1.Browser History(Using Stack)

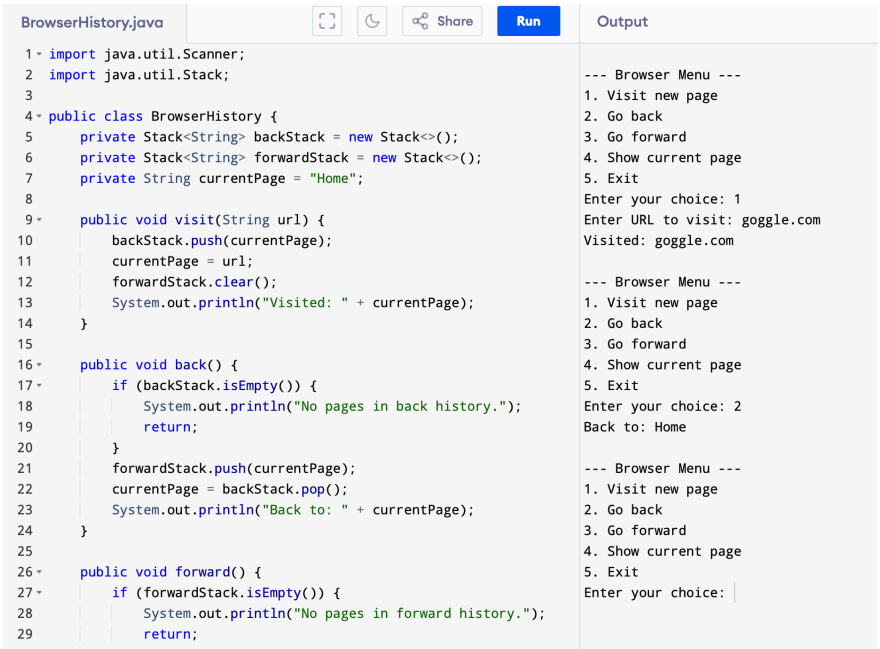
Scenario: Simulate a web browser’s back and forward functionality using two stacks.

Code:

```
import java.util.Scanner;
import java.util.Stack;

public class BrowserHistory {
    private Stack<String> backStack = new Stack<>();
    private Stack<String> forwardStack = new Stack<>();
    private String currentPage = "Home";

    public void visit(String url) {
        backStack.push(currentPage);
        currentPage = url;
        forwardStack.clear();
        System.out.println("Visited: " + currentPage);
    }

    public void back() {
        if (backStack.isEmpty()) {
            System.out.println("No pages in back history.");
            return;
        }
        forwardStack.push(currentPage);
        currentPage = backStack.pop();
        System.out.println("Back to: " + currentPage);
    }

    public void forward() {
        if (forwardStack.isEmpty()) {
            System.out.println("No pages in forward history.");
            return;
        }
        backStack.push(currentPage);
        currentPage = forwardStack.pop();
        System.out.println("Forward to: " + currentPage);
    }

    public void printCurrentPage() {
        System.out.println("Current Page: " + currentPage);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        BrowserHistory browser = new BrowserHistory();
        int choice;

        do {
            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("5. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter URL to visit: ");
                    String url = scanner.nextLine();
                    browser.visit(url);
                    break;
                case 2:
                    browser.back();
                    break;
                case 3:
                    browser.forward();
                    break;
                case 4:
                    browser.printCurrentPage();
                    break;
                case 5:
                    System.out.println("Exiting browser.");
                    break;
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        } while (choice != 5);

        scanner.close();
    }
}
```

Output:



2.Print Queue(Using LinkedList as Queue)

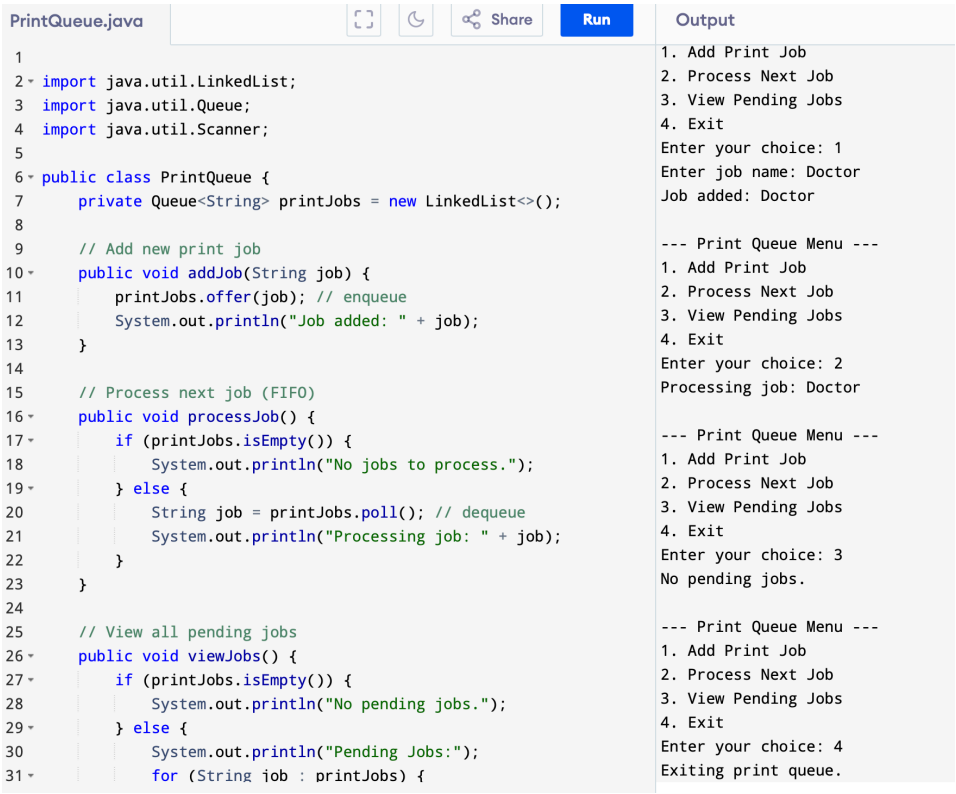
Scenario: Simulate a printer that handles print jobs in FIFO order.

Code:

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

public class PrintQueue {
    private Queue<String> printJobs = new LinkedList<>();
    public void addJob(String job) {
        printJobs.offer(job); // enqueue
        System.out.println("Job added: " + job);
    }
    public void processJob() {
        if (printJobs.isEmpty()) {
            System.out.println("No jobs to process.");
        } else {
            String job = printJobs.poll();
            System.out.println("Processing job: " + job);
        }
    }
    public void viewJobs() {
        if (printJobs.isEmpty()) {
            System.out.println("No pending jobs.");
        } else {
            System.out.println("Pending Jobs:");
            for (String job : printJobs) {
                System.out.println("- " + job);
            }
        }
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PrintQueue printer = new PrintQueue();
        int choice;

        do {
            System.out.println("\n--- Print Queue Menu ---");
            System.out.println("1. Add 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 = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter job name: ");
                    String job = scanner.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. Try again.");
        }
    } while (choice != 4);

    scanner.close();
  }
}
```

Output:



4.Undo-Redo Function(Using Stack)

Scenario:Track documents edits with undo and redo.

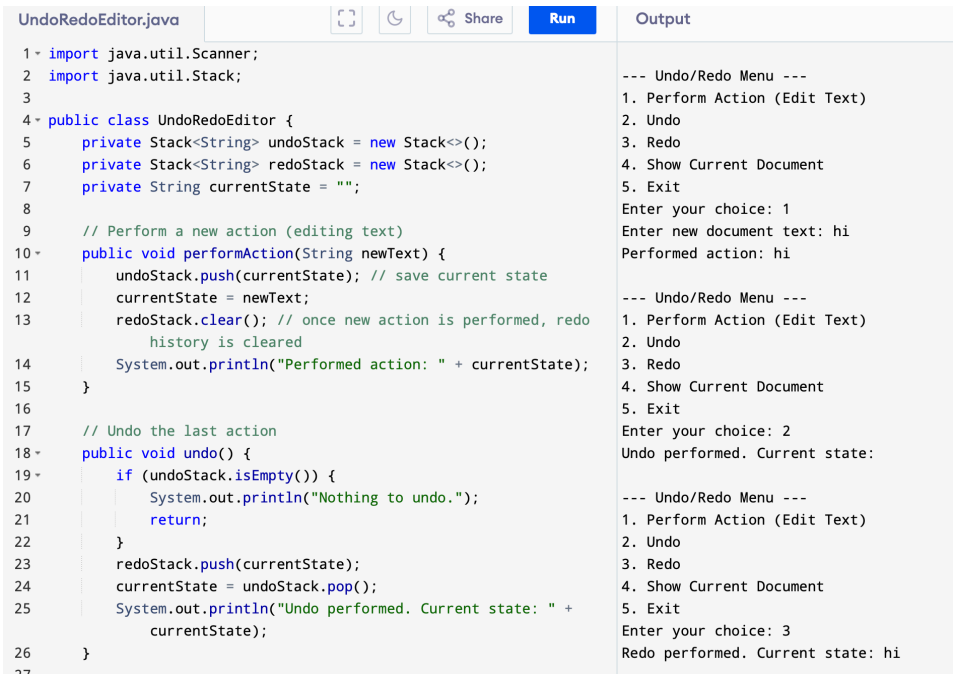
Code:

```
import java.util.Scanner;
import java.util.Stack;
```

```
public class UndoRedoEditor {
    private Stack<String> undoStack = new Stack<>();
    private Stack<String> redoStack = new Stack<>();
    private String currentState = "";
    public void performAction(String newText) {
        undoStack.push(currentState);
        currentState = newText;
        redoStack.clear();
        System.out.println("Performed action: " + currentState);
    }
    public void undo() {
        if (undoStack.isEmpty()) {
            System.out.println("Nothing to undo.");
            return;
        }
        redoStack.push(currentState);
        currentState = undoStack.pop();
        System.out.println("Undo performed. Current state: " + currentState);
    }
    public void redo() {
        if (redoStack.isEmpty()) {
            System.out.println("Nothing to redo.");
            return;
        }
        undoStack.push(currentState);
        currentState = redoStack.pop();
        System.out.println("Redo performed. Current state: " + currentState);
    }
    public void display() {
        System.out.println("Current Document State: " + currentState);
    }

    public static void main(String[] args) {
        UndoRedoEditor editor = new UndoRedoEditor();
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\n--- Undo/Redo Menu ---");
            System.out.println("1. Perform Action (Edit Text)");
            System.out.println("2. Undo");
            System.out.println("3. Redo");
            System.out.println("4. Show Current Document");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");

            while (!scanner.hasNextInt()) {
                System.out.println("Please enter a valid number (1-5): ");
                scanner.next();
            }

            choice = scanner.nextInt();
```

```
            scanner.nextLine();
            switch (choice) {
                case 1:
                    System.out.print("Enter new document text: ");
                    String newText = scanner.nextLine();
                    editor.performAction(newText);
                    break;

                case 2:
                    editor.undo();
                    break;

                case 3:
                    editor.redo();
                    break;

                case 4:
                    editor.display();
                    break;

                case 5:
                    System.out.println("Exiting editor.");
                    break;

                default:
                    System.out.println("Invalid choice. Try again.");
            }

        } while (choice != 5);

        scanner.close();
    }
}
```

Output:



5.Ticket Booking System(Using Queue)

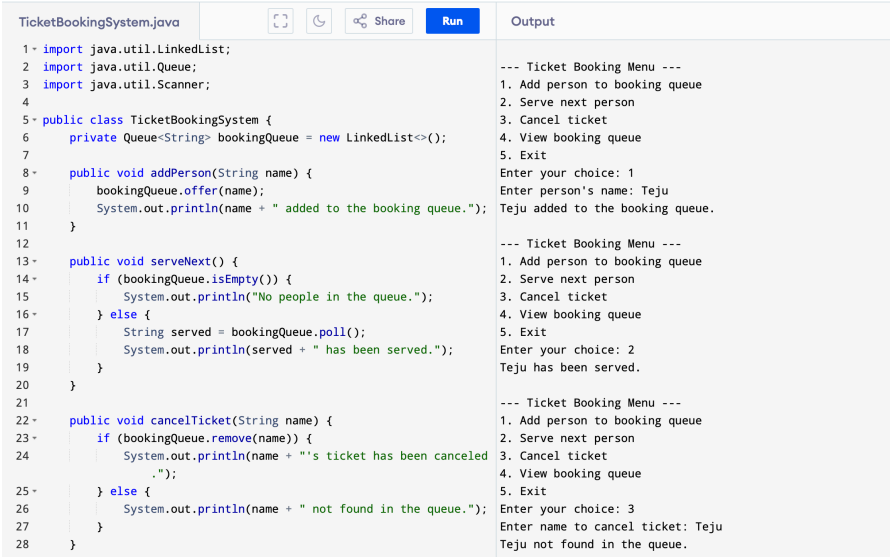
Scenario: People are queued to book movie/train tickets.

Code:

```
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

public class TicketBookingSystem {
    private Queue<String> bookingQueue = new LinkedList<>();

    public void addPerson(String name) {
        bookingQueue.offer(name);
        System.out.println(name + " added to the booking queue.");
    }

    public void serveNext() {
        if (bookingQueue.isEmpty()) {
            System.out.println("No people in the queue.");
        } else {
            String served = bookingQueue.poll();
            System.out.println(served + " has been served.");
        }
    }

    public void cancelTicket(String name) {
        if (bookingQueue.remove(name)) {
            System.out.println(name + "'s ticket has been canceled.");
        } else {
            System.out.println(name + " not found in the queue.");
        }
    }

    public void displayQueue() {
        if (bookingQueue.isEmpty()) {
            System.out.println("No pending bookings.");
        } else {
            System.out.println("People in the booking queue:");
            for (String person : bookingQueue) {
                System.out.println("- " + person);
            }
        }
    }

    public static void main(String[] args) {
        TicketBookingSystem system = new TicketBookingSystem();
        Scanner scanner = new Scanner(System.in);
        int choice = 0;

        while (choice != 5) {
            System.out.println("\n--- Ticket Booking Menu ---");
```

```
            System.out.println("1. Add person to booking queue");
            System.out.println("2. Serve next person");
            System.out.println("3. Cancel ticket");
            System.out.println("4. View booking queue");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");

            if (scanner.hasNextInt()) {
                choice = scanner.nextInt();
                scanner.nextLine();

                if (choice == 1) {
                    System.out.print("Enter person's name: ");
                    String name = scanner.nextLine();
                    system.addPerson(name);
                } else if (choice == 2) {
                    system.serveNext();
                } else if (choice == 3) {
                    System.out.print("Enter name to cancel ticket: ");
                    String cancelName = scanner.nextLine();
                    system.cancelTicket(cancelName);
                } else if (choice == 4) {
                    system.displayQueue();
                } else if (choice == 5) {
                    System.out.println("Exiting system.");
                } else {
                    System.out.println("Invalid choice. Please enter 1-5.");
                }
            } else {
                System.out.println("Please enter a number.");
                scanner.next(); // discard invalid input
            }
        }

        scanner.close();
    }
}
```

Output:



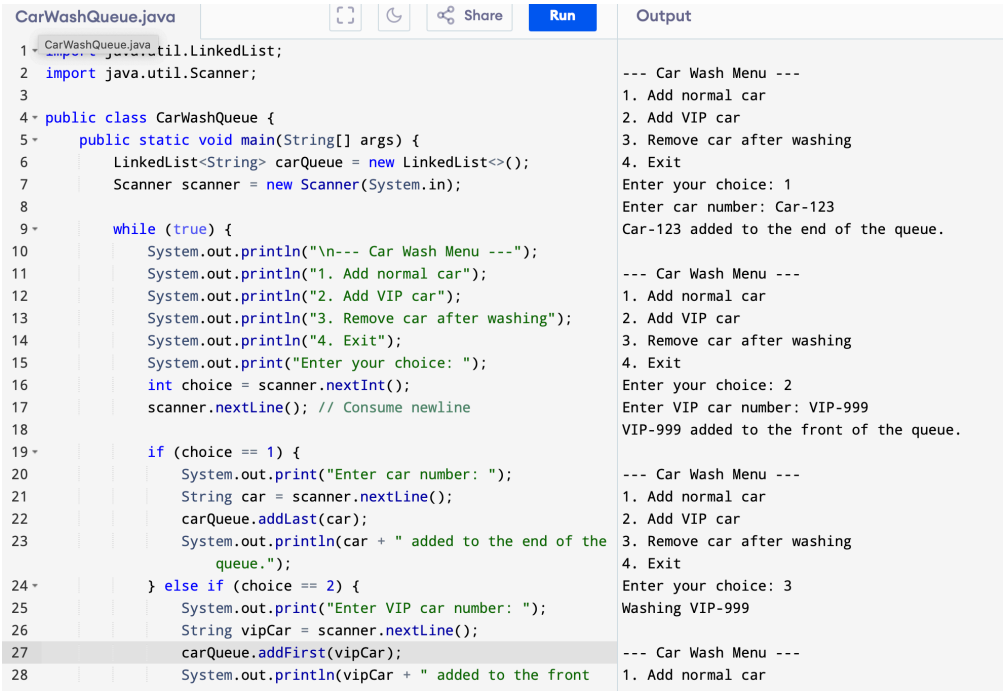
6.Car Wash Service Queue

Scenario: Cars line up at a car wash centre.

Code:

```
import java.util.LinkedList;
import java.util.Scanner;

public class CarWashQueue {
    public static void main(String[] args) {
        LinkedList<String> carQueue = new LinkedList<>();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\n--- Car Wash Menu ---");
            System.out.println("1. Add normal car");
            System.out.println("2. Add VIP car");
            System.out.println("3. Remove car after washing");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine();

            if (choice == 1) {
                System.out.print("Enter car number: ");
                String car = scanner.nextLine();
                carQueue.addLast(car);
                System.out.println(car + " added to the end of the queue.");
            } else if (choice == 2) {
                System.out.print("Enter VIP car number: ");
                String vipCar = scanner.nextLine();
                carQueue.addFirst(vipCar);
                System.out.println(vipCar + " added to the front of the
queue.");
            } else if (choice == 3) {
                if (carQueue.isEmpty()) {
                    System.out.println("No cars in the queue to wash.");
                } else {
                    String washedCar = carQueue.removeFirst();
                    System.out.println("Washing " + washedCar);
                }
            } else if (choice == 4) {
                System.out.println("Exiting system.");
                break;
            } else {
                System.out.println("Invalid choice. Try again.");
            }
        }

        scanner.close();
    }
}
```

Output:



7.Library Book Stack(Using Stack)

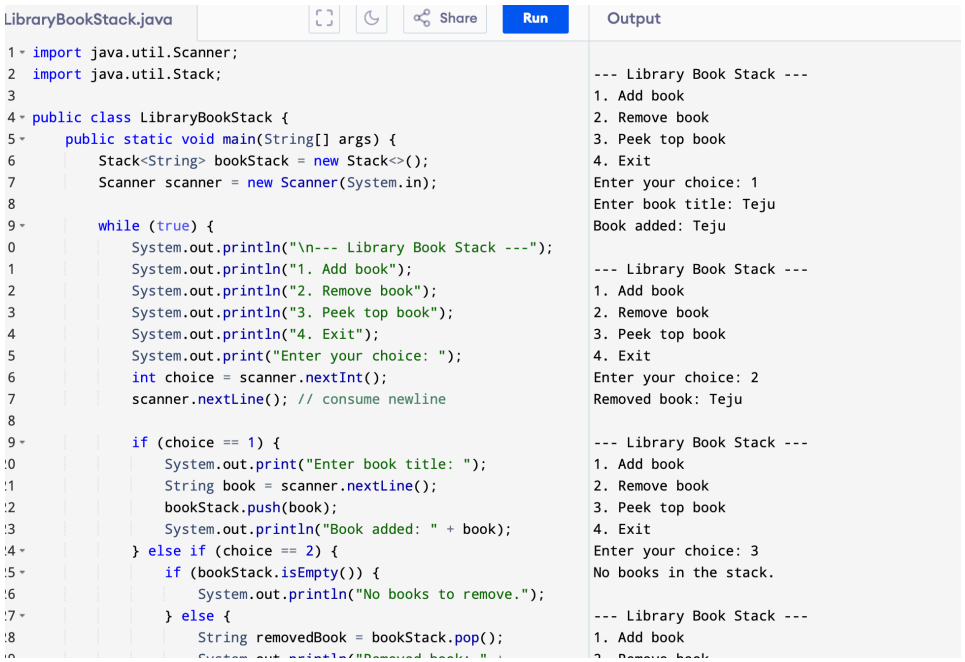
Scenario:Books are stacked in a last-in-first-out order.

Code:

```
import java.util.Scanner;
import java.util.Stack;

public class LibraryBookStack {
    public static void main(String[] args) {
        Stack<String> bookStack = new Stack<>();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\n--- Library Book Stack ---");
            System.out.println("1. Add book");
            System.out.println("2. Remove book");
            System.out.println("3. Peek top book");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            if (choice == 1) {
                System.out.print("Enter book title: ");
                String book = scanner.nextLine();
                bookStack.push(book);
                System.out.println("Book added: " + book);
            } else if (choice == 2) {
                if (bookStack.isEmpty()) {
```

```
                System.out.println("No books to remove.");
            } else {
                String removedBook = bookStack.pop();
                System.out.println("Removed book: " + removedBook);
            }
        } else if (choice == 3) {
            if (bookStack.isEmpty()) {
                System.out.println("No books in the stack.");
            } else {
                System.out.println("Top book: " + bookStack.peek());
            }
        } else if (choice == 4) {
            System.out.println("Exiting system.");
            break;
        } else {
            System.out.println("Invalid choice.");
        }
    }

    scanner.close();
  }
}
```

Output:



8.Expression Evaluator(Infix to postfix & Evaluate)

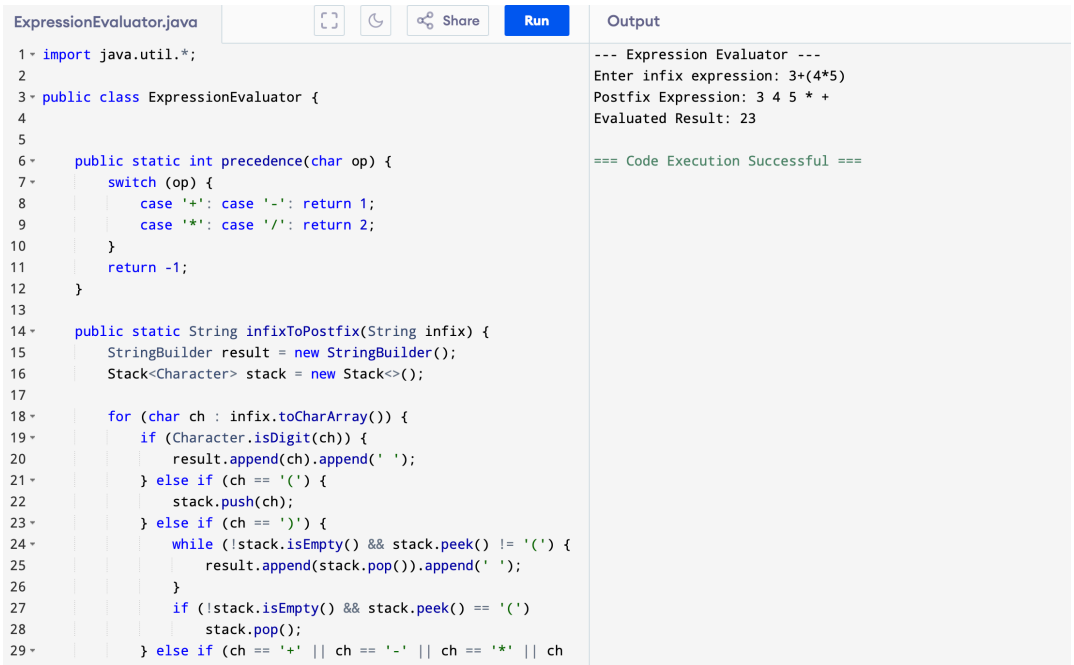
Scenario:Create a calculator to evaluate expressions.

Code:

```
import java.util.*;

public class ExpressionEvaluator {
    public static int precedence(char op) {
        switch (op) {
            case '+': case '-': return 1;
            case '*': case '/': return 2;
        }
        return -1;
    }
    public static String infixToPostfix(String infix) {
        StringBuilder result = new StringBuilder();
        Stack<Character> stack = new Stack<>();

        for (char ch : infix.toCharArray()) {
            if (Character.isDigit(ch)) {
                result.append(ch).append(' ');
            } else if (ch == '(') {
                stack.push(ch);
            } else if (ch == ')') {
                while (!stack.isEmpty() && stack.peek() != '(') {
                    result.append(stack.pop()).append(' ');
                }
                if (!stack.isEmpty() && stack.peek() == '(')
                    stack.pop();
            } else if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
                while (!stack.isEmpty() && precedence(ch) <=
precedence(stack.peek())) {
                    result.append(stack.pop()).append(' ');
                }
                stack.push(ch);
            }
        }

        while (!stack.isEmpty()) {
            result.append(stack.pop()).append(' ');
        }

        return result.toString().trim();
    }
    public static int evaluatePostfix(String postfix) {
        Stack<Integer> stack = new Stack<>();
        Scanner tokenizer = new Scanner(postfix);

        while (tokenizer.hasNext()) {
            if (tokenizer.hasNextInt()) {
                stack.push(tokenizer.nextInt());
            } else {
                char op = tokenizer.next().charAt(0);
                int b = stack.pop();
                int a = stack.pop();
                switch (op) {
```

```
                    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();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("--- Expression Evaluator ---");
        System.out.print("Enter infix expression (e.g., 3+(4*5)-6): ");
        String infix = scanner.nextLine().replaceAll("\\s+", "");

        String postfix = infixToPostfix(infix);
        System.out.println("Postfix Expression: " + postfix);

        int result = evaluatePostfix(postfix);
        System.out.println("Evaluated Result: " + result);

        scanner.close();
    }
}
```

Output:



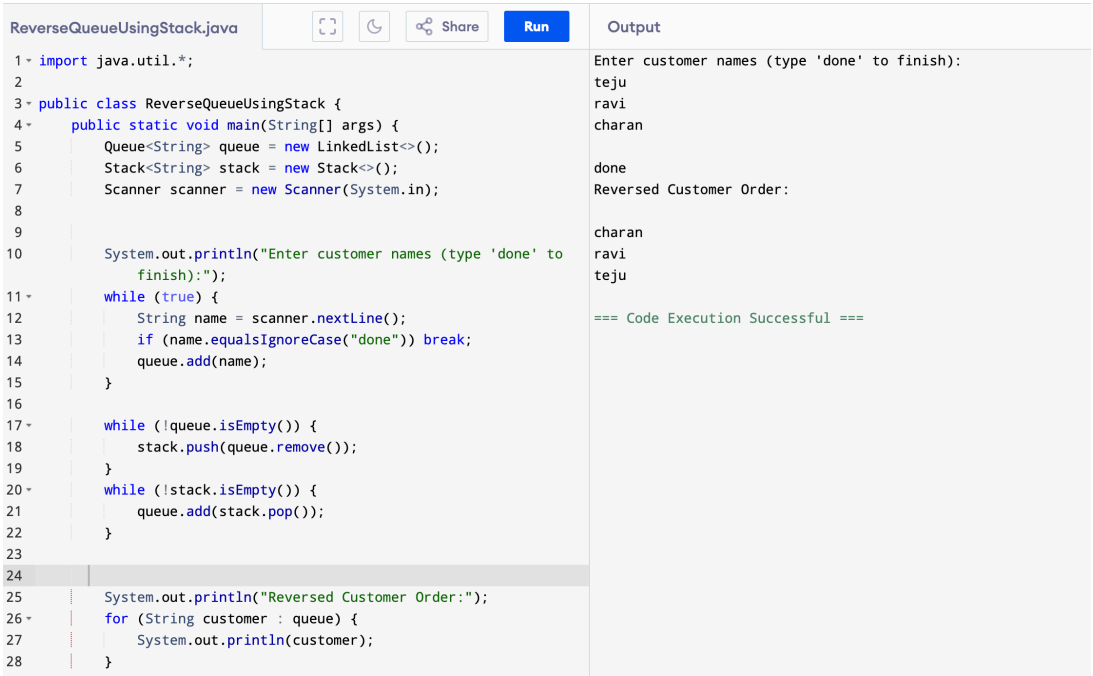
9.Reverse Queue Using Stack

Scenario: Reverse the order of a customer service queue.

Code:

```
import java.util.*;

public class ReverseQueueUsingStack {
    public static void main(String[] args) {
        Queue<String> queue = new LinkedList<>();
        Stack<String> stack = new Stack<>();
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter customer names (type 'done' to finish):");
        while (true) {
            String name = scanner.nextLine();
            if (name.equalsIgnoreCase("done")) break;
            queue.add(name);
        }

        while (!queue.isEmpty()) {
            stack.push(queue.remove());
        }
        while (!stack.isEmpty()) {
            queue.add(stack.pop());
        }

        System.out.println("Reversed Customer Order:");
        for (String customer : queue) {
            System.out.println(customer);
        }

        scanner.close();
    }
}
```

Output:



10.Student Admission Queue with Emergency Slot

Scenario: College admission line where VIP quota students are handled first.

Code:

```
import java.util.LinkedList;
import java.util.Scanner;

public class StudentAdmissionQueue {
    public static void main(String[] args) {
        LinkedList<String> admissionQueue = new LinkedList<>();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\n1. Add normal student");
            System.out.println("2. Add VIP student");
            System.out.println("3. Admit next student");
            System.out.print("Choose option: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            if (choice == 1) {
                System.out.print("Enter student name: ");
                String normalStudent = scanner.nextLine();
                admissionQueue.addLast(normalStudent);

            } else if (choice == 2) {
                System.out.print("Enter VIP student name: ");
                String vipStudent = scanner.nextLine();
                admissionQueue.addFirst(vipStudent);

            } else if (choice == 3) {
                if (!admissionQueue.isEmpty()) {
                    String admitted = admissionQueue.removeFirst();
                    System.out.println("Admitted: " + admitted);
                } else {
                    System.out.println("No students in queue.");
                }

            } else {
                System.out.println("Invalid option.");
            }
        }
    }
}
```

Output:

StudentAdmissionQueue.java Share Run Output

```java
import java.util.LinkedList;
import java.util.Scanner;

public class StudentAdmissionQueue {
    public static void main(String[] args) {
        LinkedList<String> admissionQueue = new LinkedList<>();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\n1. Add normal student");
            System.out.println("2. Add VIP student");
            System.out.println("3. Admit next student");
            System.out.print("Choose option: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            if (choice == 1) {
                System.out.print("Enter student name: ");
                String normalStudent = scanner.nextLine();
                admissionQueue.addLast(normalStudent);

            } else if (choice == 2) {
                System.out.print("Enter VIP student name: ");
                String vipStudent = scanner.nextLine();
                admissionQueue.addFirst(vipStudent);

            } else if (choice == 3) {
                if (!admissionQueue.isEmpty()) {
                    String admitted = admissionQueue.removeFirst
```

```
1. Add normal student
2. Add VIP student
3. Admit next student
Choose option: 1
Enter student name: teju

1. Add normal student
2. Add VIP student
3. Admit next student
Choose option: 2
Enter VIP student name: Teju

1. Add normal student
2. Add VIP student
3. Admit next student
Choose option: 3
Admitted: Teju

1. Add normal student
2. Add VIP student
3. Admit next student
Choose option:
```