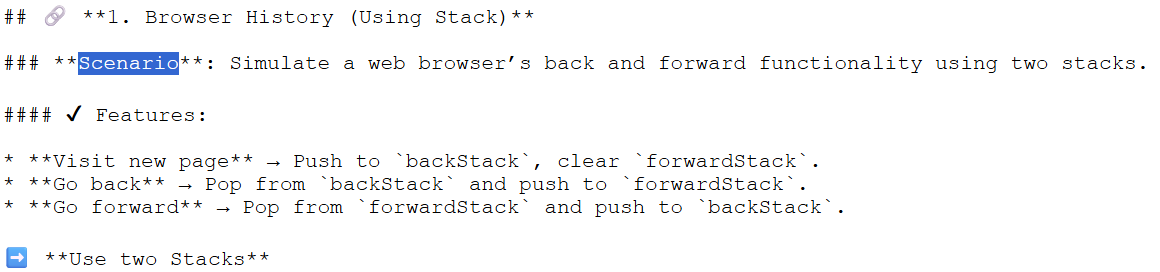
**Scenario Based Question:**

**RAKSHITHA J K**

**1.**



**CODE:**

import java.util.Scanner;

import java.util.Stack;

class BrowserHistory {

private Stack<String> backStack;

private Stack<String> forwardStack;

private String currentPage;

public BrowserHistory() {

backStack = new Stack<>();

forwardStack = new Stack<>();

currentPage = "Home"; // Starting point

}

public void visit(String url) {

if (currentPage != null) {

backStack.push(currentPage);

}

currentPage = url;

forwardStack.clear();

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

}

public void goBack() {

if (!backStack.isEmpty()) {

forwardStack.push(currentPage);

currentPage = backStack.pop();

System.out.println("Went back to: " + currentPage);

} else {

System.out.println("No back history.");

}

}

public void goForward() {

if (!forwardStack.isEmpty()) {

backStack.push(currentPage);

currentPage = forwardStack.pop();

System.out.println("Went forward to: " + currentPage);

} else {

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

}

}

public void showCurrentPage() {

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

}

}

// ✅ Entry point expected by OnlineGDB

public class Main {

public static void main(String[] args) {

BrowserHistory browser = new BrowserHistory();

Scanner sc = new Scanner(System.in);

int choice;

while (true) {

System.out.println("\n--- Browser Menu ---");

System.out.println("1. Visit new page");

System.out.println("2. Go back");

System.out.println("3. Go forward");

System.out.println("4. Show current page");

System.out.println("0. Exit");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

sc.nextLine(); // consume newline

switch (choice) {

case 1:

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

String url = sc.nextLine();

browser.visit(url);

break;

case 2:

browser.goBack();

break;

case 3:

browser.goForward();

break;

case 4:

browser.showCurrentPage();

break;

case 0:

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

return;

default:

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

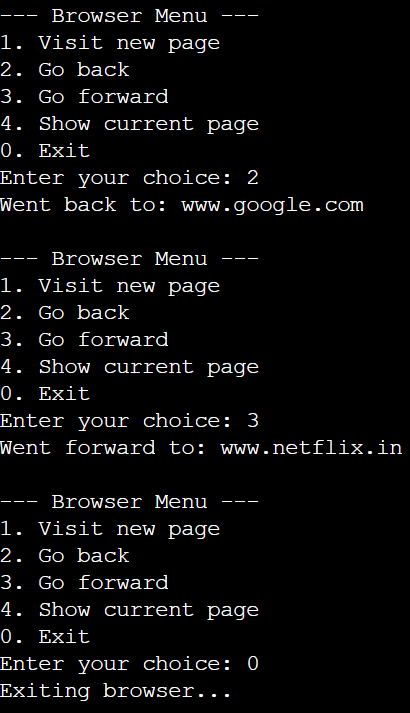
}

}

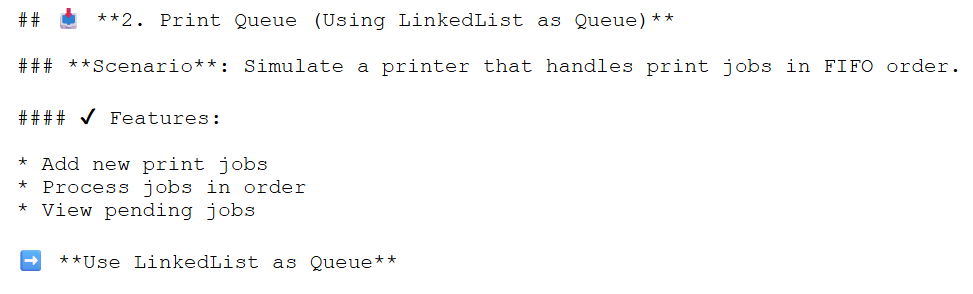
}

}

**OUTPUT:**

**2.**



**CODE:**

import java.util.LinkedList;

import java.util.Queue;

import java.util.Scanner;

class PrintQueue {

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

public void addJob(String job) {

queue.offer(job);

System.out.println("Added print job: " + job);

}

public void processJob() {

if (!queue.isEmpty()) {

System.out.println("Processing: " + queue.poll());

} else {

System.out.println("No jobs to process.");

}

}

public void viewJobs() {

if (!queue.isEmpty()) {

System.out.println("Pending jobs: " + queue);

} else {

System.out.println("No pending jobs.");

}

}

}

public class Main {

public static void main(String[] args) {

PrintQueue printer = new PrintQueue();

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- Print Queue Menu ---");

System.out.println("1. Add new print job");

System.out.println("2. Process next job");

System.out.println("3. View pending jobs");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

sc.nextLine(); // consume newline

switch (choice) {

case 1:

System.out.print("Enter job name: ");

String job = sc.nextLine();

printer.addJob(job);

break;

case 2:

printer.processJob();

break;

case 3:

printer.viewJobs();

break;

case 4:

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

break;

default:

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

}

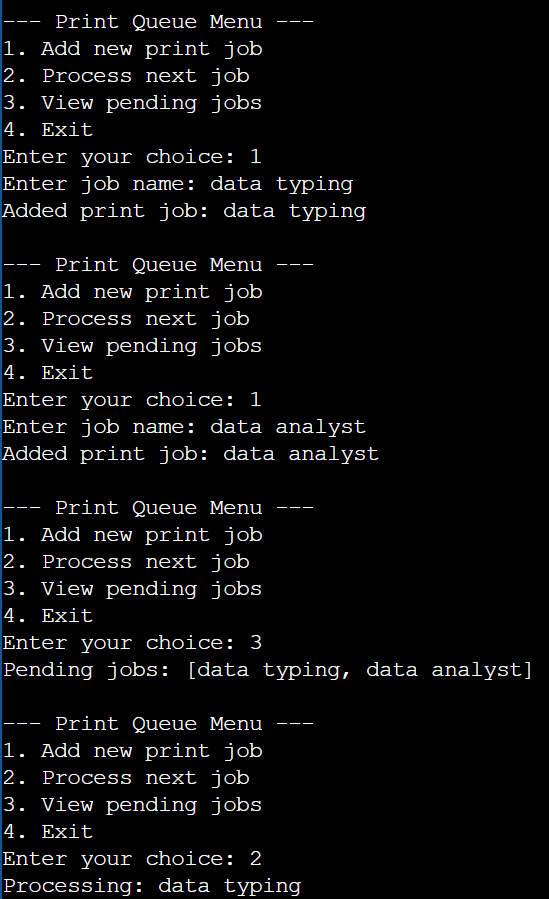
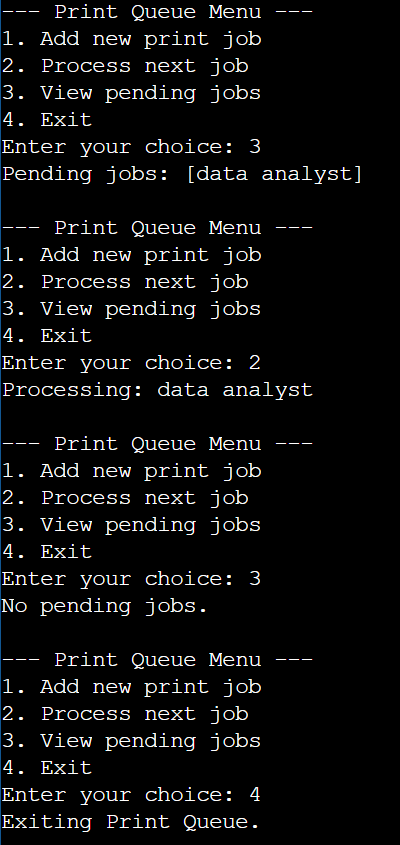
} while (choice != 4);

sc.close();

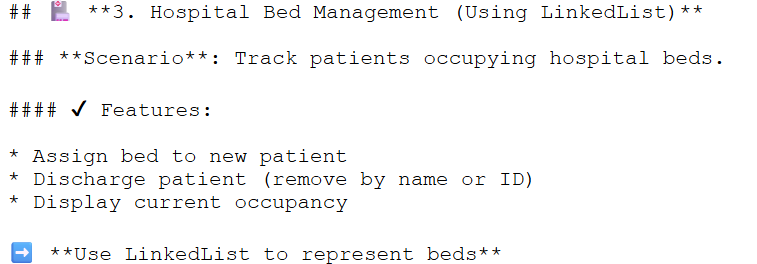
}

}

**OUTPUT:**

**3.**



**CODE:**

import java.util.LinkedList;

import java.util.Scanner;

class HospitalBedManagement {

private LinkedList<String> beds = new LinkedList<>();

public void assignBed(String patient) {

beds.add(patient);

System.out.println("Assigned bed to: " + patient);

}

public void discharge(String patient) {

if (beds.remove(patient)) {

System.out.println("Discharged: " + patient);

} else {

System.out.println("Patient not found.");

}

}

public void viewBeds() {

if (!beds.isEmpty()) {

System.out.println("Current Occupancy: " + beds);

} else {

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

}

}

}

public class Main {

public static void main(String[] args) {

HospitalBedManagement hospital = new HospitalBedManagement();

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- Hospital Bed Management Menu ---");

System.out.println("1. Assign bed to new patient");

System.out.println("2. Discharge patient");

System.out.println("3. View current occupancy");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.print("Enter patient name: ");

String patient = sc.nextLine();

hospital.assignBed(patient);

break;

case 2:

System.out.print("Enter patient name to discharge: ");

String name = sc.nextLine();

hospital.discharge(name);

break;

case 3:

hospital.viewBeds();

break;

case 4:

System.out.println("Exiting Hospital Bed Management.");

break;

default:

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

}

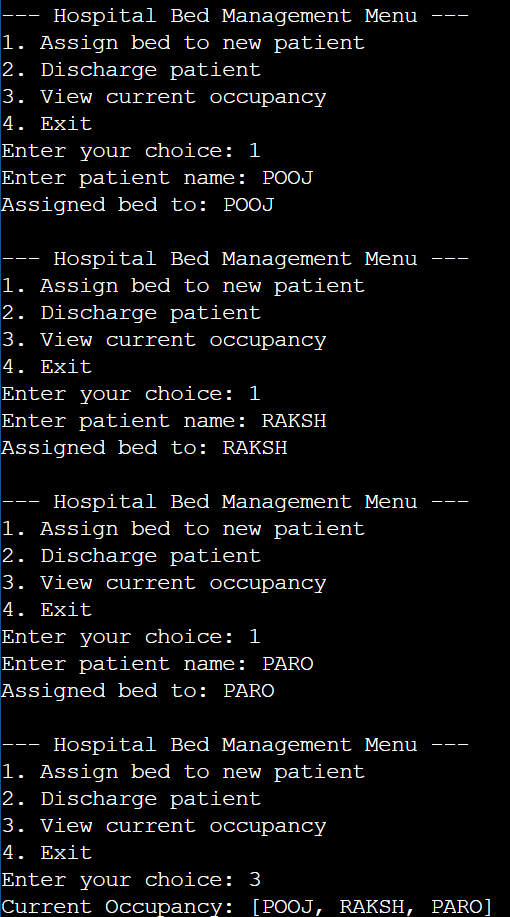
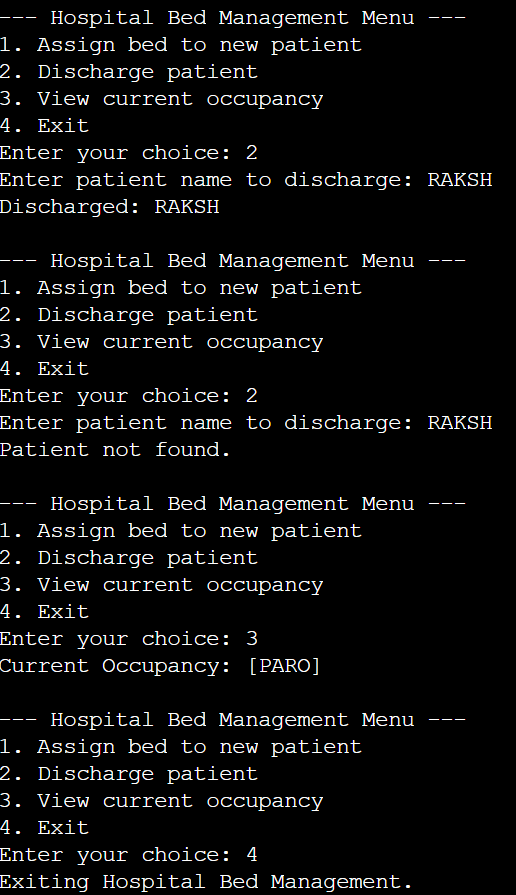
} while (choice != 4);

sc.close();

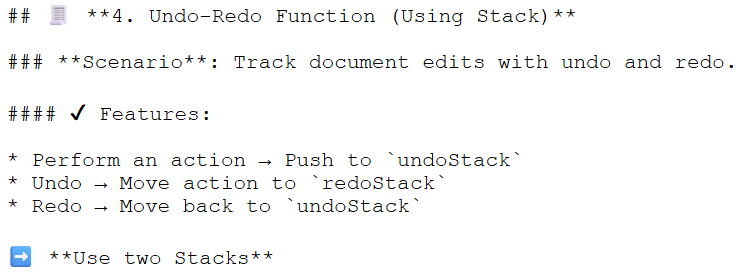
}

}

**OUTPUT:**

**4.**



**CODE:**

import java.util.\*;

class UndoRedo {

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

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

void doAction(String action) {

undo.push(action);

redo.clear();

System.out.println("Action performed: " + action);

}

void undoAction() {

if (!undo.isEmpty()) {

String undone = undo.pop();

redo.push(undone);

System.out.println("Undo: " + undone);

} else {

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

}

}

void redoAction() {

if (!redo.isEmpty()) {

String redone = redo.pop();

undo.push(redone);

System.out.println("Redo: " + redone);

} else {

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

}

}

void showState() {

System.out.println("Current actions: " + undo);

}

}

public class Main {

public static void main(String[] args) {

UndoRedo ur = new UndoRedo();

ur.doAction("Write");

ur.doAction("Delete");

ur.undoAction();

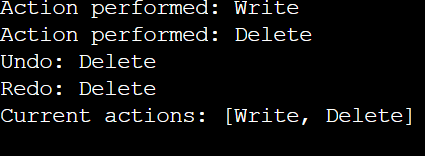
ur.redoAction();

ur.showState();

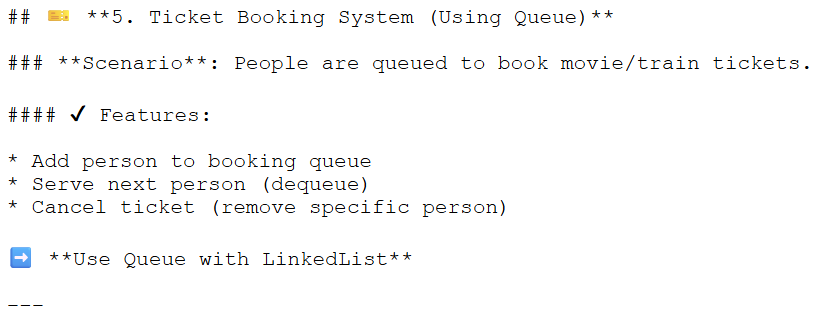
}

}

**OUTPUT:**



**5.**



**CODE:**

import java.util.\*;

class TicketBooking {

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

void book(String name) {

queue.offer(name);

System.out.println("Booked: " + name);

}

void serve() {

if (!queue.isEmpty()) {

System.out.println("Serving: " + queue.poll());

} else {

System.out.println("No one to serve.");

}

}

void cancel(String name) {

if (queue.remove(name)) {

System.out.println("Canceled booking for: " + name);

} else {

System.out.println("Name not found: " + name);

}

}

void viewQueue() {

if (queue.isEmpty()) {

System.out.println("No one is waiting.");

} else {

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

for (String p : queue) {

System.out.println("- " + p);

}

}

}

}

public class Main {

public static void main(String[] args) {

TicketBooking tb = new TicketBooking();

tb.book("Alice");

tb.book("Bob");

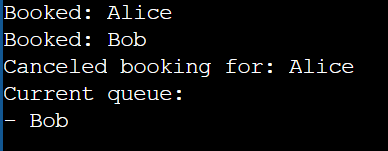
tb.cancel("Alice");

tb.viewQueue();

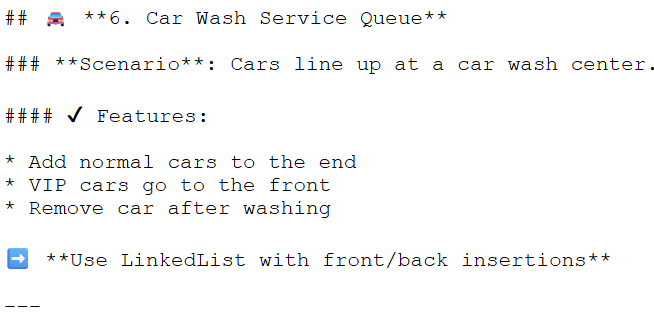
}

}

**OUTPUT:**



**6.**

****

**CODE:**

import java.util.\*;

class CarWash {

LinkedList<String> cars = new LinkedList<>();

void addNormal(String car) {

cars.addLast(car);

System.out.println("Normal car added: " + car);

}

void addVIP(String car) {

cars.addFirst(car);

System.out.println("VIP car added: " + car);

}

void wash() {

if (!cars.isEmpty()) {

String washed = cars.removeFirst();

System.out.println("Washing: " + washed);

} else {

System.out.println("No cars in the queue.");

}

}

void showQueue() {

if (cars.isEmpty()) {

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

} else {

System.out.println("Cars in queue:");

for (String car : cars) {

System.out.println("- " + car);

}

}

}

}

public class Main {

public static void main(String[] args) {

CarWash cw = new CarWash();

cw.addNormal("Car1");

cw.addVIP("VIPCar");

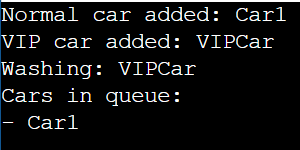
cw.wash();

cw.showQueue();

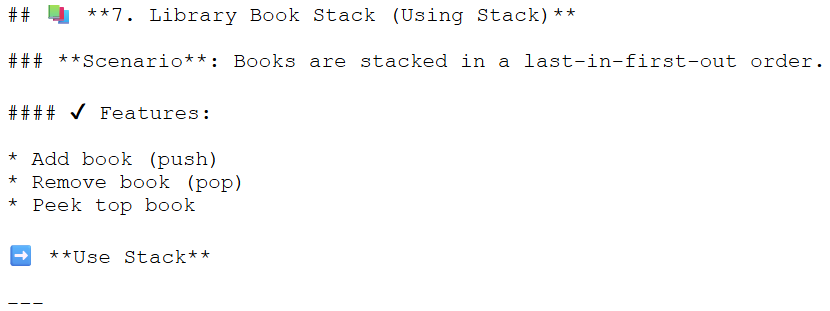
}

}

**OUTPUT:**

****

**7.**



**CODE:**

import java.util.\*;

class BookStack {

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

void addBook(String book) {

books.push(book);

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

}

void removeBook() {

if (!books.isEmpty()) {

String removed = books.pop();

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

} else {

System.out.println("No books to remove.");

}

}

void peekBook() {

if (!books.isEmpty()) {

System.out.println("Top book: " + books.peek());

} else {

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

}

}

}

public class Main {

public static void main(String[] args) {

BookStack bs = new BookStack();

bs.addBook("Book1");

bs.addBook("Book2");

bs.peekBook();

bs.removeBook();

bs.peekBook();

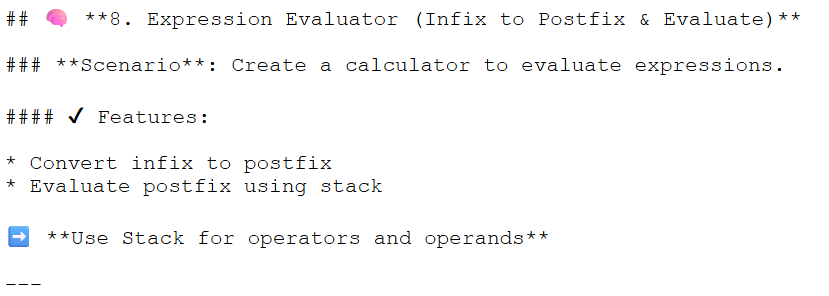
}

}

**OUTPUT:**



**8.**



**CODE:**

import java.util.\*;

class ExpressionEvaluator {

int evaluatePostfix(String exp) {

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

for (char c : exp.toCharArray()) {

if (Character.isDigit(c)) {

stack.push(c - '0');

} else {

int b = stack.pop(), a = stack.pop();

switch (c) {

case '+': stack.push(a + b); break;

case '-': stack.push(a - b); break;

case '\*': stack.push(a \* b); break;

case '/': stack.push(a / b); break;

}

}

}

return stack.pop();

}

String toPostfix(String infix) {

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

StringBuilder output = new StringBuilder();

for (char c : infix.toCharArray()) {

if (Character.isLetterOrDigit(c)) {

output.append(c);

} else if (c == '(') {

stack.push(c);

} else if (c == ')') {

while (!stack.isEmpty() && stack.peek() != '(') output.append(stack.pop());

stack.pop(); // remove '('

} else {

while (!stack.isEmpty() && precedence(c) <= precedence(stack.peek())) {

output.append(stack.pop());

}

stack.push(c);

}

}

while (!stack.isEmpty()) output.append(stack.pop());

return output.toString();

}

int precedence(char op) {

if (op == '+' || op == '-') return 1;

if (op == '\*' || op == '/') return 2;

return 0;

}

}

public class Main {

public static void main(String[] args) {

ExpressionEvaluator ee = new ExpressionEvaluator();

String infix = "3+(2\*4)";

String postfix = ee.toPostfix(infix);

System.out.println("Infix: " + infix);

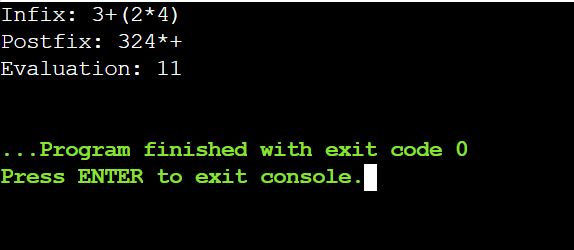
System.out.println("Postfix: " + postfix);

System.out.println("Evaluation: " + ee.evaluatePostfix(postfix));

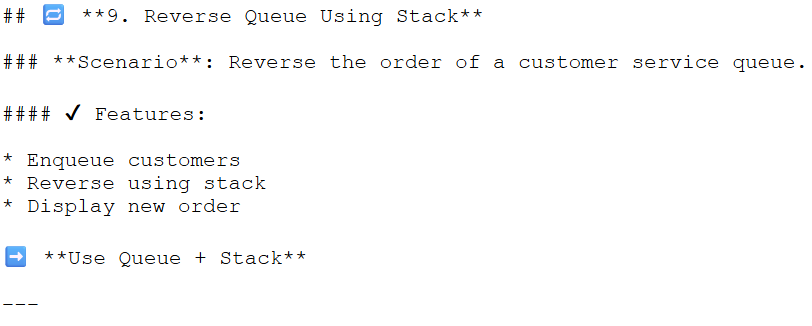
}

}

**OUTPUT:**



**9.**



**CODE:**

import java.util.\*;

class QueueReverser {

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

// Enqueue customer

void addCustomer(String name) {

queue.offer(name);

System.out.println("Enqueued: " + name);

}

// Reverse using stack

void reverseQueue() {

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

while (!queue.isEmpty()) {

stack.push(queue.poll());

}

while (!stack.isEmpty()) {

queue.offer(stack.pop());

}

System.out.println("Queue reversed successfully.");

}

// Display queue

void showQueue() {

if (queue.isEmpty()) {

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

} else {

System.out.println("Current queue order:");

for (String name : queue) {

System.out.println("- " + name);

}

}

}

}

public class Main {

public static void main(String[] args) {

QueueReverser qr = new QueueReverser();

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- Customer Service Queue Menu ---");

System.out.println("1. Enqueue customer");

System.out.println("2. Reverse queue");

System.out.println("3. Display queue");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

sc.nextLine(); // consume newline

switch (choice) {

case 1:

System.out.print("Enter customer name: ");

String name = sc.nextLine();

qr.addCustomer(name);

break;

case 2:

qr.reverseQueue();

break;

case 3:

qr.showQueue();

break;

case 4:

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

break;

default:

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

}

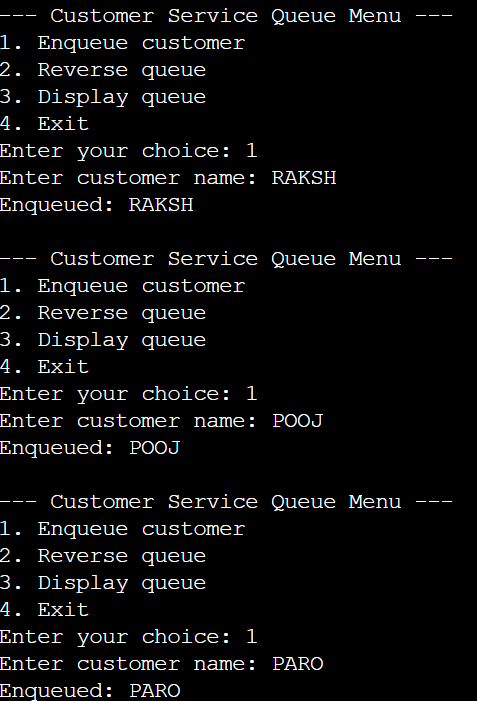
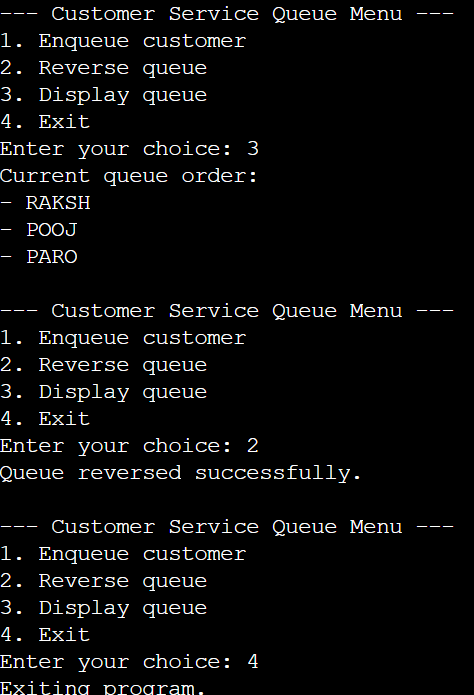
} while (choice != 4);

sc.close();

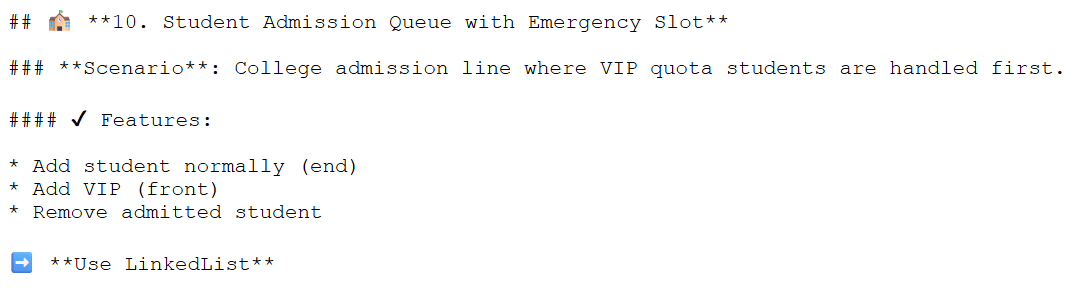
}

}

**OUTPUT:**

**10.**



**CODE:**

import java.util.\*;

class StudentAdmissionQueue {

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

// Add normal student to the end

void addStudent(String name) {

queue.addLast(name);

System.out.println("Student added: " + name);

}

// Add VIP student to the front

void addVIP(String name) {

queue.addFirst(name);

System.out.println("VIP student added: " + name);

}

// Remove admitted student from front

void admitStudent() {

if (!queue.isEmpty()) {

String admitted = queue.removeFirst();

System.out.println("Admitted: " + admitted);

} else {

System.out.println("No students in queue.");

}

}

// Display current queue

void showQueue() {

if (queue.isEmpty()) {

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

} else {

System.out.println("Current Admission Queue:");

for (String student : queue) {

System.out.println("- " + student);

}

}

}

}

public class Main {

public static void main(String[] args) {

StudentAdmissionQueue saq = new StudentAdmissionQueue();

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- Student Admission Menu ---");

System.out.println("1. Add normal student");

System.out.println("2. Add VIP student");

System.out.println("3. Admit next student");

System.out.println("4. View queue");

System.out.println("5. Exit");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

sc.nextLine(); // consume newline

switch (choice) {

case 1:

System.out.print("Enter student name: ");

String name = sc.nextLine();

saq.addStudent(name);

break;

case 2:

System.out.print("Enter VIP student name: ");

String vip = sc.nextLine();

saq.addVIP(vip);

break;

case 3:

saq.admitStudent();

break;

case 4:

saq.showQueue();

break;

case 5:

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

break;

default:

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

}

} while (choice != 5);

sc.close();

}

}

**OUTPUT:**

