Topics covered in the Final

- Java Programming concepts and tools
  - Pre and post conditions
  - Interface
  - Exception
- Programming and algorithms
  - Recursion
  - Time and space complexity
  - Sorting algorithms
    - Selection sort
    - Insertion sort
    - Merge sort
    - Quick sort

Topics covered in the Final (2)

- Abstract data types (ADT)
  - Java interface
  - Specification of
    - Data
    - Operations on the data
  - No implementation
- Implementation of ADT in Java
  - Java class
  - Data
  - Constructor
  - Basic methods

Topics covered in the Final (3)

- List of Abstract data types
  - Vectors
  - Lists
    - Singly linked list
    - Doubly linked list
    - Circularly linked list
  - Binary trees
  - Binary search trees
  - Stack
  - Queue
  - Hash tables

Java Programming concepts and Tools

1.1 Pre and pos conditions and Readability

- A precondition tells when (under what circumstances) a method can be called:

  ```java
  public Ratio(int top, int bottom)
  /* Initialize the receiver to be the fraction
   * whose numerator is the top and whose
   * denominator is the bottom.
   * The bottom cannot be zero.
   * pre: bottom != 0
  */
  ```
1.2. Interface

- There are three aspects for any class:
  - Protocol: a description of the resources that the class provides.
  - Implementation: the code that describes the private state of instances and implements the computational resources of the class.
  - Use: code in a using class causes new instances of the used class to be created and invokes computations on these instances.

A Java interface provides a mechanism to formally specify the protocol.

1.3. Java’s Exception

- When we write a piece of code that we know might produce an exception and we want to catch (handle) the exception, we encapsulate that code in a try block.
- If no exception is thrown by code within the try block (or the methods that are called within the try block), the code executes normally.
- If an exception arises in the try block the execution of the try block terminates execution immediately and a catch-clause is sought to handle the exception. Either
  1. An appropriate catch clause is found, in which case it is executed, or
  2. The exception is propagated to the calling method.

Programming and Algorithms

2.1 Recursion

- Recursion occurs when a method calls itself, either directly or indirectly.
- For recursion to terminate, two conditions must be met:
  - there must be one or more simple cases that do not make recursive calls.
  - the recursive call must somehow be simpler than the original call so that they lead to the base case.

To write a recursive function

- One needs to transform the given problem into a same problem with a smaller size such that the solution can be obtained based on the solution to the smaller problem.
- One needs to identify a boundary problem with a simple solution (i.e., it can be solved without recursion.)

2.2 Time and Space Complexity

- The time and space requirements of a method can be determined by analysis.
- Since these values depend on the data size, we often describe them as a function of the data size (referred to as n).
- These functions are called respectively, the time complexity, time(n), and space complexity, space(n), of the algorithm.

Big-O definition

- If an algorithm has time complexity order n^2 we write time(n) = O(n^2)
- The technical definition of big-O is:
  - time(n) is O(g(n)) if and only if
    \[ \lim_{n \to \infty} \frac{\text{time}(n)}{g(n)} = c, \text{ where } c \text{ is a constant} \]
- In English, this means time(n) grows no faster than the function g(n)
- Two important rules:
  - make g(n) as small as possible
  - g(n) never contains unnecessary constants or terms
Common complexity orders

<table>
<thead>
<tr>
<th>NAME</th>
<th>Big-O</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>O(1)</td>
</tr>
<tr>
<td>logarithmic</td>
<td>O(\log(n))</td>
</tr>
<tr>
<td>linear</td>
<td>O(n)</td>
</tr>
<tr>
<td>quadratic</td>
<td>O(n^2)</td>
</tr>
<tr>
<td>cubic</td>
<td>O(n^3)</td>
</tr>
<tr>
<td>exponential</td>
<td>O(2^n)</td>
</tr>
</tbody>
</table>

2.3 Sorting Algorithms

- Selection sort
  - O(n^2)
- Insertion sort
  - O(n^2)
- Quick sort
  - O(n^2)
- Merge sort
  - O(n \log(n))

Abstract Data Types (ADT)

- all discussed abstract data types
  - Vectors, lists, binary tree, binary search tree
  - Stack, queue, hash table
- All the major implementation techniques
  - Array, linked list
- Time and space complexity of all major methods

4. Vector and its implementation

- Array
- Vector
- How to use an array to implement a container with unlimited size, like vectors
- Time and space complexity of our implementations

7. Format of the Final

- Understanding java programs
  - Debug
  - Program tracing
  - Specify the pre and post conditions
- Programming
  - To implement a given interface
  - To write a recursive functions for solving a given simple program
  - To deal with an exception

Format of the Final (continued)

- Analyze a given algorithm/function
  - Time complexity
  - Space complexity
- Various sorting algorithms
- Use and implementation of discussed ADTs