

## 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:

BrowserHistory.java	Output
<pre> 1 import java.util.Scanner; 2 import java.util.Stack; 3 4 public class BrowserHistory { 5     private Stack&lt;String&gt; backStack = new Stack&lt;&gt;(); 6     private Stack&lt;String&gt; forwardStack = new Stack&lt;&gt;(); 7     private String currentPage = "Home"; 8 9     public void visit(String url) { 10         backStack.push(currentPage); 11         currentPage = url; 12         forwardStack.clear(); 13         System.out.println("Visited: " + currentPage); 14     } 15 16     public void back() { 17         if (backStack.isEmpty()) { 18             System.out.println("No pages in back history."); 19             return; 20         } 21         forwardStack.push(currentPage); 22         currentPage = backStack.pop(); 23         System.out.println("Back to: " + currentPage); 24     } 25 26     public void forward() { 27         if (forwardStack.isEmpty()) { 28             System.out.println("No pages in forward history."); 29             return; </pre>	<pre> --- Browser Menu --- 1. Visit new page 2. Go back 3. Go forward 4. Show current page 5. Exit Enter your choice: 1 Enter URL to visit: goggle.com Visited: goggle.com  --- Browser Menu --- 1. Visit new page 2. Go back 3. Go forward 4. Show current page 5. Exit Enter your choice: 2 Back to: Home  --- Browser Menu --- 1. Visit new page 2. Go back 3. Go forward 4. Show current page 5. Exit Enter your choice:   </pre>

## 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:
                    break;
            }
        } while (choice != 4);
    }
}
```

```

        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:

PrintQueue.java	Output
<pre> 1 2 import java.util.LinkedList; 3 import java.util.Queue; 4 import java.util.Scanner; 5 6 public class PrintQueue { 7     private Queue&lt;String&gt; printJobs = new LinkedList&lt;&gt;(); 8 9     // Add new print job 10 public void addJob(String job) { 11     printJobs.offer(job); // enqueue 12     System.out.println("Job added: " + job); 13 } 14 15 // Process next job (FIFO) 16 public void processJob() { 17     if (printJobs.isEmpty()) { 18         System.out.println("No jobs to process."); 19     } else { 20         String job = printJobs.poll(); // dequeue 21         System.out.println("Processing job: " + job); 22     } 23 } 24 25 // View all pending jobs 26 public void viewJobs() { 27     if (printJobs.isEmpty()) { 28         System.out.println("No pending jobs."); 29     } else { 30         System.out.println("Pending Jobs:"); 31         for (String job : printJobs) { </pre>	<pre> 1. Add Print Job 2. Process Next Job 3. View Pending Jobs 4. Exit Enter your choice: 1 Enter job name: Doctor Job added: Doctor  --- Print Queue Menu --- 1. Add Print Job 2. Process Next Job 3. View Pending Jobs 4. Exit Enter your choice: 2 Processing job: Doctor  --- Print Queue Menu --- 1. Add Print Job 2. Process Next Job 3. View Pending Jobs 4. Exit Enter your choice: 3 No pending jobs.  --- Print Queue Menu --- 1. Add Print Job 2. Process Next Job 3. View Pending Jobs 4. Exit Enter your choice: 4 Exiting print queue. </pre>

## 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:

UndoRedoEditor.java	Output
<pre> 1 import java.util.Scanner; 2 import java.util.Stack; 3 4 public class UndoRedoEditor { 5     private Stack&lt;String&gt; undoStack = new Stack&lt;&gt;(); 6     private Stack&lt;String&gt; redoStack = new Stack&lt;&gt;(); 7     private String currentState = ""; 8 9     // Perform a new action (editing text) 10    public void performAction(String newText) { 11        undoStack.push(currentState); // save current state 12        currentState = newText; 13        redoStack.clear(); // once new action is performed, redo 14        // history is cleared 15        System.out.println("Performed action: " + currentState); 16    } 17 18    // Undo the last action 19    public void undo() { 20        if (undoStack.isEmpty()) { 21            System.out.println("Nothing to undo."); 22            return; 23        } 24        redoStack.push(currentState); 25        currentState = undoStack.pop(); 26        System.out.println("Undo performed. Current state: " + 27            currentState); </pre>	<pre> --- Undo/Redo Menu --- 1. Perform Action (Edit Text) 2. Undo 3. Redo 4. Show Current Document 5. Exit Enter your choice: 1 Enter new document text: hi Performed action: hi  --- Undo/Redo Menu --- 1. Perform Action (Edit Text) 2. Undo 3. Redo 4. Show Current Document 5. Exit Enter your choice: 2 Undo performed. Current state:  --- Undo/Redo Menu --- 1. Perform Action (Edit Text) 2. Undo 3. Redo 4. Show Current Document 5. Exit Enter your choice: 3 Redo performed. Current state: hi </pre>

## 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:

TicketBookingSystem.java	Output
<pre> 1- import java.util.LinkedList; 2- import java.util.Queue; 3- import java.util.Scanner; 4 5- public class TicketBookingSystem { 6-     private Queue&lt;String&gt; bookingQueue = new LinkedList&lt;&gt;(); 7 8-     public void addPerson(String name) { 9-         bookingQueue.offer(name); 10-         System.out.println(name + " added to the booking queue."); 11-     } 12 13-     public void serveNext() { 14-         if (bookingQueue.isEmpty()) { 15-             System.out.println("No people in the queue."); 16-         } else { 17-             String served = bookingQueue.poll(); 18-             System.out.println(served + " has been served."); 19-         } 20-     } 21 22-     public void cancelTicket(String name) { 23-         if (bookingQueue.remove(name)) { 24-             System.out.println(name + "'s ticket has been canceled 25-         } else { 26-             System.out.println(name + " not found in the queue."); 27-         } 28-     } </pre>	<pre> --- Ticket Booking Menu --- 1. Add person to booking queue 2. Serve next person 3. Cancel ticket 4. View booking queue 5. Exit Enter your choice: 1 Enter person's name: Teju Teju added to the booking queue.  --- Ticket Booking Menu --- 1. Add person to booking queue 2. Serve next person 3. Cancel ticket 4. View booking queue 5. Exit Enter your choice: 2 Teju has been served.  --- Ticket Booking Menu --- 1. Add person to booking queue 2. Serve next person 3. Cancel ticket 4. View booking queue 5. Exit Enter your choice: 3 Enter name to cancel ticket: Teju Teju not found in the queue. </pre>



## 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:

CarWashQueue.java	Output
<pre>1 CarWashQueue.java 2 import java.util.Scanner; 3 4 public class CarWashQueue { 5     public static void main(String[] args) { 6         LinkedList&lt;String&gt; carQueue = new LinkedList&lt;&gt;(); 7         Scanner scanner = new Scanner(System.in); 8 9         while (true) { 10             System.out.println("\n--- Car Wash Menu ---"); 11             System.out.println("1. Add normal car"); 12             System.out.println("2. Add VIP car"); 13             System.out.println("3. Remove car after washing"); 14             System.out.println("4. Exit"); 15             System.out.print("Enter your choice: "); 16             int choice = scanner.nextInt(); 17             scanner.nextLine(); // Consume newline 18 19             if (choice == 1) { 20                 System.out.print("Enter car number: "); 21                 String car = scanner.nextLine(); 22                 carQueue.addLast(car); 23                 System.out.println(car + " added to the end of the queue."); 24             } else if (choice == 2) { 25                 System.out.print("Enter VIP car number: "); 26                 String vipCar = scanner.nextLine(); 27                 carQueue.addFirst(vipCar); 28                 System.out.println(vipCar + " added to the front of the queue.");</pre>	<pre>--- Car Wash Menu --- 1. Add normal car 2. Add VIP car 3. Remove car after washing 4. Exit Enter your choice: 1 Enter car number: Car-123 Car-123 added to the end of the queue.  --- Car Wash Menu --- 1. Add normal car 2. Add VIP car 3. Remove car after washing 4. Exit Enter your choice: 2 Enter VIP car number: VIP-999 VIP-999 added to the front of the queue.  --- Car Wash Menu --- 1. Add normal car 2. Add VIP car 3. Remove car after washing 4. Exit Enter your choice: 3 Washing VIP-999  --- Car Wash Menu --- 1. Add normal car</pre>

## 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("Stack is empty");
                } else {
                    bookStack.pop();
                    System.out.println("Book removed");
                }
            } else if (choice == 3) {
                if (bookStack.isEmpty()) {
                    System.out.println("Stack is empty");
                } else {
                    System.out.println("Top book: " + bookStack.peek());
                }
            } else if (choice == 4) {
                System.out.println("Exiting...");
                break;
            }
        }
    }
}
```

```

        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:

LibraryBookStack.java	Output
<pre> 1 import java.util.Scanner; 2 import java.util.Stack; 3 4 public class LibraryBookStack { 5     public static void main(String[] args) { 6         Stack&lt;String&gt; bookStack = new Stack&lt;&gt;(); 7         Scanner scanner = new Scanner(System.in); 8 9         while (true) { 10             System.out.println("\n--- Library Book Stack ---"); 11             System.out.println("1. Add book"); 12             System.out.println("2. Remove book"); 13             System.out.println("3. Peek top book"); 14             System.out.println("4. Exit"); 15             System.out.print("Enter your choice: "); 16             int choice = scanner.nextInt(); 17             scanner.nextLine(); // consume newline 18 19             if (choice == 1) { 20                 System.out.print("Enter book title: "); 21                 String book = scanner.nextLine(); 22                 bookStack.push(book); 23                 System.out.println("Book added: " + book); 24             } else if (choice == 2) { 25                 if (bookStack.isEmpty()) { 26                     System.out.println("No books to remove."); 27                 } else { 28                     String removedBook = bookStack.pop(); 29                     System.out.println("Removed book: " + removedBook); 30                 } 31             } else if (choice == 3) { 32                 if (bookStack.isEmpty()) { 33                     System.out.println("No books in the stack."); 34                 } else { 35                     System.out.println("Top book: " + bookStack.peek()); 36                 } 37             } else if (choice == 4) { 38                 System.out.println("Exiting system."); 39                 break; 40             } else { 41                 System.out.println("Invalid choice."); 42             } 43         } 44 45         scanner.close(); 46     } 47 } </pre>	<pre> --- Library Book Stack --- 1. Add book 2. Remove book 3. Peek top book 4. Exit Enter your choice: 1 Enter book title: Teju Book added: Teju  --- Library Book Stack --- 1. Add book 2. Remove book 3. Peek top book 4. Exit Enter your choice: 2 Removed book: Teju  --- Library Book Stack --- 1. Add book 2. Remove book 3. Peek top book 4. Exit Enter your choice: 3 No books in the stack.  --- Library Book Stack --- 1. Add book 2. Remove book 3. Peek top book 4. Exit Enter your choice: 4 Exiting system. </pre>

## 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:

ExpressionEvaluator.java	Output
<pre> 1 import java.util.*; 2 3 public class ExpressionEvaluator { 4 5 6     public static int precedence(char op) { 7         switch (op) { 8             case '+': case '-': return 1; 9             case '*': case '/': return 2; 10        } 11        return -1; 12    } 13 14    public static String infixToPostfix(String infix) { 15        StringBuilder result = new StringBuilder(); 16        Stack&lt;Character&gt; stack = new Stack&lt;&gt;(); 17 18        for (char ch : infix.toCharArray()) { 19            if (Character.isDigit(ch)) { 20                result.append(ch).append(' '); 21            } else if (ch == '(') { 22                stack.push(ch); 23            } else if (ch == ')') { 24                while (!stack.isEmpty() &amp;&amp; stack.peek() != '(') { 25                    result.append(stack.pop()).append(' '); 26                } 27                if (!stack.isEmpty() &amp;&amp; stack.peek() == '(') 28                    stack.pop(); 29            } else if (ch == '+'    ch == '-'    ch == '*'    ch </pre>	<pre> --- Expression Evaluator --- Enter infix expression: 3+(4*5)-6) Postfix Expression: 3 4 5 * + Evaluated Result: 23  === Code Execution Successful === </pre>

## 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:

ReverseQueueUsingStack.java	Run	Output
<pre>1- import java.util.*; 2 3- public class ReverseQueueUsingStack { 4-     public static void main(String[] args) { 5         Queue&lt;String&gt; queue = new LinkedList&lt;&gt;(); 6         Stack&lt;String&gt; stack = new Stack&lt;&gt;(); 7         Scanner scanner = new Scanner(System.in); 8 9 10        System.out.println("Enter customer names (type 'done' to         finish):"); 11-        while (true) { 12            String name = scanner.nextLine(); 13            if (name.equalsIgnoreCase("done")) break; 14            queue.add(name); 15        } 16 17-        while (!queue.isEmpty()) { 18            stack.push(queue.remove()); 19        } 20-        while (!stack.isEmpty()) { 21            queue.add(stack.pop()); 22        } 23 24 25        System.out.println("Reversed Customer Order:"); 26-        for (String customer : queue) { 27            System.out.println(customer); 28        } 29    } 30 }</pre>		<pre>Enter customer names (type 'done' to finish): teju ravi charan done Reversed Customer Order: charan ravi teju  === Code Execution Successful ===</pre>

## 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	Output
1 <code>import java.util.LinkedList;</code>	1. Add normal student
2 <code>import java.util.Scanner;</code>	2. Add VIP student
3	3. Admit next student
4 <code>public class StudentAdmissionQueue {</code>	Choose option: 1
5 <code>public static void main(String[] args) {</code>	Enter student name: teju
6 <code>LinkedList&lt;String&gt; admissionQueue = new LinkedList&lt;&gt;();</code>	
7 <code>Scanner scanner = new Scanner(System.in);</code>	1. Add normal student
8	2. Add VIP student
9 <code>while (true) {</code>	3. Admit next student
0 <code>System.out.println("\n1. Add normal student");</code>	Choose option: 2
1 <code>System.out.println("2. Add VIP student");</code>	Enter VIP student name: Teju
2 <code>System.out.println("3. Admit next student");</code>	
3 <code>System.out.print("Choose option: ");</code>	1. Add normal student
4 <code>int choice = scanner.nextInt();</code>	2. Add VIP student
5 <code>scanner.nextLine(); // consume newline</code>	3. Admit next student
6	Choose option: 3
7 <code>if (choice == 1) {</code>	Admitted: Teju
8 <code>System.out.print("Enter student name: ");</code>	
9 <code>String normalStudent = scanner.nextLine();</code>	1. Add normal student
0 <code>admissionQueue.addLast(normalStudent);</code>	2. Add VIP student
1	3. Admit next student
2 <code>} else if (choice == 2) {</code>	Choose option:
3 <code>System.out.print("Enter VIP student name: ");</code>	
4 <code>String vipStudent = scanner.nextLine();</code>	
5 <code>admissionQueue.addFirst(vipStudent);</code>	
6	
7 <code>} else if (choice == 3) {</code>	
8 <code>if (!admissionQueue.isEmpty()) {</code>	
9 <code>String admitted = admissionQueue.removeFirst</code>	