CSCI 3901: Software Development Concepts

Mike McAllister

Winter 2023

Acknowledgement

- We are based in Mi'kma'ki, the ancestral and unceded territory of the Mi'kmaq people.
- This territory is covered by the treaties of Peace and Friendship, which Mi'kmaq and Maliseet people first signed in 1725.
- The treaties did not deal with surrender of lands and resources but in fact recognized Mi'kmaq and Maliseet title and established the rules for what was the be an ongoing relationship between nations.
- We also acknowledge the contributions of the African Nova Scotian community in building the province for over 400 years.

DALHOUSIE UNIVERSITY
Inspiring Minds

CSCI 3901

- Ensure that you are able to implement key elements of
  - Csci 2110 – data structures & algorithms
  - Csci 2132 – software development
  - Csci 2141 – introduction to database systems
  - Csci 3130 – software engineering
- Know how to implement and design from the basics

Learning Outcomes

- **Data Structures and Algorithms**
  - Use abstract data types (ADTs), including lists, stacks, queues, maps, dictionaries.
  - Implement fundamental data structures, such as linked lists, trees, graphs, and hash tables.
  - Implement traversals, recursive search and state-space exploration algorithms.
  - Implement simple iterative and recursive algorithms to solve moderately simple tasks.
  - Select the appropriate data structure to implement a given ADT under a given set of constraints.
  - Select and use appropriate abstract data types, data structures, and algorithms to solve real-world problems.

Learning Outcomes

- **Databases**
  - Describe the properties of multiuser database transactions (ACID).
  - Describe the purpose, function, evolution, classification, and building blocks of data models and data modeling.
  - Describe the basic components of a relational model and how relations are implemented.
  - Derive business rules from requirements' specifications and translate these rules into database table and relationship designs.
  - Use SQL data definition and manipulation operations.
  - Construct an entity relationship diagram (ERD).
  - Describe normalization and denormalization, and their role in database design.
  - Perform normalization and denormalization on a database.

Learning Outcomes

- **Software Engineering**
  - Design a software system and prepare detailed design documentation.
  - Implement moderate-sized programs.
  - Create unit tests for a software development project.
  - Effectively debug a program.
  - Understand and apply the concepts of code coverage in testing.
  - Apply standard software processes for version control.
  - Create readable and maintainable code
  - Apply standards of good program design through the SOLID principles.
  - Identify code smells and know how to remedy them.

Grading

- Labs – 10%
- Assignments – 40% (5 assignments, equally weighed)
- Module quizzes – 20%
- Participation – 10%
  - 2% on academic integrity quiz by January 22, 2023
  - 8% on visible engagement. It's not an attendance mark.
- Final project – 20%

- No late assignments accepted
- Use the university's policy on self-declaration of short-term illness
  - Can be used twice in the term for up to 3 days of illness
  - Gives a 3 day extension on assignments. Makes a lab exempt from grading.

Important Dates

- February 3
  - Munro day
- February 20 -- 24
  - Winter study break
- April 7
  - Good Friday
- April 11
  - Last day of classes

- February 2
  - Last day to add or drop courses without penalty
- March 13
  - Last day to drop courses without academic penalty

DALHOUSIE UNIVERSITY
Inspiring Minds

Tentative Schedule

- Consult syllabus

Course textbook

- No official textbook
  - We cover so much material that you would need several
- If I had to choose one:
  - Course text from csci 2110
  - “Data Structures Outside-In With Java”

Homework

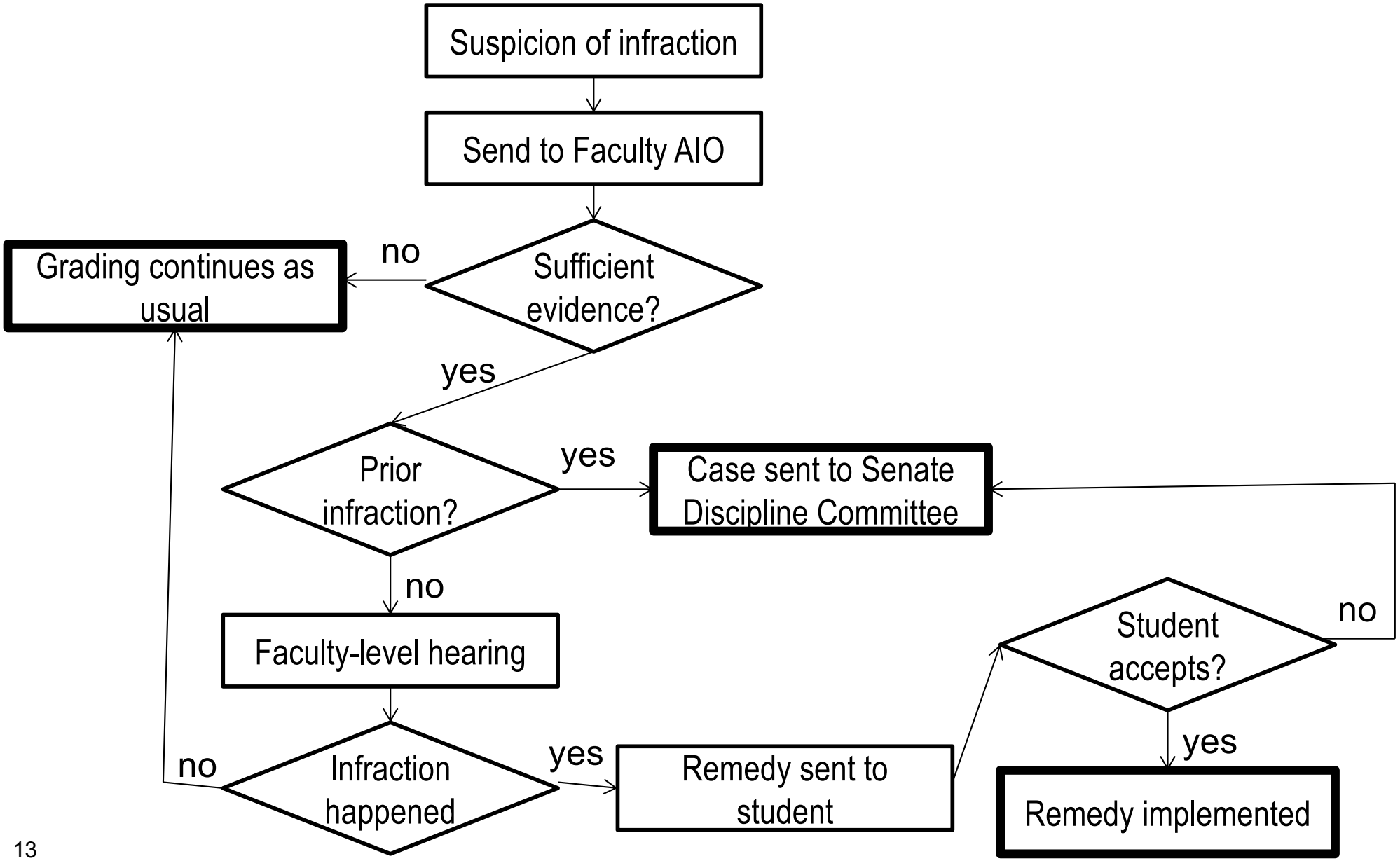
- Get Java working on your computer
- Select and install an IDE for Java
- Install a secure shell program and a file transfer program
- Write a “hello world” program using the IDE that can also work on timberlea.cs.dal.ca
  - Compile foo.java on timberlea with “javac foo.java”
  - Run foo.class on timberlea with “java foo”
- Write a program that uses one loop to print the integers from 1 to 10 on individual lines and with a # symbol before the numbers 5 and 6
- Read up on / review linked lists, stacks, queues, and abstract data types

DALHOUSIE UNIVERSITY
Inspiring Minds

Academic Integrity

- Do the module by January 22
- Attribute the work and help that you get from others
- Learn to work collaboratively and safely
  - Don’t have academic integrity fears isolate you from talking with others
- If in doubt, cite and ask!

Academic Integrity Officer (AIO) Process



Concluding Reminders

- Accessibility

- Responsible computing
- Code of Conduct
- Culture of Respect
- Scent-free university

Lessons from last time

- Understand the problem early
  - Ask early about parts that seem unclear
- Test your program before submitting it
- Document every program
  - Internal comments, not just one per method
  - External documentation that doesn't just duplicate the assignment text
- When submitting work
  - Do not upload your entire development directory

Lessons learned from last time

- **Best practice coding**
  - Do not write huge methods
  - Do not leave class attributes as public
  - Avoid hard-coding constants
    - Create static final variables to hold the values then use the variable names in the code
  - Use braces around the body of all if and loop statements
  - Aim to have classes in their own files
    - One class per file unless the class is a private support class
  - Avoid making class attributes static unless it is a static final constant

Lessons learned from last time

- Adhere to the input and output specifications
  - Keep debugging print statements inactive
  - Don’t “improve” on the user interface unless the assignment gives you that leeway
  - Don’t decide that the API needs to be changed

Understanding a problem (class' list)

- What are the inputs needed?
- Algorithms?
  - Standard known ones?
  - Flexibility
  - Pseudocode to understand better
- Constraints
- What does the output look like?
- Mode of transport?
- Characteristics of the solution. What is important about it?
- What is required vs would be nice to have
- What are failure conditions and how do I handle them?
- Scalability?
- Multiple stops for navigation?
- Costs and which can I afford: efficiency, space, use by the user,
- What do I value most? --- …ilities: maintainability, scalability, usability maintainability, portability, …