1004 Final Review Slides

Table of Contents

1. Structure of the Final
2. How to Study for the Final
3. Important things to Review
4. Big-O Notation
5. Parameters vs Arguments
6. Checked vs Unchecked Exceptions
7. Static fields and methods
8. Interfaces
9. Class Interaction
10. Inheritance and Polymorphism
11. Circuits
12. Turing Machines
13. Computer Networks
14. Von Neumann Architecture

1. Structure of the Final

Structure of the Final

There are three sections on the final:

1. Multiple Choice
    a. 40 questions (1pt each)
        i. 10 are term definitions and 25 come from test banks you've seen prior on the quizzes
        ii. Which means only 5 questions are not like ones who have seen in the past
2. Short Answer:
    a. 15 questions (3pts each)
3. Free Response Java Coding Questions
    a. 3 questions (5pts each)

2. How to Study for the Final

How to Study for the Final

1. Read through the suggested sections on the Lecture 28 slideshow
2. Review examples from class and understand how they work
3. Review Quizzes
4. Review OH Slides
5. Review Additional Resources posted throughout the semester

3. Important things to Review

Important things to Review

Topics that have appeared on your quizzes will be the most tested. You should at least be conceptually familiar with all topics covered throughout the semester but be reasonable about your own personal expectations for the final exam.

Topics covered in this review are only a subset of the entire course content covered throughout the semester.

4. Big-O Notation

Big-O Notation

Big-O notation is used to demonstrate the efficiency of algorithms in terms of their runtime and space like complexity.

You are not required to know how to determine Big-O costs yet. But you should be aware of the costs associated with these algorithms

Selection Sort $O(N^2)$

Insertion Sort: $O(N^2)$

Linear Search: $O(N)$

Binary Search: $O(\log N)$*

*Note when CS people say logN they mean $\log_2 N$

5. Parameters vs Arguments

Parameters vs Arguments

Parameters are the lists of inputs for methods, explicit parameters are anything you write. The implicit parameter is a reference to the instance of the class that you are operating on.

To think of parameters better: for y = f(x)

‘x’ is a parameter for the function

Arguments are what you pass into methods during runtime. Your argument list must match the parameter list in terms of the number of elements you pass in and their data types.

To think of arguments better: for y = f(2)

‘2’ is an argument for the function

6. Checked vs Unchecked Exceptions

Checked vs Unchecked Exceptions

Checked Exceptions

- You cannot compile a java program without acknowledging them
- Either use a try-catch block or the throws clause on a method signature
- FileNotFoundException is an example
- Compile Time Exceptions

Unchecked Exceptions

- You can compile a java program without acknowledging them
- Do not need to be caught and handled but is good practice
- NullPointerException is an example
- Runtime Exceptions

7. Static fields and methods

Static Fields and Methods

Static Fields

- Can be accessed via ClassName.fieldName
- Usually constants ex Math.PI
- Each class as the same copy if you modify one of them you modify all of them

Static Methods

- Can be accessed via ClassName.methodName
- The Main Method is Static
- You do not need an instantiation of the class to use the methods
- Math.sin() - static method call, you’ve never instantiated an instance of the Math class

8. Interfaces

Interfaces

An Interface in Java programming language is defined as an abstract type used to specify the behavior of a class. An interface in Java is a blueprint of a class. A Java interface contains static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not the method body. It cannot have a method body. Java Interface also represents the IS-A relationship.

9. Class Interaction

Class Interactions

Classes can interact with each other. For example in programming project 4, Your Deck and Card classes interacted with each other because the main structure for the Deck class implementation was an array of type Card. Classes interact with each other to build a larger project. In large scale industrial projects, having multiple classes interact with each other is the standard as each individual class solves a particular problem and then those solutions come together to solve larger problems and so on. More intricate class interactions come from inheritance.

10. Inheritance and Polymorphism

Inheritance and Polymorphism

Inheritance in Java is shown through the extends keyword, when a subclass extends a superclass the subclass has inherited all public methods and attributes of the superclass. Meaning if the superclass has a public method called playTurn() the subclass now as the same playTurn() method.

Polymorphism is the ability to take multiple forms, Java has static polymorphism and dynamic polymorphism, static polymorphism occurs at compile time and dynamic polymorphism occurs at runtime.

Inheritance and Polymorphism

Static Polymorphism

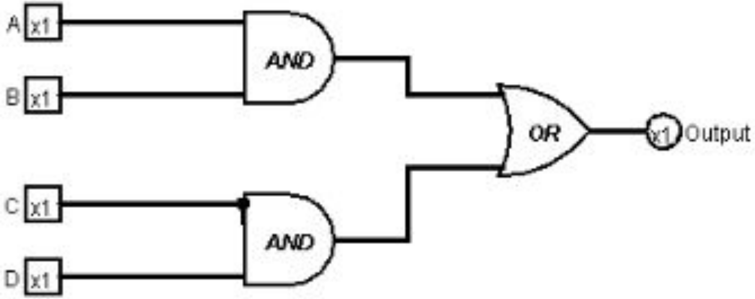
- The methods use the same name but the parameter varies.
- Method must exist for the DECLARED type otherwise a compiler error will occur
- Method overloading

Dynamic Polymorphism

- Uses the method of the instantiated type during runtime
- Method overriding to change implementation

11. Circuits

Circuits



Equivalent Boolean: (A&&B) || (C&&D)

Three main logic gates for use: AND, NOT, OR. Be able to read truth tables and circuits to construct boolean expressions. Be able to construct circuits from boolean expressions.

Given the following boolean expression construct a circuit representation: (A&&B) || (C&&D)

1. First Step: Determine the number of Inputs you have (In this case we have 4: A, B, C, and D)

2. Second Step: Determine what logic gates you'll need (In this case we will need 2 AND Gates and an OR gate)

3. Third Step: Split into smaller circuits, construct and then combine results together

12. Turing Machines

Turing Machines

A Turing machine is a theoretical model of computation that includes a (conceptual) tape extending infinitely in both directions. The tape is divided into cells, each of which contains one symbol.

The Turing machine is designed to carry out only one type of primitive operation. Each time such an operation is done, three actions take place:

1. Write a symbol in the cell (replacing the symbol already there).

2. Go into a new state (it might be the same as the current state).

3. Move the “read head” one cell left or right.

Turing Machines

Church-Turing Thesis: "If there exists an algorithm to do a symbol manipulation task, then there exists a Turing machine to do that task"

State Diagrams for Turing Machines: You may be asked to draw State Diagrams that coordinate with the instruction sets from time to time.

1. Draw a bubble for each state, in the last example there are two state so 2 bubbles
2. Draw arrows to bubbles with labels that contain (current symbol, next symbol, direction)

13. Computer Networks

Computer Networks

“Know the Stack and some Protocols”

1. Physical Layer: RS232, 100BaseTX, ISDN, 11
2. Data Link Layer: RAPA, PPP, ATM, Fiber Cables
3. Network Layer: IPv5, IPv6, ICMP, IPSEC, ARP, MPLS
4. Transport Layer: TCP, UDP
5. Session Layer: NetBIOS, SAP
6. Presentation Layer: MPEG, ASCH, SSL, TLS
7. Application Layer: HTTP, FTP, POP, SMTP, DNS

14. Von Neumann Architecture

Von Neumann Architecture

The Arithmetic Logic Unit: The arithmetic/logic unit (ALU) is the subsystem that performs such mathematical and logical operations as addition, subtraction, and comparison for equality.

The Control Unit: The most fundamental characteristic of the Von Neumann architecture is

the stored program—a sequence of machine language instructions stored as binary values in memory. It is the task of the control unit to (1) fetch from memory the next instruction to be executed, (2) decode it—that is, determine what is to be done, and (3) execute it by issuing the appropriate command to the ALU, memory, or I/O controllers. These three steps are repeated over and over until we reach the last instruction in the program in your case this command is HALT

Together the Control Unit and the ALU construct what we all know as the Central Processing Unit or CPU

Von Neumann Architecture

Memory: Memory is the functional unit of a computer that stores and retrieves instructions and data. All information stored in memory is represented internally using binary. The memory unit contains two special registers, the MAR which holds the address of the cell to be accessed, and the MDR which contains the data value being stored or retrieved.

Input/Output and Mass Storage: The input/output (I/O) units are the devices that allow a computer system to communicate and interact with the outside world as well as store information for the long term.