

Stack - Lặp vòng

Đoạn code phía dưới là thiết kế chi tiết lớp `ArrayStack<T>`, và phương thức `main` để kiểm tra. Em hãy hiện thực lớp này với gợi ý sau:

`ArrayStack<T>` là kiểu dữ liệu Generic, cho phép lưu trữ dữ liệu kiểu số (thừa kế từ `Number`). Như tên đã thể hiện, nó lưu trữ dữ liệu bên trong bằng mảng có kích thước cố định. Hỗ trợ tối thiểu các phương thức thức như trong thiết kế.

Kích thước mảng lưu trữ là cố định, khi số phần tử đưa vào stack vượt quá số kích thước mảng thì loại bỏ phần tử đưa vào sớm nhất (First in)

Input

+ Dòng đầu tiên gồm ba số n ($1 \leq n \leq 2 \cdot 10^5$) là kích thước mảng lưu trữ - tham khảo thiết kế, m ($0 \leq m \leq 2 \cdot 10^5$) là số phần tử khởi tạo cho stack, k ($0 \leq k \leq 2 \cdot 10^5$) số câu lệnh cần thực hiện

+ Dòng tiếp theo chứa m phần tử được khởi tạo cho stack (m có thể lớn hơn n)

+ k dòng tiếp theo chứa các câu lệnh theo dạng: "command [parameters]". Command là tên method, parameters là tham số tương ứng với file thiết kế

Output

+ Với các command có trả ra giá trị (sum, average, peek, pop), xuất kết quả vào standard output

Example

Input:

```
10 5 10
683 376 -871 323 -98
peek
push 619
push 432
peek
push 5
push 275
pop
push 664
push -810
```

Output:

```
-98
432
275
```

Program

```
import java.io.*;
import java.util.*;
```

```

class ArrayStack<T extends Number> {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int m = sc.nextInt();

        ArrayStack<Integer> stack = new ArrayStack<Integer>(n);
        // You code here
    }

    // Your code here
    Object[] data;
    int lastIndex = 0;

    public ArrayStack(int capacity) {
        data = new Object[capacity];
    }

    public int count() {
        // Your code here
        return 0;
    }

    public double sum() {
        // Your code here
        // T number;
        // number.doubleValue();
        return 0;
    }

    public double average() {
        // Your code here
        return 0;
    }

    /**
     * @description: When stack is full => override the earliest added item
     */
    public void push(T item) {
        // Sample: data[lastIndex++] = item;
        // Your code here
    }

    /**
     * @return: return and remove top item, or null when stack is empty
     */
    public T pop() {
        // Your code here
        return null;
    }

    /**
     * @return: return top item, or null when stack is empty
     */
    @SuppressWarnings("unchecked")
    public T peek() {
        // Sample return (T) data[lastIndex - 1];
        // Your code here
        return null;
    }
}

```

```
}  
}
```

Gợi ý testcases:

Trên 30% là testcases đơn giản chỉ kiểm tra các tác vụ riêng lẻ. Các em có thể submit từng phần để kiểm tra xem đúng được những tác vụ nào:

1. Testcases 1-13: chỉ bao gồm các tác vụ đơn lẻ (peek, pop, average, sum - PPES)
2. Testcase 12-20: kết hợp các tác vụ, nhưng đảm bảo số phần tử thêm vào không vượt quá kích thước mảng lưu trữ
3. Testcase 21-30: kết hợp các tác vụ, số phần tử thêm vào vượt quá kích thước mảng lưu trữ
4. Testcase 31-35: kết hợp các tác vụ, số phần tử thêm vào và số câu lệnh lớn hơn 10^5