

----- Question 1-----

```
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 goBack() {
```

```
        if (!backStack.isEmpty()) {
```

```
            forwardStack.push(currentPage);
```

```
            currentPage = backStack.pop();
```

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

```
        } else {
```

```
            System.out.println("No pages in back history.");
```

```
        }
```

```
}
```

```
public void goForward() {
```

```
    if (!forwardStack.isEmpty()) {
```

```
        backStack.push(currentPage);
```

```
        currentPage = forwardStack.pop();
```

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

```
    } else {
```

```
        System.out.println("No pages in forward history.");
```

```
    }
```

```
}
```

```
public void showCurrentPage() {
```

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

```
}
```

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

```
    BrowserHistory browser = new BrowserHistory();
```

```
    Scanner scanner = new Scanner(System.in);
```

```
    while (true) {
```

```
        System.out.println("\n1. Visit New Page\n2. Go Back\n3. Go Forward\n4. Show Current Page\n5. Exit");
```

```
        int choice = scanner.nextInt();
```

```
        scanner.nextLine();
```

```
        switch (choice) {
```

case 1:

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

```
String url = scanner.nextLine();
```

```
browser.visit(url);
```

```
break;
```

case 2:

```
browser.goBack();
```

```
break;
```

case 3:

```
browser.goForward();
```

```
break;
```

case 4:

```
browser.showCurrentPage();
```

```
break;
```

case 5:

```
System.exit(0);
```

```
}
```

```
}
```

```
}
```

```
}
```



BrowserHistory.java		Run	Output
44	System.out.println("\n1. Visit New Page\n2. Go Back\n3. Go Forward\n4. Show Current Page\n5. Exit");		1. Visit New
45	int choice = scanner.nextInt();		2. Go Back
46	scanner.nextLine();		3. Go Forward
47	switch (choice) {		4. Show Curre
48	case 1:		5. Exit
49	System.out.print("Enter URL: ");		
50	String url = scanner.nextLine();		
51	browser.visit(url);		
52	break;		
53	case 2:		
54	browser.goBack();		
55	break;		
56	case 3:		
57	browser.goForward();		
58	break;		
59	case 4:		
60	browser.showCurrentPage();		

-----Question 2-----

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

```
import java.util.Queue;
```

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

```
public class PrintQueue {
```

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

```
    public void addJob(String job) {
```

```
        queue.offer(job);
```

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

```
}
```

```
public void processJob() {  
    if (!queue.isEmpty()) {  
        String job = queue.poll();  
        System.out.println("Processing job: " + job);  
    } else {  
        System.out.println("No jobs to process.");  
    }  
}
```

```
public void viewJobs() {  
    if (queue.isEmpty()) {  
        System.out.println("No pending jobs.");  
    } else {  
        System.out.println("Pending jobs: " + queue);  
    }  
}
```

```
public static void main(String[] args) {  
    PrintQueue printer = new PrintQueue();  
    Scanner scanner = new Scanner(System.in);  
    while (true) {  
        System.out.println("\n1. Add Job\n2. Process Job\n3. View Jobs\n4. Exit");
```

```
int 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.exit(0);


}

}




}
```

Programiz  
Online Java Compiler

Premium Coding  
Courses by Programiz



PrintQueue.java

 Share

Run

Output

```
31     PrintQueue printer = new PrintQueue();
32     Scanner scanner = new Scanner(System.in);
33     while (true) {
34         System.out.println("\n1. Add Job\n2. Process Job\n3.
           View Jobs\n4. Exit");
35         int choice = scanner.nextInt();
36         scanner.nextLine();
37         switch (choice) {
38             case 1:
39                 System.out.print("Enter job name: ");
40                 String job = scanner.nextLine();
41                 printer.addJob(job);
42                 break;
43             case 2:
44                 printer.processJob();
45                 break;
46             case 3:
47                 printer.viewJobs();
48                 break;
49             case 4:
50                 System.exit(0);
51         }
52     }
53 }
```

1. Add Job


2. Process Job

3. View Jobs

4. Exit

26°C

Search



-----Question 3-----

```
import java.util.*;
```

```
public class UndoRedoManager {
```

```
    Stack<String> undoStack;
```

```
    Stack<String> redoStack;
```

```
    public UndoRedoManager() {
```

```
        undoStack = new Stack<>();
```

```
        redoStack = new Stack<>();
```

```
    }
```

```
public void performAction(String action) {  
    undoStack.push(action);  
    redoStack.clear();  
    System.out.println("Performed: " + action);  
}
```

```
public void undo() {  
    if (undoStack.isEmpty()) {  
        System.out.println("Nothing to undo.");  
        return;  
    }  
    String action = undoStack.pop();  
    redoStack.push(action);  
    System.out.println("Undone: " + action);  
}
```

```
public void redo() {  
    if (redoStack.isEmpty()) {  
        System.out.println("Nothing to redo.");  
        return;  
    }  
    String action = redoStack.pop();  
    undoStack.push(action);  
}
```



```
        System.out.println("Redone: " + action);
    }
}
```

```
public void showHistory() {
    System.out.println("Undo Stack: " + undoStack);
    System.out.println("Redo Stack: " + redoStack);
}
```

```
public static void main(String[] args) {
    UndoRedoManager manager = new UndoRedoManager();
    Scanner scanner = new Scanner(System.in);
    while (true) {
        System.out.println("\n1. Perform Action\n2. Undo\n3. Redo\n4. Show History\n5. Exit");
        System.out.print("Enter choice: ");
        int choice = scanner.nextInt();
        scanner.nextLine();
        switch (choice) {
            case 1:
                System.out.print("Enter action: ");
                String action = scanner.nextLine();
                manager.performAction(action);
                break;
            case 2:
                manager.undo();
            case 3:
                manager.redo();
            case 4:
                manager.showHistory();
            case 5:
                return;
        }
    }
}
```

```
        break;

    case 3:

        manager.redo();

        break;

    case 4:

        manager.showHistory();

        break;

    case 5:

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

        return;

    default:

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

    }

}

}
```

Online Java Compiler

Courses by Programiz

UndoRedoManager.java

```
50 scanner.nextLine();
51 switch (choice) {
52     case 1:
53         System.out.println("Enter action: ");
54         String action = scanner.nextLine();
55         manager.performAction(action);
56         break;
57     case 2:
58         manager.undo();
59         break;
60     case 3:
61         manager.redo();
62         break;
63     case 4:
64         manager.showHistory();
65         break;
66     case 5:
67         System.out.println("Exiting...");
68         return;
69     default:
70         System.out.println("Invalid choice");
71 }
72 }
73 }
```

Output

```
1. Perform Action
2. Undo
3. Redo
4. Show History
5. Exit
Enter choice: 1
Enter action: []
Performed: []

1. Perform Action
2. Undo
3. Redo
4. Show History
5. Exit
Enter choice:
```

```
import java.util.*;
```

```
public class TicketBookingSystem {
```

```
    Queue<String> bookingQueue;
```

```
    public TicketBookingSystem() {
```

```
        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 one in the queue.");  
        return;  
    }  
    String name = bookingQueue.poll();  
    System.out.println(name + " has been served.");  
}
```

```
public void cancelTicket(String name) {  
    if (bookingQueue.remove(name)) {  
        System.out.println("Ticket cancelled for " + name);  
    } else {  
        System.out.println(name + " not found in queue.");  
    }  
}
```

```
public void displayQueue() {  
    if (bookingQueue.isEmpty()) {  
        System.out.println("Booking queue is empty.");  
    } else {  
        System.out.println("Current 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);  
    while (true) {  
        System.out.println("\n1. Add Person\n2. Serve Next\n3. Cancel Ticket\n4. Display  
Queue\n5. Exit");  
        System.out.print("Enter choice: ");  
        int choice = scanner.nextInt();  
        scanner.nextLine();  
        switch (choice) {  
            case 1:  
                System.out.print("Enter name: ");  
                String name = scanner.nextLine();  
                system.addPerson(name);  
                break;  
            case 2:  
                system.serveNext();  
                break;  
            case 3:  
                System.out.print("Enter name to cancel: ");  
                String cancelName = scanner.nextLine();
```

```

        system.cancelTicket(cancelName);

        break;

    case 4:

        system.displayQueue();

        break;

    case 5:

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

        return;

    default:

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

    }

}

}

}

}

```

The screenshot shows a Java IDE with the file `TicketBookingSystem.java` open. The code is a switch-case menu for a ticket booking system. The output window on the right shows the program's execution flow:

```

TicketBookingSystem.java
52      case 1:
53          System.out.print("Enter name: ");
54          String name = scanner.nextLine();
55          system.addPerson(name);
56          break;
57      case 2:
58          system.serveNext();
59          break;
60      case 3:
61          System.out.print("Enter name to cancel: ");
62          String cancelName = scanner.nextLine();
63          system.cancelTicket(cancelName);
64          break;
65      case 4:
66          system.displayQueue();
67          break;
68      case 5:
69          System.out.println("Exiting...");
70          return;
71      default:
72          System.out.println("Invalid choice");
73  }
74  }
75  }
76  }

Output
1. Add Person
2. Serve Next
3. Cancel Ticket
4. Display Queue
5. Exit
Enter choice: 1
Enter name: xyz
xyz added to the booking queue.

1. Add Person
2. Serve Next
3. Cancel Ticket
4. Display Queue
5. Exit
Enter choice: 2
xyz has been served.

1. Add Person
2. Serve Next
3. Cancel Ticket
4. Display Queue
5. Exit
Enter choice:

```

