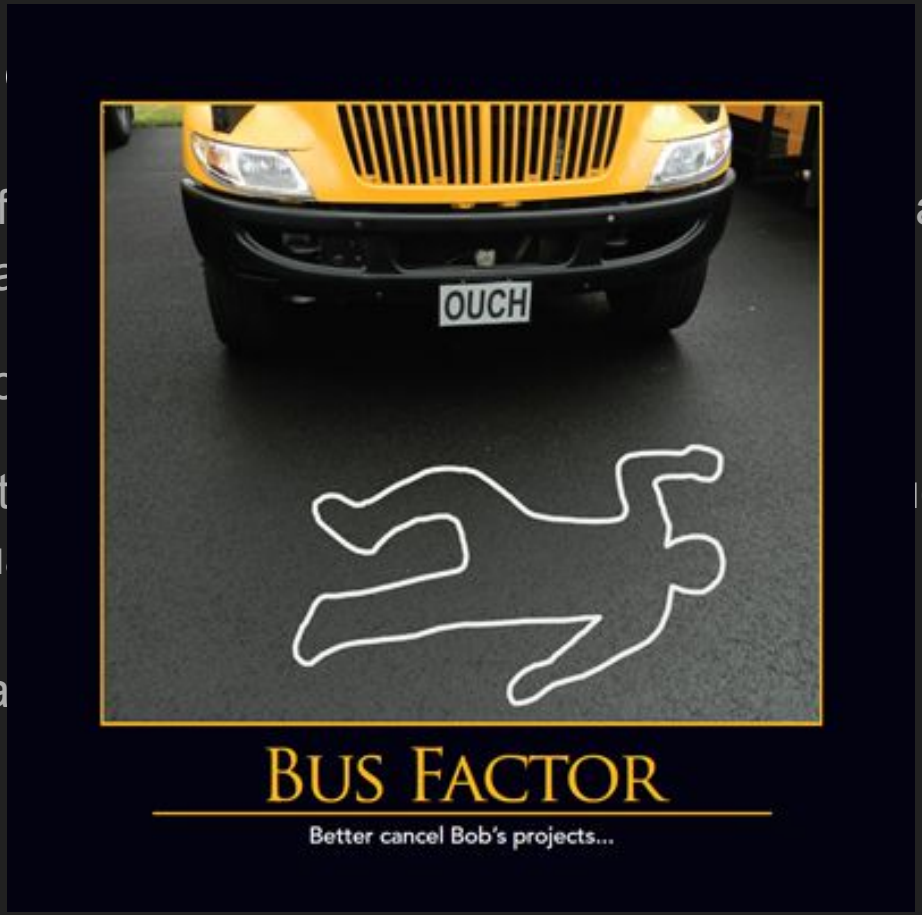
- Detect errors/inconsistencies
- Clear intention
- Design/software meets requirements
- Developed in uniform manner using standards

Inspection Ben

Inspections find a diff alysis. They find bugs where there is a

Additionally, inspectio

- Learn about syst es
- Produce high qu
- Promote culture
- Reduce “truck-fa



Inspection Roles

- **Author:** person who developed the artifact
- **Inspectors:** inspect artifact. Everyone except the author.
- **Moderator:** keeps everyone on task. Typically a member of a quality assurance team
- **Scribe:** takes notes of issues found during inspection
- **Reader:** person who interprets artifact for inspectors

Reading Code

```
for (int i = 0; i < array.size(); i++) {

   //do something with element at index i

}
```

Bad Reading: Reading the code as written

For int i is equal to 0, while i is less than array.size(), increment i...

Good Reading: Abstracting the code to a higher level

Every element of the array is modified in the following way...

Inspection Meetings

(Note: there is more to holding a formal inspection than just the meeting, but those details will be in a later class)

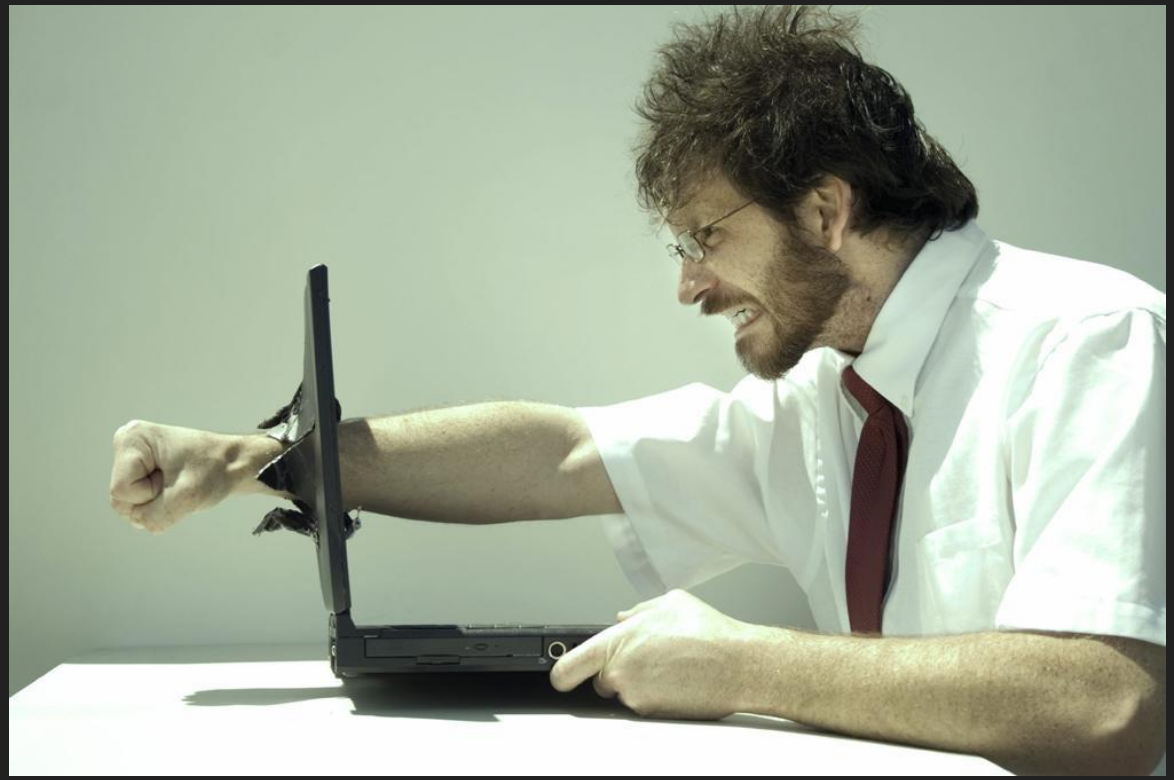
- Moderator calls meeting to order
- Reader interprets artifact in sections (for code, by method or chunk)
- Inspectors note problems or ask questions
- Author answers questions
- Scribes record issues

Good meetings are useful and don't waste time

Inspection Guidelines

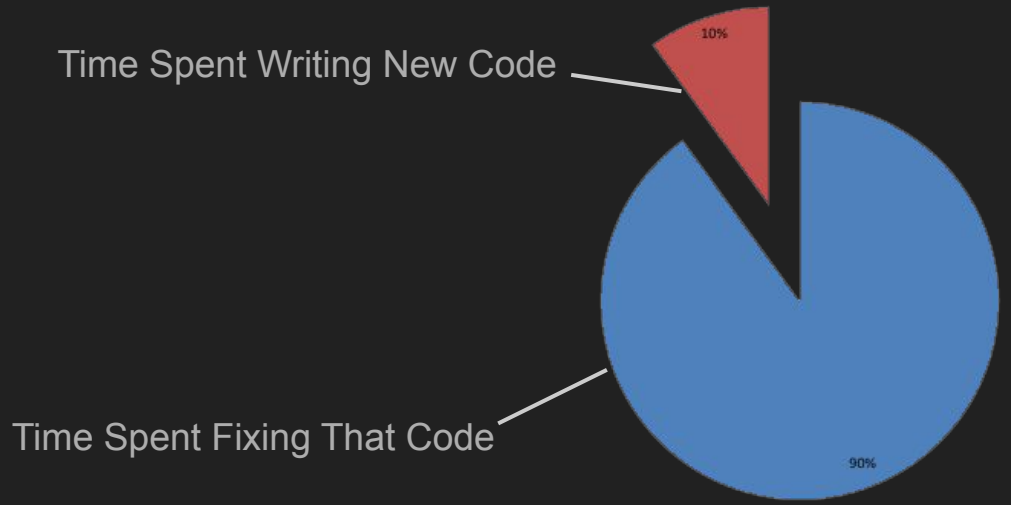
- Do NOT correct faults, instead give a course of action (the goal is to find all of the defects, you can correct them later)
- Author answers questions and does NOT justify decisions
- No personal attacks on authors (one day the author will be you!)
- Focus on important issues, less on style issues (note and move on)
- Inspections should last no longer than 2 hours in one session

Debugging! (Quick Reminders)



What is Debugging?

- Stepping through code one line at a time to troubleshoot problems
- The **MAJORITY** of software development is testing, debugging and bug-fixing

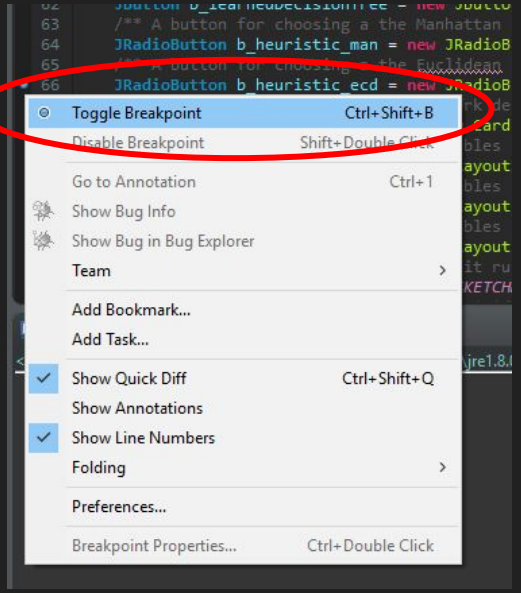


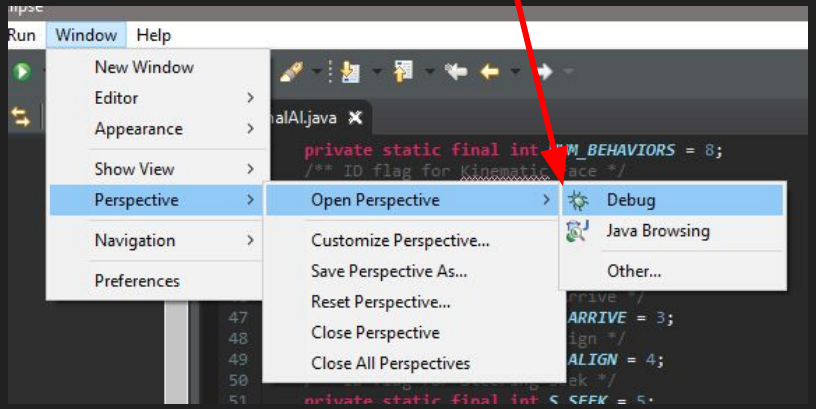
Source Level Debugging

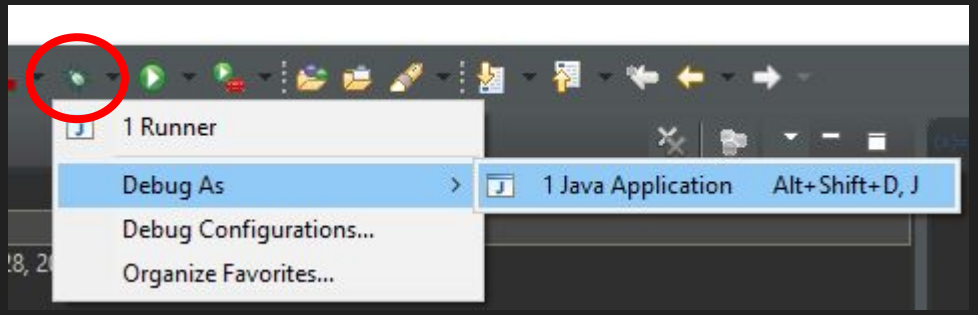
- Symbolic debugging lets you single step through program, and modify/examine variables while program executes.
- Lets you suspend program when errors/exceptions happen and allows you to trace back where the problem came from.
- Built into most IDEs.

Debugging in Eclipse

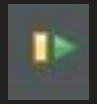
- Anything that can be run in eclipse can be debugged.
- Add breakpoints to stop your code at certain points.
- The debugging perspective.

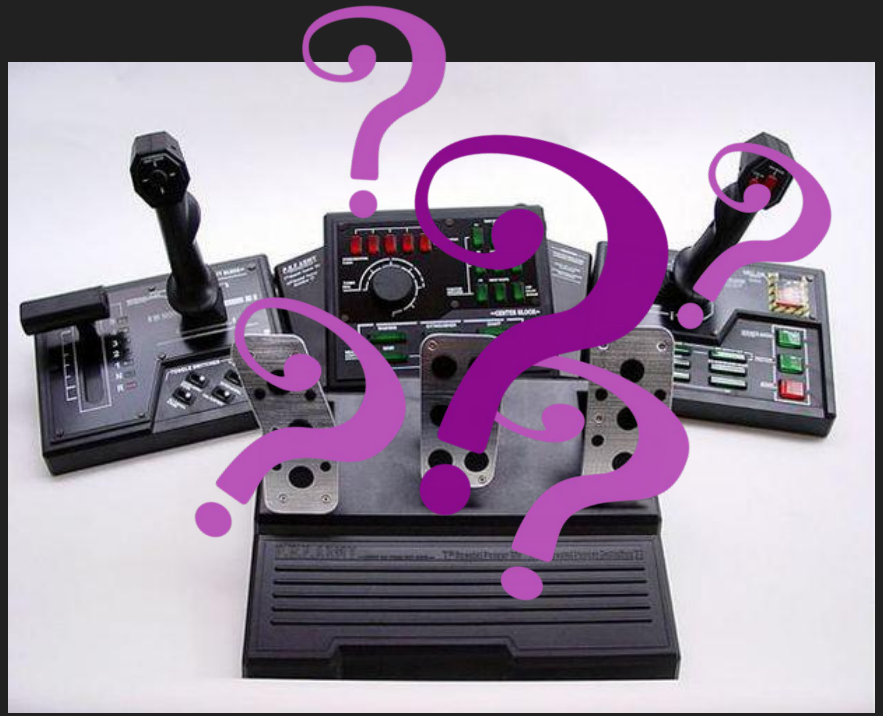




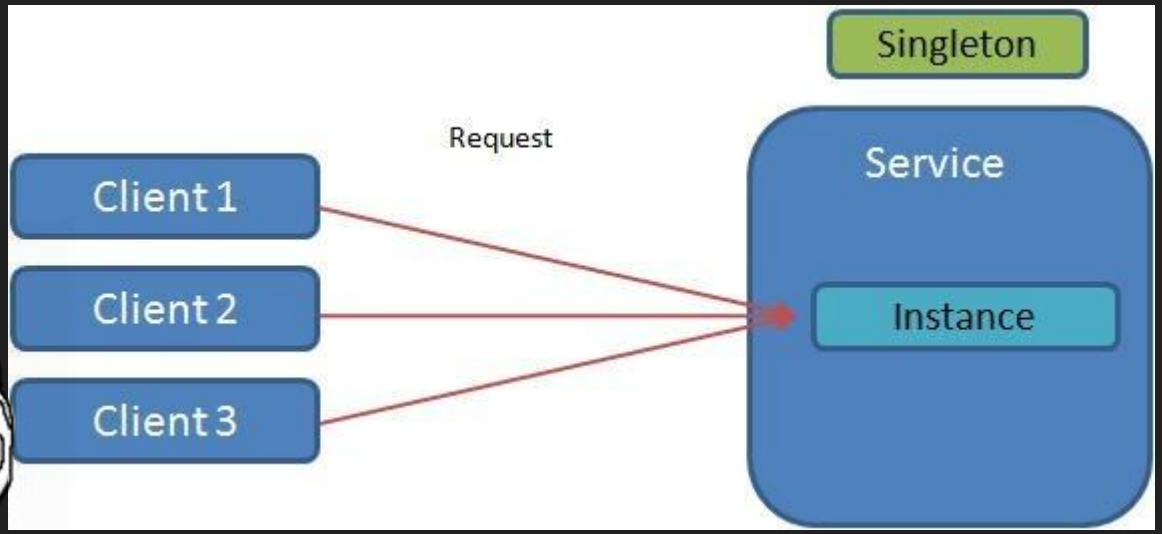


Eclipse Debugging Controls:

- Continue 
- Suspend 
- Stop 
- Step into 
- Step over 



Bonus Note: Singleton Pattern



Implement with caution!

Team Introductions

Welcome to your team for CSC 217 Lab 5 through Lab 8!

Take a few minutes to get to know each other. Make sure to do the following:

- Exchange contact information and create common chat group
- Discuss what tasks you prefer and which you want to practice
- **Move one teammate's Lab 4 PackScheduler** into your new Lab 5 repo
  - Decide whose project to use (don't just start pushing yours!)
  - Follow the Git guide for GP1 -> GP2 (or GP2 -> GP3)
  - Briefly inspect previous code (share tricks or solve problems)

Lab Overview

Lab activities are in Moodle

- Setup PackScheduler in new repo!
- Extract superclass
- Inspect code
- Debug code
- Test code
- All while **pair-programming** with your new team!

But Wait! There’s More...

Wrap-Up

General Wrap-Up

- Deadline Reminder (see board)
- Exchange contact information with your partner
- Make a plan for finishing up the lab

Participation Outside of Lab (Guess which the teaching staff prefer?)

- If you pair program/design, **note that in the commit comments so everyone gets credit!**
- If you split the work, at least one contribution by each partner

REMINDER: We are expecting a significant contribution from all team members outside of lab!

- If you pair program/design, you **MUST** note it in your commit messages or there will be deductions
- Students who don’t allow their partners to contribute will receive deductions
- Students who don’t contribute will receive deductions

Record Tasks & Owners

Tasks only get done when someone owns them!

Identify the tasks required to complete Lab 5

- Edit README.md to list the tasks required to complete Lab 5
- Add an owner to each task
- Add a deadline to each task

Deadlines should be at least 48 hours before the lab deadline so team members can help out and finish the lab if a team member runs into issues.

Notify team early if you run into problems with your tasks!