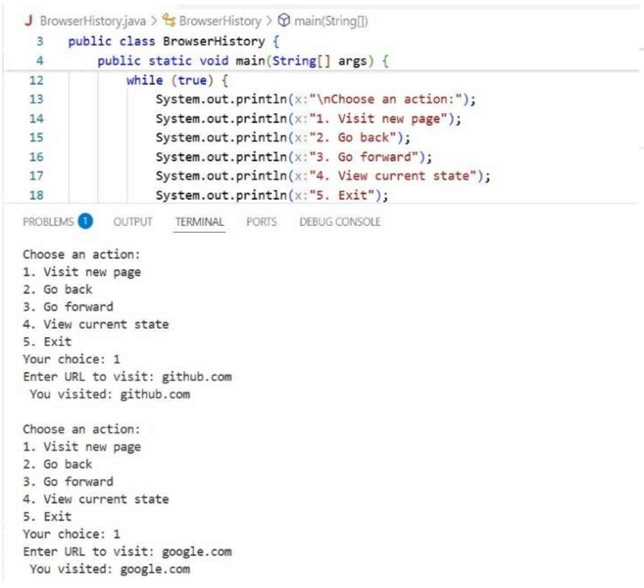
1. **Browser History (Using Stack)**

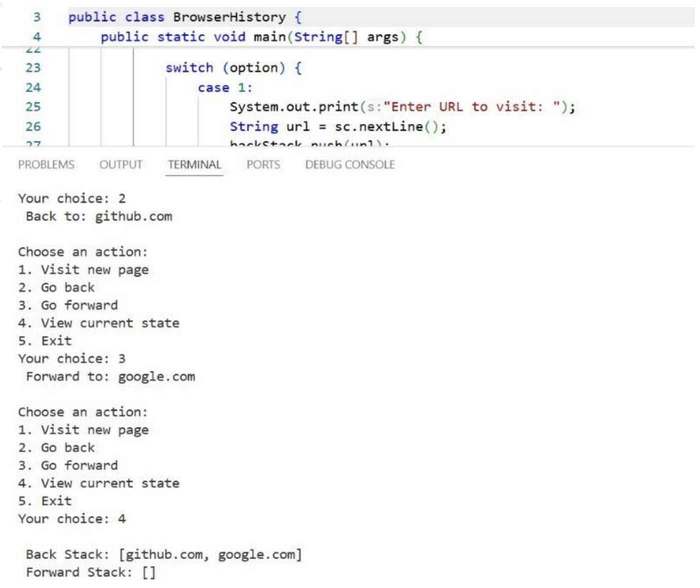
```
import java.util.*;

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

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

        System.out.println("Simple Browser History");

        while (true) {
            System.out.println("\nChoose an action:");
            System.out.println("1. Visit new page");
            System.out.println("2. Go back");
            System.out.println("3. Go forward");
            System.out.println("4. View current state");
            System.out.println("5. Exit");
            System.out.print("Your choice: ");
            int option = sc.nextInt();
            sc.nextLine();

            switch (option) {
                case 1:
                    System.out.print("Enter URL to visit: ");
                    String url = sc.nextLine();
                    backStack.push(url);
                    forwardStack.clear();
                    System.out.println(" You visited: " + url);
                    break;

                case 2:
                    if (backStack.size() <= 1) {
                        System.out.println(" No page to go back to.");
                    } else {
                        String lastPage = backStack.pop();
                        forwardStack.push(lastPage);
                        System.out.println(" Back to: " + backStack.peek());
                    }
                    break;
```

```java
                case 3:
                    if (forwardStack.isEmpty()) {
                        System.out.println(" No forward page available.");
                    } else {
                        String nextPage = forwardStack.pop();
                        backStack.push(nextPage);
                        System.out.println(" Forward to: " + nextPage);
                    }
                    break;

                case 4:
                    System.out.println("\n Back Stack: " + backStack);
                    System.out.println(" Forward Stack: " + forwardStack);
                    break;

                case 5:
                    System.out.println(" Exiting browser...");
                    return;

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



```java
public class BrowserHistory {
    public static void main(String[] args) {
        while (true) {
            System.out.println(x:"\nChoose an action:");
            System.out.println(x:"1. Visit new page");
            System.out.println(x:"2. Go back");
            System.out.println(x:"3. Go forward");
            System.out.println(x:"4. View current state");
            System.out.println(x:"5. Exit");
```

```
Choose an action:
1. Visit new page
2. Go back
3. Go forward
4. View current state
5. Exit
Your choice: 1
Enter URL to visit: github.com
 You visited: github.com

Choose an action:
1. Visit new page
2. Go back
3. Go forward
4. View current state
5. Exit
Your choice: 1
Enter URL to visit: google.com
 You visited: google.com
```

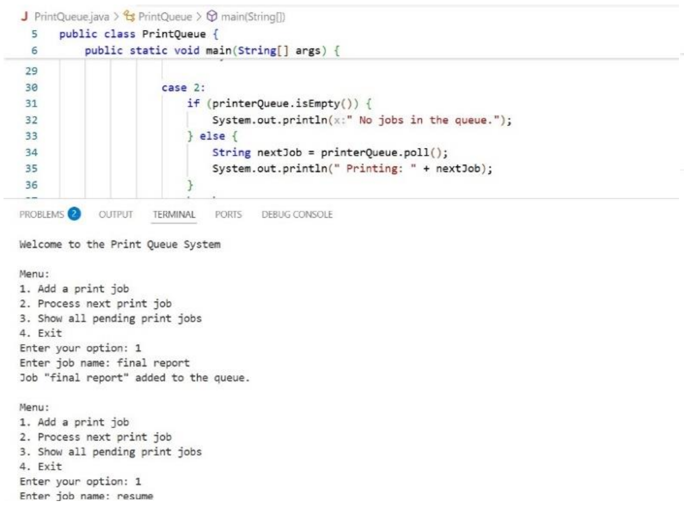


```java
public class BrowserHistory {
    public static void main(String[] args) {
            switch (option) {
                case 1:
                    System.out.print(s:"Enter URL to visit: ");
                    String url = sc.nextLine();
```

```
Your choice: 2
 Back to: github.com

Choose an action:
1. Visit new page
2. Go back
3. Go forward
4. View current state
5. Exit
Your choice: 3
 Forward to: google.com

Choose an action:
1. Visit new page
2. Go back
3. Go forward
4. View current state
5. Exit
Your choice: 4

 Back Stack: [github.com, google.com]
 Forward Stack: []
```

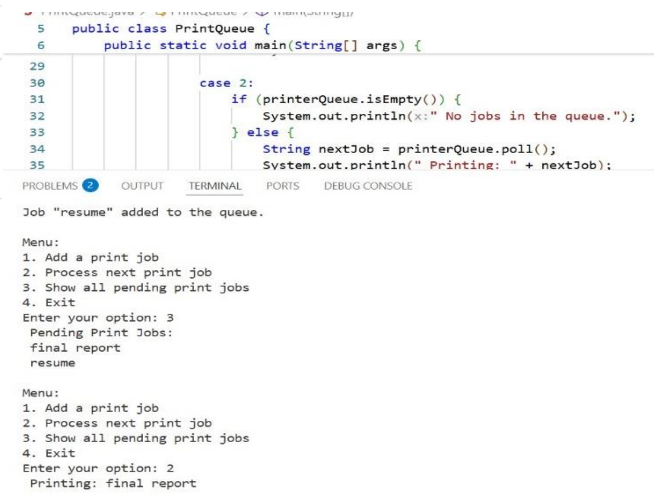
2. **Print Queue (Using LinkedList as Queue)**

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

public class PrintQueue {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Queue<String> printerQueue = new LinkedList<>();

        System.out.println("Welcome to the Print Queue System");

        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Add a print job");
            System.out.println("2. Process next print job");
            System.out.println("3. Show all pending print jobs");
            System.out.println("4. Exit");
            System.out.print("Enter your option: ");
            int choice = sc.nextInt();
            sc.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter job name: ");
                    String job = sc.nextLine();
                    printerQueue.offer(job);
                    System.out.println("Job \"" + job + "\" added to the queue.");
                    break;

                case 2:
                    if (printerQueue.isEmpty()) {
                        System.out.println(" No jobs in the queue.");
                    } else {
                        String nextJob = printerQueue.poll();
                        System.out.println(" Printing: " + nextJob);
                    }
                    break;

                case 3:
                    if (printerQueue.isEmpty()) {
                        System.out.println(" No pending print jobs.");
                    } else {
                        System.out.println(" Pending Print Jobs:");
                        for (String task : printerQueue) {
                            System.out.println(" " + task);
                        }
                    }
                    break;

                case 4:
                    System.out.println(" Exiting Print Queue System.");
                    return;

                default:
                    System.out.println(" Please enter a valid option.");
            }
        }
    }
}
```



```
PrintQueue.java > PrintQueue > main(String[])
public class PrintQueue {
    public static void main(String[] args) {

                case 2:
                    if (printerQueue.isEmpty()) {
                        System.out.println(x:" No jobs in the queue.");
                    } else {
                        String nextJob = printerQueue.poll();
                        System.out.println(" Printing: " + nextJob);
                    }
```

```
Welcome to the Print Queue System

Menu:
1. Add a print job
2. Process next print job
3. Show all pending print jobs
4. Exit
Enter your option: 1
Enter job name: final report
Job "final report" added to the queue.

Menu:
1. Add a print job
2. Process next print job
3. Show all pending print jobs
4. Exit
Enter your option: 1
Enter job name: resume
```

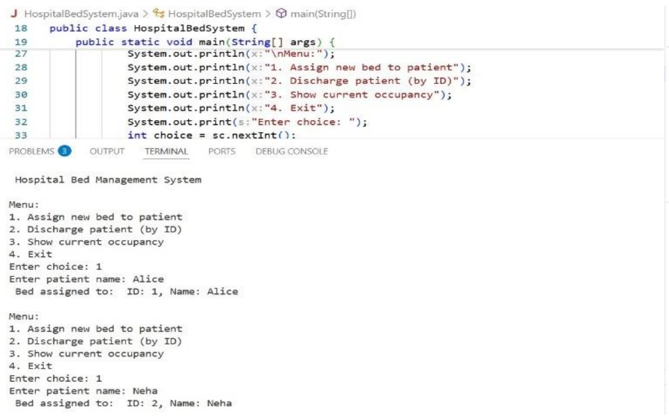


```
public class PrintQueue {
    public static void main(String[] args) {

                case 2:
                    if (printerQueue.isEmpty()) {
                        System.out.println(x:" No jobs in the queue.");
                    } else {
                        String nextJob = printerQueue.poll();
                        System.out.println(" Printing: " + nextJob);
```

```
Job "resume" added to the queue.

Menu:
1. Add a print job
2. Process next print job
3. Show all pending print jobs
4. Exit
Enter your option: 3
 Pending Print Jobs:
 final report
 resume

Menu:
1. Add a print job
2. Process next print job
3. Show all pending print jobs
4. Exit
Enter your option: 2
 Printing: final report
```

3. Hospital Bed Management (Using LinkedList)

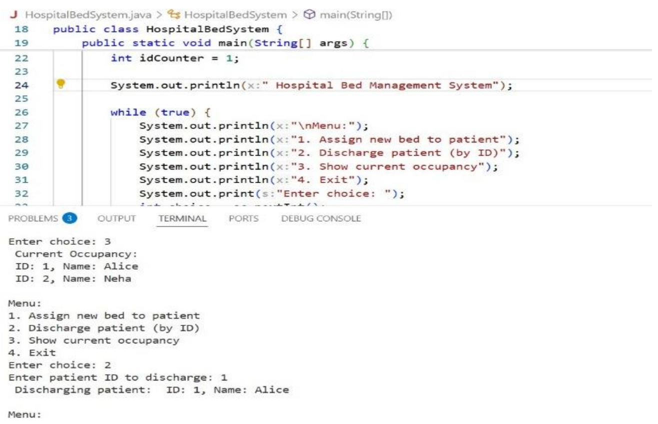
```
import java.util.LinkedList;
import java.util.Scanner;

class Patient {
    String name;
    int id;
    public Patient(String name, int id) {
        this.name = name;
        this.id = id;
    }

    public String toString() {
        return " ID: " + id + ", Name: " + name;
    }
}

public class HospitalBedSystem {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedList<Patient> bedList = new LinkedList<>();
        int idCounter = 1;

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

        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Assign new bed to patient");
            System.out.println("2. Discharge patient (by ID)");
            System.out.println("3. Show current occupancy");
            System.out.println("4. Exit");
            System.out.print("Enter choice: ");
            int choice = sc.nextInt();
            sc.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter patient name: ");
                    String name = sc.nextLine();
                    Patient newPatient = new Patient(name, idCounter++);
                    bedList.addLast(newPatient);
                    System.out.println(" Bed assigned to: " + newPatient);
                    break;

                case 2:
                    System.out.print("Enter patient ID to discharge: ");
                    int removeId = sc.nextInt();
                    boolean removed = false;
        for (int i = 0; i < bedList.size(); i++) {
          if (bedList.get(i).id == removeId) {
            System.out.println(" Discharging patient: " + bedList.get(i));
            bedList.remove(i);
            removed = true;
            break;
          }
        }
        if (!removed) {
          System.out.println(" No patient found with ID: " + removeId);
        }
        break;

      case 3:
        if (bedList.isEmpty()) {
          System.out.println(" All beds are empty.");
        } else {
          System.out.println(" Current Occupancy:");
          for (Patient p : bedList) {
            System.out.println(p);
          }
        }
        break;

      case 4:
        System.out.println(" Exiting Hospital System...");
        return;

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



HospitalBedSystem.java > HospitalBedSystem > main(String[])

```
public class HospitalBedSystem {
    public static void main(String[] args) {
            System.out.println(x:"\nMenu:");
            System.out.println(x:"1. Assign new bed to patient");
            System.out.println(x:"2. Discharge patient (by ID)");
            System.out.println(x:"3. Show current occupancy");
            System.out.println(x:"4. Exit");
            System.out.print(s:"Enter choice: ");
            int choice = sc.nextInt();
```

```
 Hospital Bed Management System

Menu:
1. Assign new bed to patient
2. Discharge patient (by ID)
3. Show current occupancy
4. Exit
Enter choice: 1
Enter patient name: Alice
 Bed assigned to:  ID: 1, Name: Alice

Menu:
1. Assign new bed to patient
2. Discharge patient (by ID)
3. Show current occupancy
4. Exit
Enter choice: 1
Enter patient name: Neha
 Bed assigned to:  ID: 2, Name: Neha
```



HospitalBedSystem.java > HospitalBedSystem > main(String[])

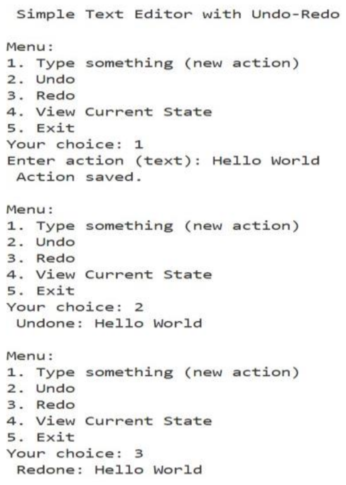
```
public class HospitalBedSystem {
    public static void main(String[] args) {
        int idCounter = 1;

        System.out.println(x:" Hospital Bed Management System");

        while (true) {
            System.out.println(x:"\nMenu:");
            System.out.println(x:"1. Assign new bed to patient");
            System.out.println(x:"2. Discharge patient (by ID)");
            System.out.println(x:"3. Show current occupancy");
            System.out.println(x:"4. Exit");
            System.out.print(s:"Enter choice: ");
```

```
Enter choice: 3
 Current Occupancy:
 ID: 1, Name: Alice
 ID: 2, Name: Neha

Menu:
1. Assign new bed to patient
2. Discharge patient (by ID)
3. Show current occupancy
4. Exit
Enter choice: 2
Enter patient ID to discharge: 1
 Discharging patient:  ID: 1, Name: Alice

Menu:
```

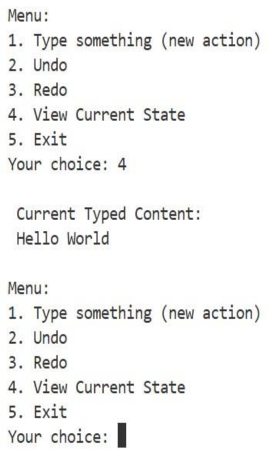
4. Undo-Redo Function (Using Stack)

```
import java.util.*;

public class UndoRedoEditor {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Stack<String> undoStack = new Stack<>();
        Stack<String> redoStack = new Stack<>();

        System.out.println(" Simple Text Editor with Undo-Redo");

        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Type something (new action)");
            System.out.println("2. Undo");
            System.out.println("3. Redo");
            System.out.println("4. View Current State");
            System.out.println("5. Exit");
            System.out.print("Your choice: ");
            int choice = sc.nextInt();
            sc.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter action (text): ");
                    String action = sc.nextLine();
                    undoStack.push(action);
                    redoStack.clear();
                    System.out.println(" Action saved.");
                    break;

                case 2:
                    if (undoStack.isEmpty()) {
                        System.out.println(" Nothing to undo.");
                    } else {
                        String undone = undoStack.pop();
                        redoStack.push(undone);
                        System.out.println(" Undone: " + undone);
                    }
                    break;

                case 3:
```

```
                if (redoStack.isEmpty()) {
                    System.out.println(" Nothing to redo.");
                } else {
                    String redone = redoStack.pop();
                    undoStack.push(redone);
                    System.out.println(" Redone: " + redone);
                }
                break;

            case 4:
                System.out.println("\n Current Typed Content:");
                for (String act : undoStack) {
                    System.out.println(" " + act);
                }
                break;

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

            default:
                System.out.println(" Invalid option.");
          }
        }
      }
    }
```



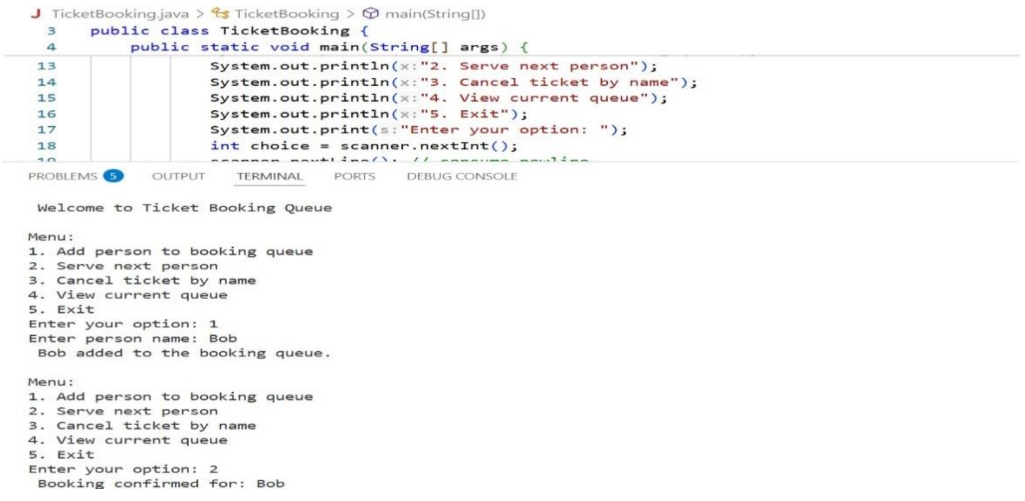
```
 Simple Text Editor with Undo-Redo

Menu:
1. Type something (new action)
2. Undo
3. Redo
4. View Current State
5. Exit
Your choice: 1
Enter action (text): Hello World
 Action saved.

Menu:
1. Type something (new action)
2. Undo
3. Redo
4. View Current State
5. Exit
Your choice: 2
 Undone: Hello World

Menu:
1. Type something (new action)
2. Undo
3. Redo
4. View Current State
5. Exit
Your choice: 3
 Redone: Hello World
```



```
Menu:
1. Type something (new action)
2. Undo
3. Redo
4. View Current State
5. Exit
Your choice: 4

 Current Typed Content:
 Hello World

Menu:
1. Type something (new action)
2. Undo
3. Redo
4. View Current State
5. Exit
Your choice:
```

5. Ticket Booking System (Using Queue)

```
import java.util.*;

public class TicketBooking {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Queue<String> bookingQueue = new LinkedList<>();

        System.out.println(" Welcome to Ticket Booking Queue");

        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Add person to booking queue");
            System.out.println("2. Serve next person");
            System.out.println("3. Cancel ticket by name");
            System.out.println("4. View current queue");
            System.out.println("5. Exit");
            System.out.print("Enter your option: ");
            int choice = sc.nextInt();
            sc.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter person name: ");
                    String name = sc.nextLine();
                    bookingQueue.offer(name);
                    System.out.println(" " + name + " added to the booking queue.");
                    break;

                case 2:
                    if (bookingQueue.isEmpty()) {
                        System.out.println(" No one is waiting in the queue.");
                    } else {
                        String next = bookingQueue.poll();
                        System.out.println(" Booking confirmed for: " + next);
                    }
                    break;

                case 3:
                    System.out.print("Enter name to cancel booking: ");
                    String cancelName = sc.nextLine();
                    if (bookingQueue.remove(cancelName)) {
                        System.out.println(" Booking cancelled for: " + cancelName);
                    } else {
                        System.out.println(" No booking found under name: " + cancelName);
                    }
                    break;

                case 4:
                    if (bookingQueue.isEmpty()) {
                        System.out.println(" The booking queue is currently empty.");
                    } else {
                        System.out.println(" People in queue:");
                        for (String person : bookingQueue) {
                            System.out.println(" " + person);
                        }
                    }
                    break;

                case 5:
                    System.out.println(" Exiting ticket booking system...");
                    return;

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

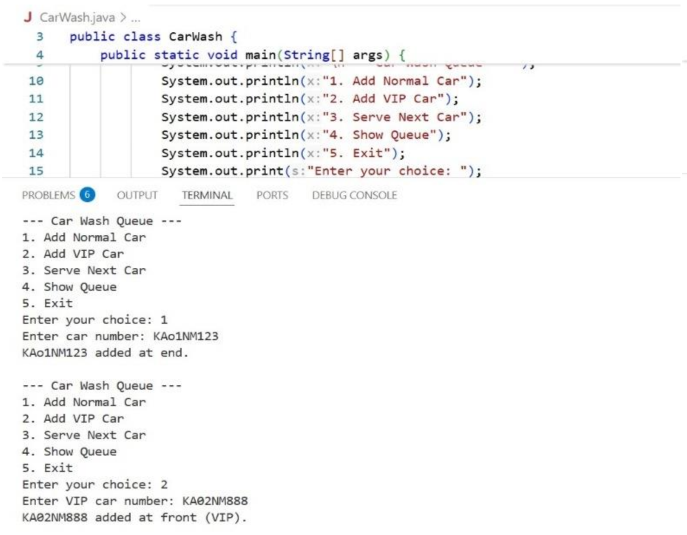


```
public class TicketBooking {
    public static void main(String[] args) {
            System.out.println(x:"2. Serve next person");
            System.out.println(x:"3. Cancel ticket by name");
            System.out.println(x:"4. View current queue");
            System.out.println(x:"5. Exit");
            System.out.print(s:"Enter your option: ");
            int choice = scanner.nextInt();
```

```
 Welcome to Ticket Booking Queue

Menu:
1. Add person to booking queue
2. Serve next person
3. Cancel ticket by name
4. View current queue
5. Exit
Enter your option: 1
Enter person name: Bob
 Bob added to the booking queue.

Menu:
1. Add person to booking queue
2. Serve next person
3. Cancel ticket by name
4. View current queue
5. Exit
Enter your option: 2
 Booking confirmed for: Bob
```

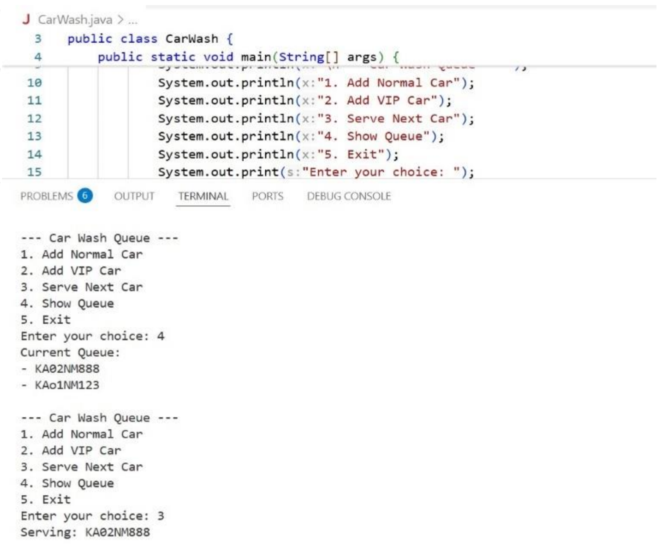
6. **Car Wash Service Queue**

```
import java.util.*;

public class CarWash {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedList<String> queue = new LinkedList<>();

        while (true) {
            System.out.println("\n--- Car Wash Queue ---");
            System.out.println("1. Add Normal Car");
            System.out.println("2. Add VIP Car");
            System.out.println("3. Serve Next Car");
            System.out.println("4. Show Queue");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            int choice = sc.nextInt();
            sc.nextLine();

            if (choice == 1) {
                System.out.print("Enter car number: ");
                String car = sc.nextLine();
                queue.addLast(car);
                System.out.println(car + " added at end.");
            } else if (choice == 2) {
                System.out.print("Enter VIP car number: ");
                String car = sc.nextLine();
                queue.addFirst(car);
                System.out.println(car + " added at front (VIP).");
            } else if (choice == 3) {
                if (queue.isEmpty()) {
                    System.out.println("No cars to serve.");
                } else {
                    System.out.println("Serving: " + queue.removeFirst());
                }
            } else if (choice == 4) {
                System.out.println("Current Queue:");
                for (String car : queue) {
                    System.out.println("- " + car);
                }
            } else if (choice == 5) {
                System.out.println("Thank you!");
                break;
            } else {
                System.out.println("Invalid choice.");
            }
        }
    }
}
```



```
CarWash.java > ...
public class CarWash {
    public static void main(String[] args) {
            System.out.println(x:"1. Add Normal Car");
            System.out.println(x:"2. Add VIP Car");
            System.out.println(x:"3. Serve Next Car");
            System.out.println(x:"4. Show Queue");
            System.out.println(x:"5. Exit");
            System.out.print(s:"Enter your choice: ");
```

```
--- Car Wash Queue ---
1. Add Normal Car
2. Add VIP Car
3. Serve Next Car
4. Show Queue
5. Exit
Enter your choice: 1
Enter car number: KAo1NM123
KAo1NM123 added at end.

--- Car Wash Queue ---
1. Add Normal Car
2. Add VIP Car
3. Serve Next Car
4. Show Queue
5. Exit
Enter your choice: 2
Enter VIP car number: KA02NM888
KA02NM888 added at front (VIP).
```



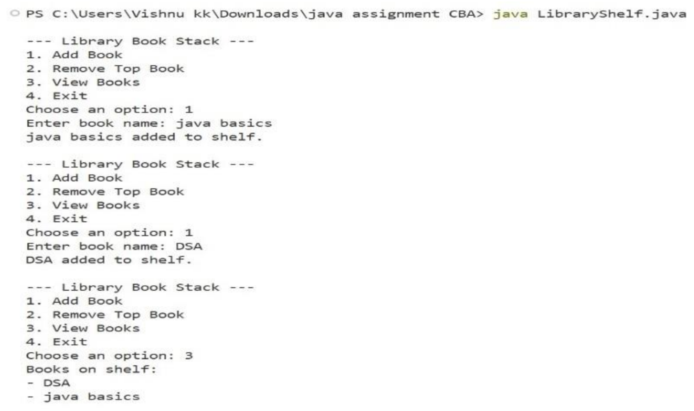
```
CarWash.java > ...
public class CarWash {
    public static void main(String[] args) {
            System.out.println(x:"1. Add Normal Car");
            System.out.println(x:"2. Add VIP Car");
            System.out.println(x:"3. Serve Next Car");
            System.out.println(x:"4. Show Queue");
            System.out.println(x:"5. Exit");
            System.out.print(s:"Enter your choice: ");
```

```
--- Car Wash Queue ---
1. Add Normal Car
2. Add VIP Car
3. Serve Next Car
4. Show Queue
5. Exit
Enter your choice: 4
Current Queue:
- KA02NM888
- KAo1NM123

--- Car Wash Queue ---
1. Add Normal Car
2. Add VIP Car
3. Serve Next Car
4. Show Queue
5. Exit
Enter your choice: 3
Serving: KA02NM888
```

---

7. Library Book Stack (Using Stack)

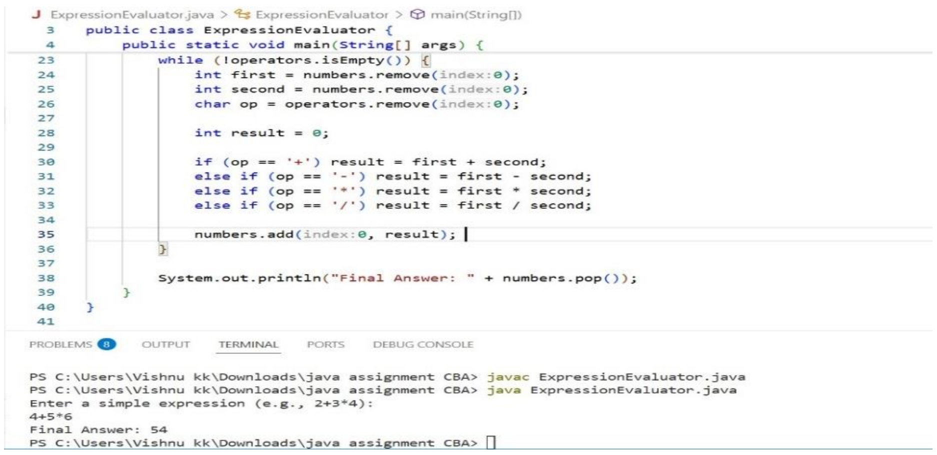
```
import java.util.*;

public class LibraryShelf {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Stack<String> shelf = new Stack<>();

        while (true) {
            System.out.println("\n--- Library Book Stack ---");
            System.out.println("1. Add Book");
            System.out.println("2. Remove Top Book");
            System.out.println("3. View Books");
            System.out.println("4. Exit");
            System.out.print("Choose an option: ");
            int option = sc.nextInt();
            sc.nextLine();
            if (option == 1) {
                System.out.print("Enter book name: ");
                String book = sc.nextLine();
                shelf.push(book);
                System.out.println(book + " added to shelf.");
            } else if (option == 2) {
                if (shelf.isEmpty()) {
                    System.out.println("No books to remove.");
                } else {
                    String removed = shelf.pop();
                    System.out.println("Removed: " + removed);
                }
            } else if (option == 3) {
                if (shelf.isEmpty()) {
                    System.out.println("Shelf is empty.");
                } else {
                    System.out.println("Books on shelf:");
                    for (int i = shelf.size() - 1; i >= 0; i--) {
                        System.out.println("- " + shelf.get(i));
                    }
                }
            } else if (option == 4) {
                System.out.println("Exiting.");
                break;
            } else {
                System.out.println("Invalid option.");
            }
        }}}
```



```
PS C:\Users\Vishnu kk\Downloads\java assignment CBA> java LibraryShelf.java

--- Library Book Stack ---
1. Add Book
2. Remove Top Book
3. View Books
4. Exit
Choose an option: 1
Enter book name: java basics
java basics added to shelf.

--- Library Book Stack ---
1. Add Book
2. Remove Top Book
3. View Books
4. Exit
Choose an option: 1
Enter book name: DSA
DSA added to shelf.

--- Library Book Stack ---
1. Add Book
2. Remove Top Book
3. View Books
4. Exit
Choose an option: 3
Books on shelf:
- DSA
- java basics
```

8. Expression Evaluator (Infix to Postfix & Evaluate)

```
import java.util.*;

public class ExpressionEvaluator {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      Stack<Integer> numbers = new Stack<>();
      Stack<Character> operators = new Stack<>();

      System.out.println("Enter a simple expression (e.g., 2+3*4):");
      String expression = input.nextLine();

      for (int i = 0; i < expression.length(); i++) {
         char ch = expression.charAt(i);

         if (Character.isDigit(ch)) {
            numbers.push(ch - '0');
         } else if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
            operators.push(ch);
         }
      }

      while (!operators.isEmpty()) {
         int first = numbers.remove(0);
         int second = numbers.remove(0);
         char op = operators.remove(0);

         int result = 0;

         if (op == '+') result = first + second;
         else if (op == '-') result = first - second;
         else if (op == '*') result = first * second;
         else if (op == '/') result = first / second;

         numbers.add(0, result);
      }

      System.out.println("Final Answer: " + numbers.pop());
   }
}
```



ExpressionEvaluator.java > ExpressionEvaluator > main(String[])

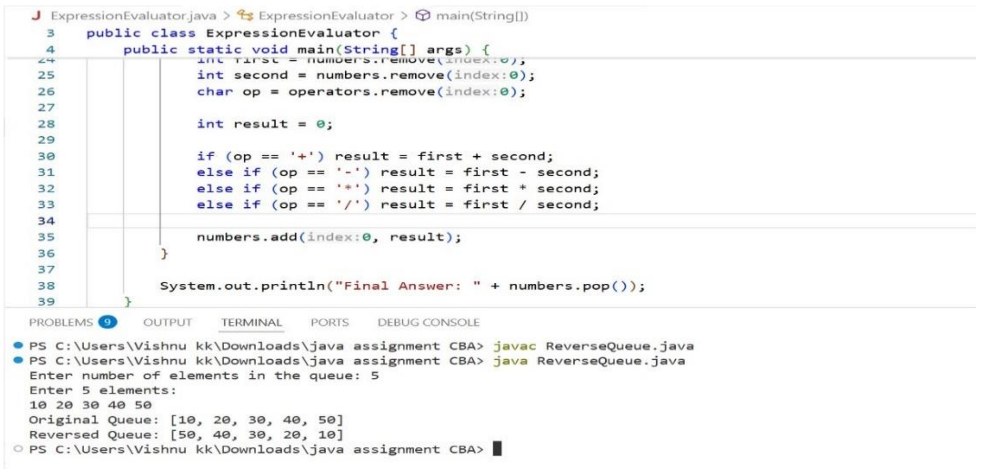
```java
public class ExpressionEvaluator {
    public static void main(String[] args) {
        while (!operators.isEmpty()) {
            int first = numbers.remove(index:0);
            int second = numbers.remove(index:0);
            char op = operators.remove(index:0);

            int result = 0;

            if (op == '+') result = first + second;
            else if (op == '-') result = first - second;
            else if (op == '*') result = first * second;
            else if (op == '/') result = first / second;

            numbers.add(index:0, result);
        }

        System.out.println("Final Answer: " + numbers.pop());
    }
}
```

```
PS C:\Users\Vishnu kk\Downloads\java assignment CBA> javac ExpressionEvaluator.java
PS C:\Users\Vishnu kk\Downloads\java assignment CBA> java ExpressionEvaluator.java
Enter a simple expression (e.g., 2+3*4):
4+5*6
Final Answer: 54
PS C:\Users\Vishnu kk\Downloads\java assignment CBA>
```

---

9. **Reverse Queue Using Stack**

```java
import java.util.*;

public class ReverseQueue {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Queue<Integer> queue = new LinkedList<>();
        Stack<Integer> stack = new Stack<>();

        System.out.print("Enter number of elements in the queue: ");
        int n = sc.nextInt();

        System.out.println("Enter " + n + " elements:");
        for (int i = 0; i < n; i++) {
            int val = sc.nextInt();
            queue.add(val);
        }

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

        //  Push all elements to stack
        while (!queue.isEmpty()) {
            stack.push(queue.remove());
        }

        // Pop from stack and add back to queue
        while (!stack.isEmpty()) {
            queue.add(stack.pop());
        }
        System.out.println("Reversed Queue: " + queue);
    }
}
```


ExpressionEvaluator.java > ExpressionEvaluator > main(String[])

```
public class ExpressionEvaluator {
    public static void main(String[] args) {
            int first = numbers.remove(index:0);
            int second = numbers.remove(index:0);
            char op = operators.remove(index:0);

            int result = 0;

            if (op == '+') result = first + second;
            else if (op == '-') result = first - second;
            else if (op == '*') result = first * second;
            else if (op == '/') result = first / second;

            numbers.add(index:0, result);
        }

        System.out.println("Final Answer: " + numbers.pop());
    }
```

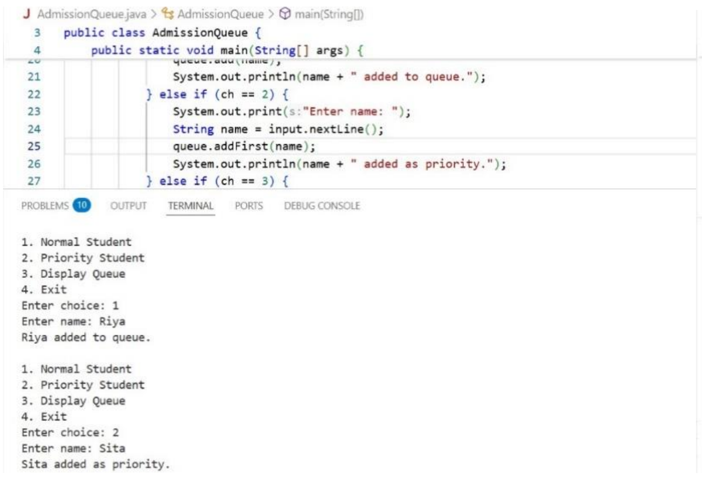
```
PS C:\Users\Vishnu kk\Downloads\java assignment CBA> javac ReverseQueue.java
PS C:\Users\Vishnu kk\Downloads\java assignment CBA> java ReverseQueue.java
Enter number of elements in the queue: 5
Enter 5 elements:
10 20 30 40 50
Original Queue: [10, 20, 30, 40, 50]
Reversed Queue: [50, 40, 30, 20, 10]
PS C:\Users\Vishnu kk\Downloads\java assignment CBA>
```

---

10. Student Admission Queue with Emergency Slot

```
import java.util.*;

public class AdmissionQueue {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        LinkedList<String> queue = new LinkedList<>();

        while (true) {
            System.out.println("\n1. Normal Student");
            System.out.println("2. Priority Student");
            System.out.println("3. Display Queue");
            System.out.println("4. Exit");
            System.out.print("Enter choice: ");
            int ch = input.nextInt();
            input.nextLine();

            if (ch == 1) {
                System.out.print("Enter name: ");
                String name = input.nextLine();
                queue.add(name);
```

```java
                System.out.println(name + " added to queue.");
            } else if (ch == 2) {
                System.out.print("Enter name: ");
                String name = input.nextLine();
                queue.addFirst(name);
                System.out.println(name + " added as priority.");
            } else if (ch == 3) {
                System.out.println("Queue:");
                for (String s : queue) {
                    System.out.println("- " + s);
                }
            } else if (ch == 4) {
                System.out.println("Exiting...");
                break;
            } else {
                System.out.println("Invalid choice.");
            }
        }
    }
}
```

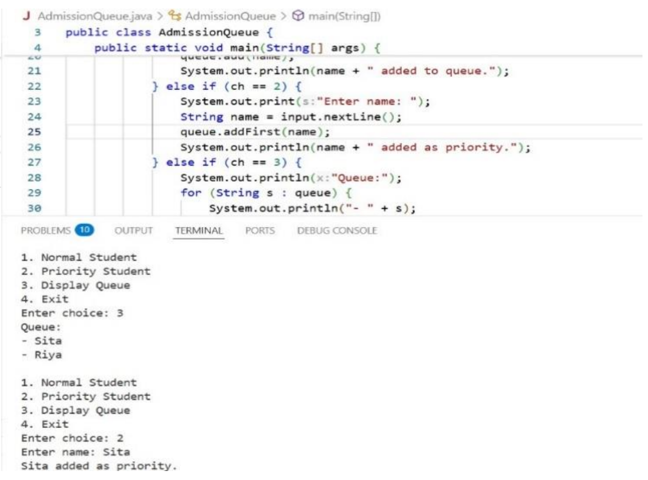


AdmissionQueue.java > AdmissionQueue > main(String[])

```java
public class AdmissionQueue {
    public static void main(String[] args) {
                queue.add(name);
                System.out.println(name + " added to queue.");
            } else if (ch == 2) {
                System.out.print(s:"Enter name: ");
                String name = input.nextLine();
                queue.addFirst(name);
                System.out.println(name + " added as priority.");
            } else if (ch == 3) {
```

PROBLEMS 10   OUTPUT   TERMINAL   PORTS   DEBUG CONSOLE

```
1. Normal Student
2. Priority Student
3. Display Queue
4. Exit
Enter choice: 1
Enter name: Riya
Riya added to queue.

1. Normal Student
2. Priority Student
3. Display Queue
4. Exit
Enter choice: 2
Enter name: Sita
Sita added as priority.
```



AdmissionQueue.java > AdmissionQueue > main(String[])

```java
public class AdmissionQueue {
    public static void main(String[] args) {
                queue.add(name);
                System.out.println(name + " added to queue.");
            } else if (ch == 2) {
                System.out.print(s:"Enter name: ");
                String name = input.nextLine();
                queue.addFirst(name);
                System.out.println(name + " added as priority.");
            } else if (ch == 3) {
                System.out.println(x:"Queue:");
                for (String s : queue) {
                    System.out.println("- " + s);
```

PROBLEMS 10   OUTPUT   TERMINAL   PORTS   DEBUG CONSOLE

```
1. Normal Student
2. Priority Student
3. Display Queue
4. Exit
Enter choice: 3
Queue:
- Sita
- Riya

1. Normal Student
2. Priority Student
3. Display Queue
4. Exit
Enter choice: 2
Enter name: Sita
Sita added as priority.
```