Practice Problem 1. Undo Feature in a Text Editor (Stack)

import java.util.\*;

public class TextEditorUndo {

public static void main(String[] args) {

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

Scanner sc = new Scanner(System.in);

System.out.println("Text Editor with Undo Feature");

System.out.println("Commands: TYPE <word>, UNDO, PRINT, EXIT\n");

while (true) {

System.out.print("Enter command: ");

String cmd = sc.next();

if (cmd.equals("TYPE")) {

String word = sc.next();

stack.push(word);

System.out.println("Added: " + word);

}

else if (cmd.equals("UNDO")) {

if (!stack.isEmpty()) {

String removed = stack.pop();

System.out.println("Removed: " + removed);

} else {

System.out.println("Nothing to undo");

}

}

else if (cmd.equals("PRINT")) {

if (stack.isEmpty()) {

System.out.println("Empty");

} else {

for (int i = 0; i < stack.size(); i++) {

System.out.print(stack.get(i) + " ");

}

System.out.println();

}

}

else if (cmd.equals("EXIT")) {

System.out.println("Exiting...");

break;

}

else {

System.out.println("Invalid command");

}

}

sc.close();

}

}

Practice Problem 2. Browser Navigation Simulation (Stack)

import java.util.\*;

public class BrowserNavigation {

public static void main(String[] args) {

Stack<String> backStack = new Stack<>();

Stack<String> forwardStack = new Stack<>();

Scanner sc = new Scanner(System.in);

String current = "Home";

System.out.println("Browser Navigation Simulator");

System.out.println("Commands: VISIT <url>, BACK, FORWARD, PRINT, EXIT\n");

System.out.println("Current Page: " + current + "\n");

while (true) {

System.out.print("Command: ");

String cmd = sc.next();

if (cmd.equals("VISIT")) {

String url = sc.next();

backStack.push(current);

current = url;

forwardStack.clear();

System.out.println("Visited: " + current);

}

else if (cmd.equals("BACK")) {

if (!backStack.isEmpty()) {

forwardStack.push(current);

current = backStack.pop();

System.out.println("Back to: " + current);

} else {

System.out.println("No previous page");

}

}

else if (cmd.equals("FORWARD")) {

if (!forwardStack.isEmpty()) {

backStack.push(current);

current = forwardStack.pop();

System.out.println("Forward to: " + current);

} else {

System.out.println("No forward page");

}

}

else if (cmd.equals("PRINT")) {

System.out.println("Current Page: " + current);

}

else if (cmd.equals("EXIT")) {

System.out.println("Closing browser...");

break;

}

else {

System.out.println("Invalid command");

}

}

sc.close();

}

}

Practice Problem 3. Print Queue System for Office Printer (Queue)

import java.util.\*;

public class PrintQueueSystem {

public static void main(String[] args) {

Queue<String> printQueue = new LinkedList<>();

Scanner sc = new Scanner(System.in);

System.out.println("Office Print Queue System");

System.out.println("Commands: ADD <document>, PRINT, EXIT\n");

while (true) {

System.out.print("Command: ");

String cmd = sc.next();

if (cmd.equals("ADD")) {

String document = sc.next();

printQueue.add(document);

System.out.println("Added to queue: " + document);

System.out.println("Queue size: " + printQueue.size());

}

else if (cmd.equals("PRINT")) {

if (!printQueue.isEmpty()) {

String doc = printQueue.remove();

System.out.println("Printing " + doc);

} else {

System.out.println("No jobs left!");

}

}

else if (cmd.equals("EXIT")) {

System.out.println("Shutting down printer...");

break;

}

else {

System.out.println("Invalid command");

}

}

sc.close();

}

}

Practice Problem 4. Customer Service Counter Simulation (Queue)

import java.util.\*;

public class CustomerServiceCounter {

public static void main(String[] args) {

Queue<String> queue = new LinkedList<>();

Scanner sc = new Scanner(System.in);

System.out.println("Customer Service Counter Simulation");

System.out.println("Commands: ARRIVE <name>, SERVE, STATUS, EXIT\n");

while (true) {

System.out.print("Command: ");

String cmd = sc.next();

if (cmd.equals("ARRIVE")) {

String name = sc.next();

queue.add(name);

System.out.println(name + " joined the queue");

System.out.println("Queue length: " + queue.size());

}

else if (cmd.equals("SERVE")) {

if (!queue.isEmpty()) {

String customer = queue.remove();

System.out.println("Serving " + customer);

} else {

System.out.println("No customers waiting");

}

}

else if (cmd.equals("STATUS")) {

if (queue.isEmpty()) {

System.out.println("Waiting: []");

} else {

System.out.println("Waiting: " + queue);

}

}

else if (cmd.equals("EXIT")) {

System.out.println("Closing counter...");

break;

}

else {

System.out.println("Invalid command");

}

}

sc.close();

}

}

Practice Problem 5. Expression Validator for Calculator App (Stack)

import java.util.\*;

public class ExpressionValidator {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Expression Validator - Parenthesis Balance Checker");

System.out.print("Enter expression: ");

String exp = sc.nextLine();

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

boolean isBalanced = true;

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

char ch = exp.charAt(i);

if (ch == '(' || ch == '{' || ch == '[') {

stack.push(ch);

}

else if (ch == ')' || ch == '}' || ch == ']') {

if (stack.isEmpty()) {

isBalanced = false;

break;

}

char top = stack.pop();

if (ch == ')' && top != '(') {

isBalanced = false;

break;

}

if (ch == '}' && top != '{') {

isBalanced = false;

break;

}

if (ch == ']' && top != '[') {

isBalanced = false;

break;

}

}

}

if (!stack.isEmpty()) {

isBalanced = false;

}

if (isBalanced) {

System.out.println("Expression is Balanced");

} else {

System.out.println("Expression is NOT Balanced");

}

sc.close();

}

}

Practice Problem 6. Hospital Emergency Room (Priority Queue Concept)

import java.util.\*;

class Patient {

String name;

int priority;

Patient(String n, int p) {

name = n;

priority = p;

}

}

public class EmergencyRoom {

public static void main(String[] args) {

PriorityQueue<Patient> pq = new PriorityQueue<>(Comparator.comparingInt(p -> -p.priority));

Scanner sc = new Scanner(System.in);

System.out.println("Hospital Emergency Room System");

System.out.println("Priority: 1=Low, 2=Medium, 3=High");

System.out.println("Commands: ARRIVE <name> <priority>, TREAT, STATUS, EXIT\n");

while (true) {

System.out.print("Command: ");

String cmd = sc.next();

if (cmd.equals("ARRIVE")) {

String name = sc.next();

int priority = sc.nextInt();

Patient patient = new Patient(name, priority);

pq.add(patient);

System.out.println(name + " arrived with priority " + priority);

}

else if (cmd.equals("TREAT")) {

if (!pq.isEmpty()) {

Patient patient = pq.poll();

System.out.println("Treating " + patient.name + " (Priority " + patient.priority + ")");

} else {

System.out.println("No patients waiting");

}

}

else if (cmd.equals("STATUS")) {

if (pq.isEmpty()) {

System.out.println("Waiting: []");

} else {

System.out.print("Waiting: [");

List<Patient> temp = new ArrayList<>(pq);

for (int i = 0; i < temp.size(); i++) {

System.out.print(temp.get(i).name);

if (i < temp.size() - 1) {

System.out.print(", ");

}

}

System.out.println("]");

}

}

else if (cmd.equals("EXIT")) {

System.out.println("Closing emergency room...");

break;

}

else {

System.out.println("Invalid command");

}

}

sc.close();

}

}