Ahmad Siddiq Priaji (22/496854/PA/21370)

Homework 7

1. Tugas nomor 1 dan 2
2. **TestReverseString.java**

import java.util.\*;

public class TestReverseString {

    public static void main(String[] args) {

        String kata;

        Stack<Character> stack = new Stack<>();

        Scanner input = new Scanner(System.in);

        System.out.println("Masukkan input: ");

        kata = input.next();

        for (int i = 0; i < kata.length(); i++) {

            char c = kata.charAt(i);

            stack.push(c);

        }

        System.out.println("Hasil reverse: ");

        while (!stack.isEmpty()) {

            char c = stack.pop();

            System.out.print(c);

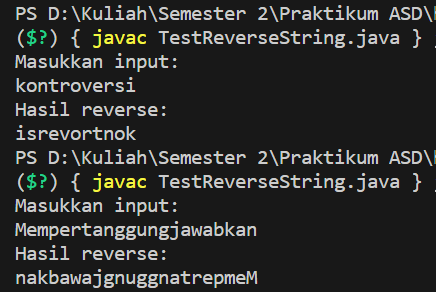
        }

        input.close();

    }

}

**Hasil:**

****

1. **TestPalindrome.java**

import java.util.\*;

public class TestPalindrome {

    public static void main(String[] args) {

        String kata;

        Stack<Character> stack = new Stack<>();

        Scanner input = new Scanner(System.in);

        System.out.println("Masukkan input: ");

        kata = input.next();

        for (int i = 0; i < kata.length(); i++) {

            char c = kata.charAt(i);

            stack.push(c);

        }

        boolean isPalindrom = false;

        for (int i = 0; i < kata.length(); i++) {

            if (kata.charAt(i) == stack.pop()) {

                isPalindrom = true;

            }

            else {

                isPalindrom = false;

                break;

            }

        }

        System.out.println("Merupakan palindrom: ");

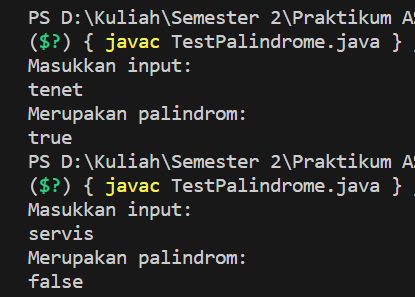
        System.out.println(isPalindrom);

        input.close();

    }

}

**Hasil:**

****

1. Sharing Stack

**SharingStack.java**

public class SharingStack {

    int maxSize;

    int[] stackArray; // membuat satu array

    int topA;

    int topB;

    boolean isFull;

    public SharingStack(int s) {

        maxSize = s;

        stackArray = new int[maxSize];

        topA = -1; // indeks awal stack A

        topB = maxSize; // indeks awal stack B

        isFull = false;

    }

    public void pushA(int j) { // menaruh data stack A dari depan array

        if (topA < topB-1){

            stackArray[++topA] = j;

        }

        else {

            isFull = true;

        }

    }

    public void pushB(int j) {  // menaruh data stack B dari belakang array

        if (topA < topB-1) {

            stackArray[--topB] = j;

        }

        else {

            isFull = true;

        }

    }

    public int popA() {

        int temp = stackArray[topA--];

        if (topA < topB-1) {

            isFull = false;

        }

        if (topA == -1) {

            System.out.println("Stack A is empty");

            return -1;

        }

        return temp;

    }

    public int popB() {

        int temp = stackArray[topB++];

        if (topA < topB-1) {

            isFull = false;

        }

        if (topB == maxSize) {

            System.out.println("Stack B is empty");

            return -1;

        }

        return temp;

    }

    public void printStack() {

        if (isFull) {

            System.out.println("Sharing stack is full");

        }

        System.out.println("Stack A: ");

        for (int i = 0; i <= topA; i++) {

            System.out.print( stackArray[i] + " ");

        }

        System.out.println();

        System.out.println("Stack B: ");

        for (int i = stackArray.length-1; i >= topB; i--) {

            System.out.print( stackArray[i] + " ");

        }

        System.out.println();

    }

    public static void main(String[] args) {

        SharingStack theStack = new SharingStack(10);

        theStack.pushA(7);

        theStack.pushA(31);

        theStack.pushA(19);

        theStack.pushA(3);

        theStack.pushB(4);

        theStack.pushB(10);

        theStack.pushB(42);

        theStack.pushB(30);

        theStack.pushB(66);

        theStack.pushB(120);

        theStack.pushB(200); // stack penuh, tidak tersimpan

        theStack.pushB(72); // stack penuh, tidak tersimpan

        theStack.printStack();

        System.out.println();

        System.out.println("Pop A: " + theStack.popA());

        System.out.println("Pop B: " + theStack.popB());

        theStack.pushB(202);

        theStack.pushB(34);

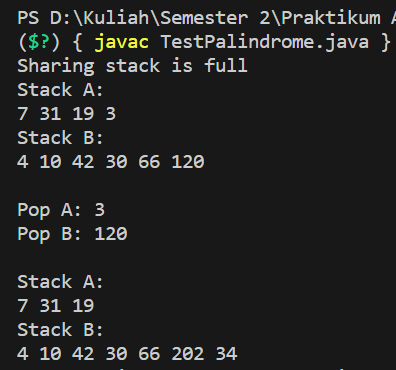
        System.out.println();

        theStack.printStack();

    }

}

**Hasil:**



1. New Queue

**NewQueue.java**

public class NewQueue {

    int maxSize;

    int[] queueArray;

    int front;

    int rear;

    int nItems;

    public NewQueue(int s) {

        this.maxSize = s;

        this.queueArray = new int[maxSize];

        this.front = 0;

        this.rear = -1;

        this.nItems = 0;

    }

    public void enqueue(int j) {

        if (rear == maxSize -1) {

            rear = -1;

        }

        queueArray[++rear] = j;

        nItems++;

    }

    public int dequeueFront() { //dequeue dari depan

        int temp = queueArray[front++];

        if (front == maxSize) {

            front = 0;

        }

        nItems--;

        return temp;

    }

    public int dequeueRear() { //dequeue dari belakang

        int temp = queueArray[rear--];

        if (rear == -1) {

            rear = maxSize -1;

        }

        nItems--;

        return temp;

    }

    public boolean isEmpty() { //true if queue is empty

        return (nItems == 0);

    }

    public boolean isFull() { //true jika nItems melebihi maxSize

        return (nItems > maxSize);

    }

    public static void main(String[] args) {

        NewQueue theQueue = new NewQueue(5);

        theQueue.enqueue(1);

        theQueue.enqueue(2);

        theQueue.enqueue(3);

        theQueue.enqueue(4);

        theQueue.enqueue(5);

        // queue awal: [1, 2, 3, 4, 5]

        System.out.println("Dequeue front: "+theQueue.dequeueFront());

        System.out.println("Dequeue rear: "+theQueue.dequeueRear());

        System.out.println("Dequeue front: "+theQueue.dequeueFront());

        theQueue.enqueue(7);

        theQueue.enqueue(8);

        theQueue.enqueue(9);

        System.out.println("Finnal queue : ");

        if (!theQueue.isFull()) { //mengeluarkan semua data queue

            while (!theQueue.isEmpty()) {

                System.out.print(theQueue.dequeueFront() + " ");

            }

            System.out.println();

        }

        else {

            System.out.println("Queue is full");

        }

    }

}

**Hasil:**

