Data Structures (Spring 2020)

Stack & Queue (3rd Lab)

2020.04.03

Seoul National University

Database Systems Lab

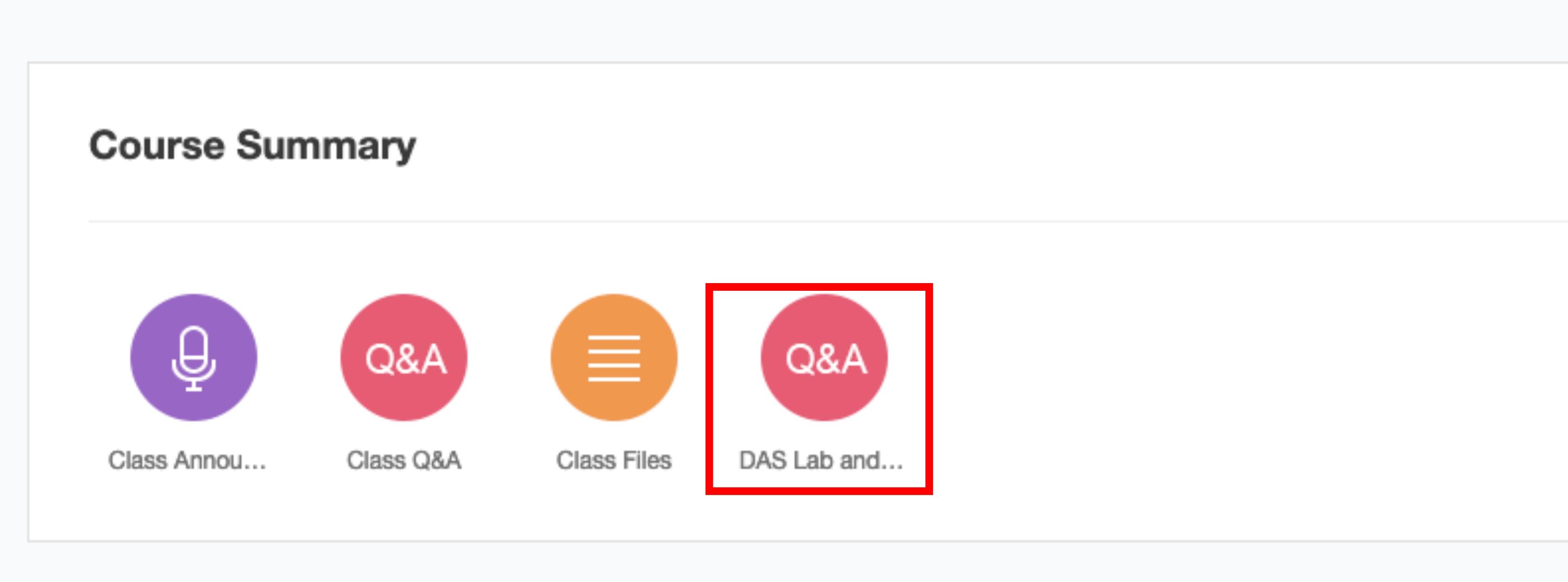
Today's Lab

- Announcement
- Question Summaries

- Stack (array)
- Queue (linked list)

Announcement

- To question
  - eTL: DAS Lab and Q&A ← Do not shame!
  - Mail: das@dbs.snu.ac.kr (Engr. 301-418) ← May fast response



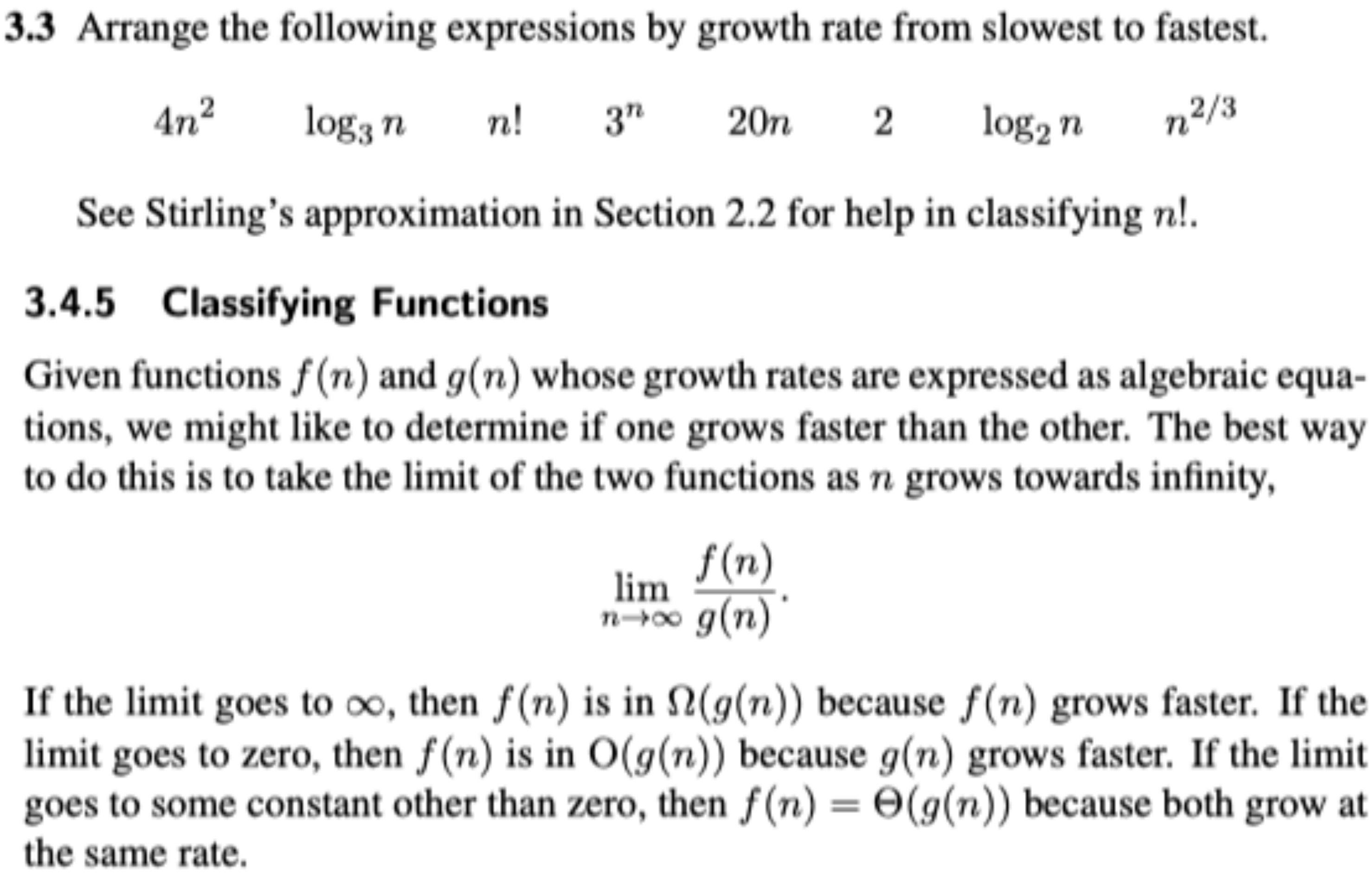
Course Summary

Question Summaries

- Questions
  - Growth rate
  - Regrading

Question

- Questions
  - Growth rate



3.3 Arrange the following expressions by growth rate from slowest to fastest.

$$4n^2 \qquad \log_3 n \qquad n! \qquad 3^n \qquad 20n \qquad 2 \qquad \log_2 n \qquad n^{2/3}$$

See Stirling's approximation in Section 2.2 for help in classifying $n!$.

3.4.5 Classifying Functions

Given functions $f(n)$ and $g(n)$ whose growth rates are expressed as algebraic equations, we might like to determine if one grows faster than the other. The best way to do this is to take the limit of the two functions as n grows towards infinity,

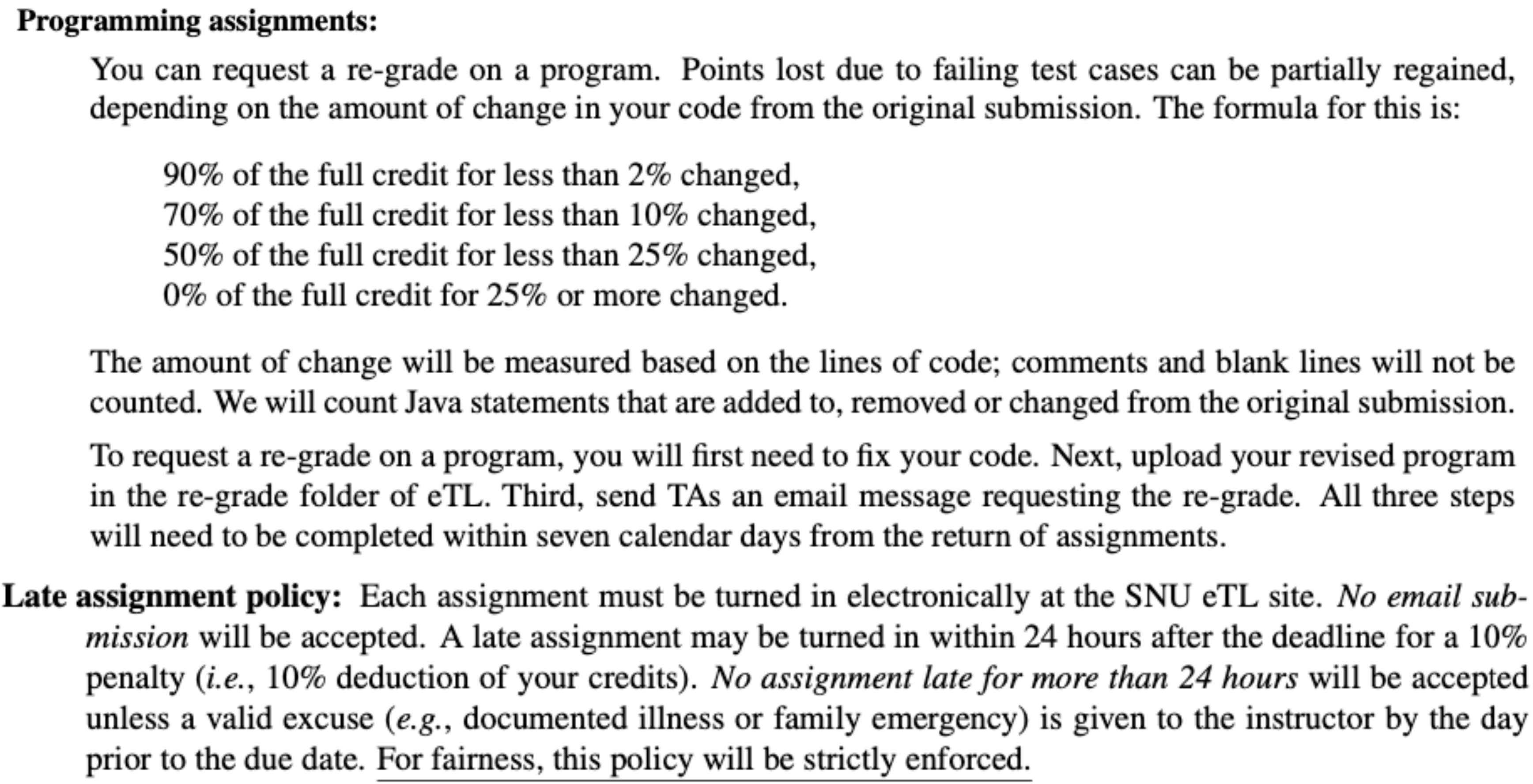
$$\lim_{n \to \infty} \frac{f(n)}{g(n)}.$$

If the limit goes to ∞, then $f(n)$ is in $\Omega(g(n))$ because $f(n)$ grows faster. If the limit goes to zero, then $f(n)$ is in $O(g(n))$ because $g(n)$ grows faster. If the limit goes to some constant other than zero, then $f(n) = \Theta(g(n))$ because both grow at the same rate.

From: Shaffer, Clifford A. *Data Structures and Algorithm Analysis*, n.d.

Question

- Questions
  - Regrading



Programming assignments:

You can request a re-grade on a program. Points lost due to failing test cases can be partially regained, depending on the amount of change in your code from the original submission. The formula for this is:

90% of the full credit for less than 2% changed,
70% of the full credit for less than 10% changed,
50% of the full credit for less than 25% changed,
0% of the full credit for 25% or more changed.

The amount of change will be measured based on the lines of code; comments and blank lines will not be counted. We will count Java statements that are added to, removed or changed from the original submission.

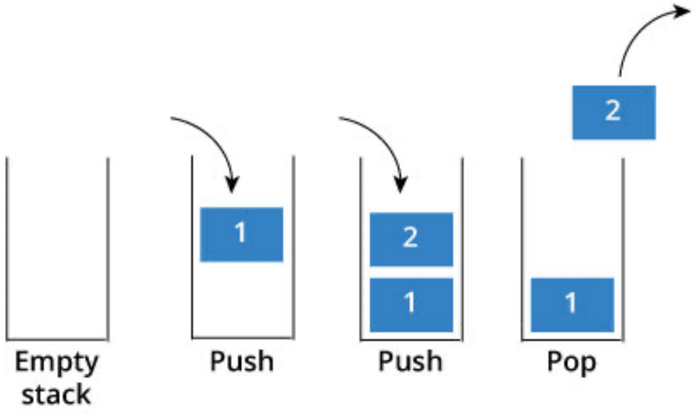
To request a re-grade on a program, you will first need to fix your code. Next, upload your revised program in the re-grade folder of eTL. Third, send TAs an email message requesting the re-grade. All three steps will need to be completed within seven calendar days from the return of assignments.

Late assignment policy: Each assignment must be turned in electronically at the SNU eTL site. *No email submission* will be accepted. A late assignment may be turned in within 24 hours after the deadline for a 10% penalty (*i.e.*, 10% deduction of your credits). *No assignment late for more than 24 hours* will be accepted unless a valid excuse (*e.g.*, documented illness or family emergency) is given to the instructor by the day prior to the due date. For fairness, this policy will be strictly enforced.

From: Bongki Moon, "Data Structures", Retrieved from syl204s20.pdf

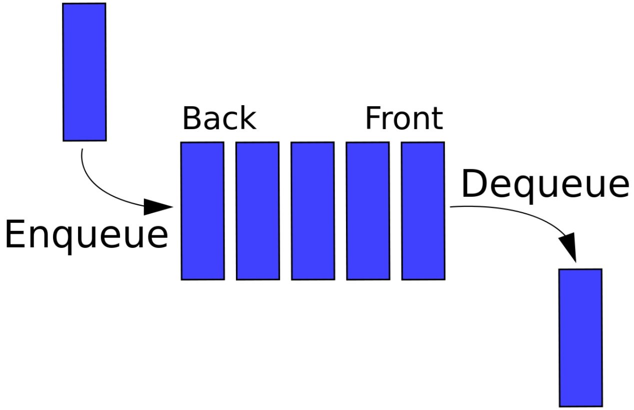
Stack

- Stack First in Last out (FILO)
  - `AStack()                  // CreateStack(), Constructor`
  - `isEmpty()`
  - `isFull()`
  - **`push()`**
  - **`pop()`**



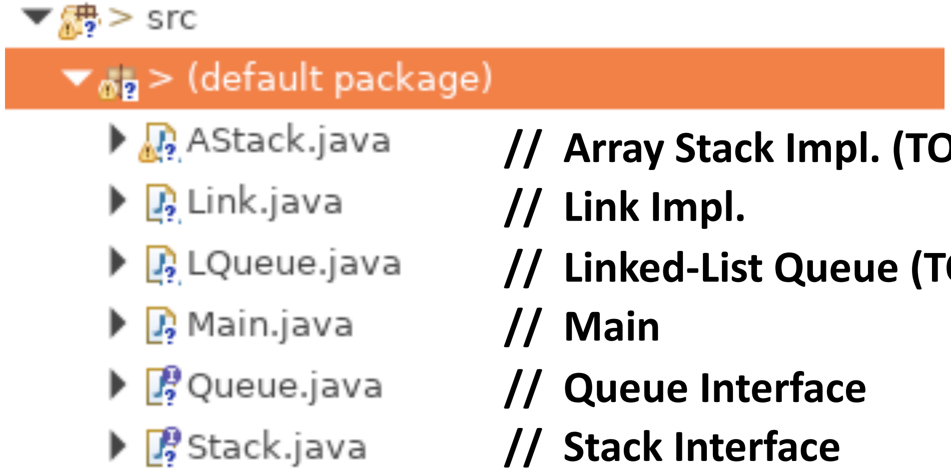
Queue

- Queue: First in First out (FIFO)
  - `LQueue()                // CreateQueue(), Constructor`
  - `isEmpty()`
  - `isFull()`
  - **`enqueue()`**
  - **`dequeue()`**



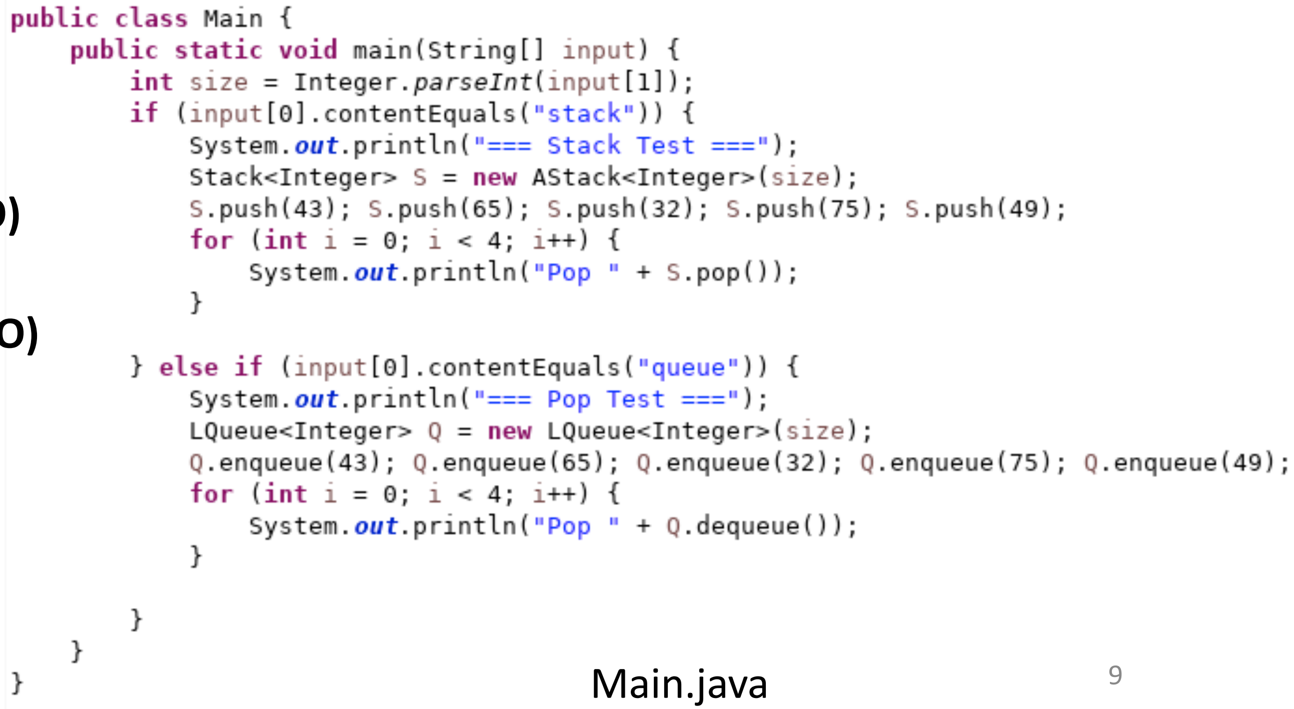
Exercises

- Fill the blank of codes
  - Write your code into "// TODO:" section (AStack.java, LQueue.java)
  - to implement Stack (using Array) and Queue (using Linked-List)



```
> src
  > (default package)
      AStack.java     // Array Stack Impl. (TODO)
      Link.java       // Link Impl.
      LQueue.java     // Linked-List Queue (TODO)
      Main.java       // Main
      Queue.java      // Queue Interface
      Stack.java      // Stack Interface
```

Project Structure



```
public class Main {
    public static void main(String[] input) {
        int size = Integer.parseInt(input[1]);
        if (input[0].contentEquals("stack")) {
            System.out.println("=== Stack Test ===");
            Stack<Integer> S = new AStack<Integer>(size);
            S.push(43); S.push(65); S.push(32); S.push(75); S.push(49);
            for (int i = 0; i < 4; i++) {
                System.out.println("Pop " + S.pop());
            }

        } else if (input[0].contentEquals("queue")) {
            System.out.println("=== Pop Test ===");
            LQueue<Integer> Q = new LQueue<Integer>(size);
            Q.enqueue(43); Q.enqueue(65); Q.enqueue(32); Q.enqueue(75); Q.enqueue(49);
            for (int i = 0; i < 4; i++) {
                System.out.println("Pop " + Q.dequeue());
            }

        }
    }
}
```

Main.java

Exercises

- Result

```
$ java Main stack 3
=== Stack Test ===
Stack is full
Stack is full
Pop 32
Pop 65
Pop 43
Stack is empty
```

```
$ java Main stack 6
=== Stack Test ===
Pop 49
Pop 75
Pop 32
Pop 65
```

```
$ java Main queue 2
=== Queue Test ===
Queue is full
Queue is full
Queue is full
Dequeue 43
Dequeue 65
Queue is empty
Queue is empty
```

```
$ java Main queue 6
=== Queue Test ===
Dequeue 43
Dequeue 65
Dequeue 32
Dequeue 75
```