Seoul National University

Data Structure

Spring 2025, Kang

Programming Assignment 1: Lists, Stacks, and Queues (Chapter 4)

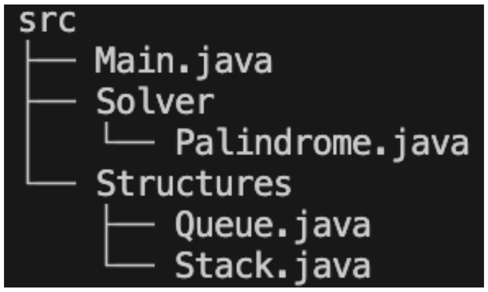
Due: April 15, 23:59, submit at eTL

Reminders

- The points of this homework add up to 100.
- Like all homework, this has to be done individually.
- Lead T.A.: Minjun Kim (snuds.ta@gmail.com)
- Write a program in Java.
- Do not use Java Collection Framework and third-party implementation from the Internet.

1. How to submit the programming assignment

1) Fill in the skeleton code with your own answer.
2) Compress your codes as 'src.zip' file. The structure of your 'src.zip' file should look like follows:



```
src
├── Main.java
├── Solver
│   └── Palindrome.java
└── Structures
    ├── Queue.java
    └── Stack.java
```

3) Your code will be compiled and executed with following commands in Linux environment. Make sure that your code runs well with following commands.

```
javac $(find src/* | grep .java)
java ./src/Main ${input_filepath} ${output_filepath}
```

4) Submit the zip file to the eTL (http://etl.snu.ac.kr/).

2. How to grade your programming assignment

1) We made a grading machine to automatically grade your programming assignment. The machine will run your program and compare the answers and outputs that your program generates for given inputs. If your program cannot generate correct answers for an input file, it will not give you the point corresponding to the input. Our machine will consider the following scenarios:

   (***Accept***) When your program generates exact outputs for an input file, the machine will give you the point of the input.

   (***Wrong Answer***) When your program runs normally but generates incorrect outputs for an input file, including typos, the machine will not give you the point of the input.

   (***Run Error***) When your program does not run or is terminated suddenly for some reason, the machine will not give you the point of an input file because it cannot generate any outputs.

   (***Time Limit***) When your program runs over a predefined execution time for an input file, our machine will stop your program, and it will not give you the point of the input. The time limit of the execution is ***5 seconds***.

2) We will generate 10 input files, and assign 10 points for each input file. For example, if your program gets 9 accepts, and 1 wrong answer by the machine, the total point will be 90 points. <u>Hence, before submitting your programming assignment, please be sure that your program makes correct answers in a reasonable time for any input case.</u>

3. Problem

Stack and **Queue** are simple but very powerful data structures when you make applications. You will implement these two data structures and solve problems using the implemented classes.

1) **Stack**

   A stack is an abstract data type that serves as a collection of elements, with the following main principal operations.

   - *Push*: Adds an element to the collection.
   - *Pop*: Removes the most recently added element.
   - *Clear*: Clear the stack.
   - *Length*: Return the length of the queue.
   - *isEmpty*: Determine if the stack is empty.

2) **Queue**

   A queue is a collection of entities that are maintained in a sequence and can be modified by the addition of entities at one end of the sequence and the removal of entities from the other end of the sequence, with the following main principal operations.

   - *Enqueue*: Adds an element to the collection.
   - *Dequeue*: Removes the oldest element in the collection.
   - *Clear*: Clear the queue.
   - *Length*: Return the length of the queue.
   - *isEmpty*: Determine if the queue is empty.

3) **PalindromeCheck**

   A palindrome is a sequence of numbers that reads the same backwards as forwards, such as 12421. With a given sequence of integers, each non-zero integer should be appended from the front one by one, and whenever a 0 is encountered, the sequence should be checked to see if it is a palindrome. Implement the following functions using the implemented data structures in Questions 3.1 and 3.2:

   - *newInt*: Adds a new integer from the sequence.
   - *clear*: Clear all required data structures.
   - *length:* Length of currently processed sequence.
   - *palindromeCheck:* Determine whether current sequence is a palindrome or not.

4. Important Notes

1) You **should not modify the Main.java** file in the skeleton. We will overwrite the Main.java file with our file during evaluation.

2) You **should not import any other packages or modules** other than already imported ones in the skeleton code. You will receive a severe penalty if you import other packages such as java.util.LinkedList or java.util.Stack.

3) The number of maximum entries of the stack or the queue will be given as an argument of the constructors of each class. For example, if we initialize a stack with a statement "`stack A = new stack(5);`", A is able to handle maximum five entries. If a data structure is full (i.e. it has reached its maximum capacity), any additional input will cause an overflow and will not be inserted. Note that the **maximum number of entries is set to 5 for all example / test cases.**

4) **You do not have to take account for detailed format of inputs and outputs in this assignment.** The Main class in skeleton would handle the i/o process of this assignment instead of you.

5. I/O Examples

Inputs and outputs of the Main.java file will be given and generated as a form of text files. Below examples are examples of sample inputs and outputs. You may test your code with the given sample files in the skeleton or your own inputs with following command after compile:

```
java ./src/Main ${input_filepath} ${output_filepath}
```

Sample Input	Sample Output
stack push 10 stack push 20 stack push 30 stack pop stack isempty stack push 40 stack length stack pop stack pop stack pop stack isempty stack push 50 stack clear stack isempty stack length	stack push: 10 stack push: 20 stack push: 30 stack pop: 30 stack is not empty stack push: 40 stack length: 3 stack pop: 40 stack pop: 20 stack pop: 10 stack is empty stack push: 50 stack cleared stack is empty stack length: 0

Sample Input	Sample Output
queue enqueue 20 queue enqueue 30 queue dequeue queue isempty queue enqueue 40 queue length queue dequeue queue dequeue queue isempty queue length queue enqueue 50 queue clear queue isempty queue length	queue enqueue: 20 queue enqueue: 30 queue dequeue: 20 queue is not empty queue enqueue: 40 queue length: 2 queue dequeue: 30 queue dequeue: 40 queue is empty queue length: 0 queue enqueue: 50 queue cleared queue is empty queue length: 0

Sample Input	**Sample Output**
palindrome 1 1 1 1 0	palindrome: new sequence
palindrome 2 3 5 3 2 0	palindrome newInt: 1
palindrome 2 3 5 5 3 2 0	palindrome newInt: 1
palindrome 2 2 0 3 3 2 2 0	palindrome newInt: 1
	palindrome newInt: 1
	palindrome Check: true 4
	palindrome: new sequence
	palindrome newInt: 2
	palindrome newInt: 3
	palindrome newInt: 5
	palindrome newInt: 3
	palindrome newInt: 2
	palindrome Check: true 5
	palindrome: new sequence
	palindrome newInt: 2
	palindrome newInt: 3
	palindrome newInt: 5
	palindrome newInt: 5
	palindrome newInt: 3
	palindrome newInt: 2
	palindrome Check: false 5
	palindrome: new sequence
	palindrome newInt: 2
	palindrome newInt: 2
	palindrome Check: true 2
	palindrome newInt: 3
	palindrome newInt: 3
	palindrome newInt: 2
	palindrome newInt: 2
	palindrome Check: false 5