Stack - Assignment 1

Problem Statement

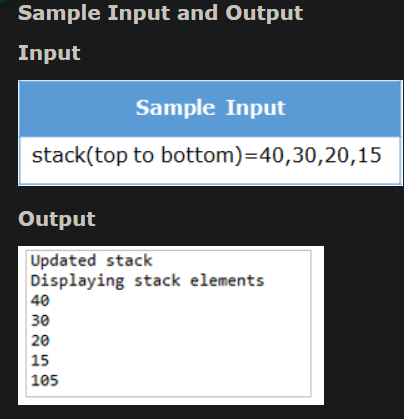
Given a stack of integers, calculate the sum of all the integers present in the stack. Modify the stack such that the sum is present in the bottom of the stack and all the other integers are present in the stack in the same order. Implement the logic inside calculateSum() method of the Tester class.

The Stack class has been provided.

Test the functionalities using the main() method of the Tester class.

Sample Input and Output

Input



public class Tester {  
 public static void main(String args[]) {  
  
 Stack stack = new Stack(10);  
 stack.push(15);  
 stack.push(20);  
 stack.push(30);  
 stack.push(40);  
  
 *calculateSum*(stack);  
  
 System.*out*.println("Updated stack");  
 stack.display();  
 }  
  
 public static void calculateSum(Stack stack) {  
 //Implement your code here  
 Stack tempStack = new Stack(12);  
 int a = 0;  
 while (!stack.isEmpty()) {  
 a += stack.peek();  
 tempStack.push(stack.pop());  
 }  
 stack.push(a);  
 while (!tempStack.isEmpty()) {  
  
 stack.push(tempStack.pop());  
 }  
  
 }  
}

—————————————————————————————————————————————————

Queue - Assignment 1

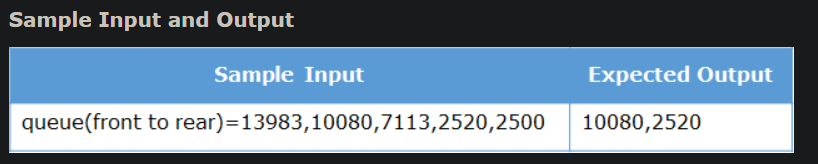
Problem Statement

Given a queue of whole numbers, find the numbers in the queue that are evenly divisible by all the numbers from 1 to 10 and return a new queue with all the evenly divisible numbers. Implement the logic inside findEvenlyDivisibleNumbers() method of the Tester class.

The Queue class has been provided.

Test the functionalities using the main() method of the Tester class.

Sample Input and Output



class Tester {  
  
 public static void main(String[] args) {  
  
 Queue queue = new Queue(7);  
 queue.enqueue(13983);  
 queue.enqueue(10080);  
 queue.enqueue(7113);  
 queue.enqueue(2520);  
 queue.enqueue(2500);  
  
 Queue outputQueue = *findEvenlyDivisibleNumbers*(queue);  
  
 System.*out*.println("Evenly divisible numbers");  
 outputQueue.display();  
  
 }  
  
 public static Queue findEvenlyDivisibleNumbers(Queue queue) {  
 // Create a result queue to hold numbers divisible by 1 to 10  
 Queue resultQueue = new Queue(queue.getMaxSize());  
  
 // Create a temporary queue to restore original contents if needed  
 Queue tempQueue = new Queue(queue.getMaxSize());  
  
 // Dequeue each element, check divisibility, and enqueue to result if valid  
 while(!queue.isEmpty()) {  
 int current = queue.dequeue();  
 if(*isDivisibleBy1To10*(current)) {  
 resultQueue.enqueue(current);  
 }  
  
 }  
  
 // Return the queue containing numbers divisible by 1..10  
 return resultQueue;  
 }  
  
 // Helper method to check if a number is divisible by all integers 1..10  
 public static boolean isDivisibleBy1To10(int number) {  
 for(int i = 1; i <= 10; i++) {  
 if(number % i != 0) {  
 return false;  
 }  
 }  
 return true;  
 }  
}