1. Browser history using Stack.

Code:

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

public class BrowserHistory {

public static void main(String[] args) { Stack<String> backStack = new Stack<>(); Stack<String> forwardStack = new Stack<>(); String currentPage = "Home";

Scanner sc = new Scanner(System.in);

System.out.println("Browser History Started. Type: visit <url>, back, forward, current, exit"); while (true) {

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

String input = sc.nextLine(); String[] parts = input.split(" ");

if (parts[0].equalsIgnoreCase("visit")) { if (parts.length < 2) {

System.out.println("Please enter a valid URL."); continue;

}

backStack.push(currentPage); currentPage = parts[1]; forwardStack.clear();

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

}

else if (parts[0].equalsIgnoreCase("back")) { if (backStack.isEmpty()) {

System.out.println("No pages to go back.");

} else {

forwardStack.push(currentPage);

currentPage = backStack.pop(); System.out.println("Back to: " + currentPage);

}

}

else if (parts[0].equalsIgnoreCase("forward")) { if (forwardStack.isEmpty()) {

System.out.println("No pages to go forward.");

} else {

backStack.push(currentPage); currentPage = forwardStack.pop();

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

}

}

else if (parts[0].equalsIgnoreCase("current")) { System.out.println("Current Page: " + currentPage);

}

else if (parts[0].equalsIgnoreCase("exit")) { System.out.println("Exiting Browser History."); break;

}

else {

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

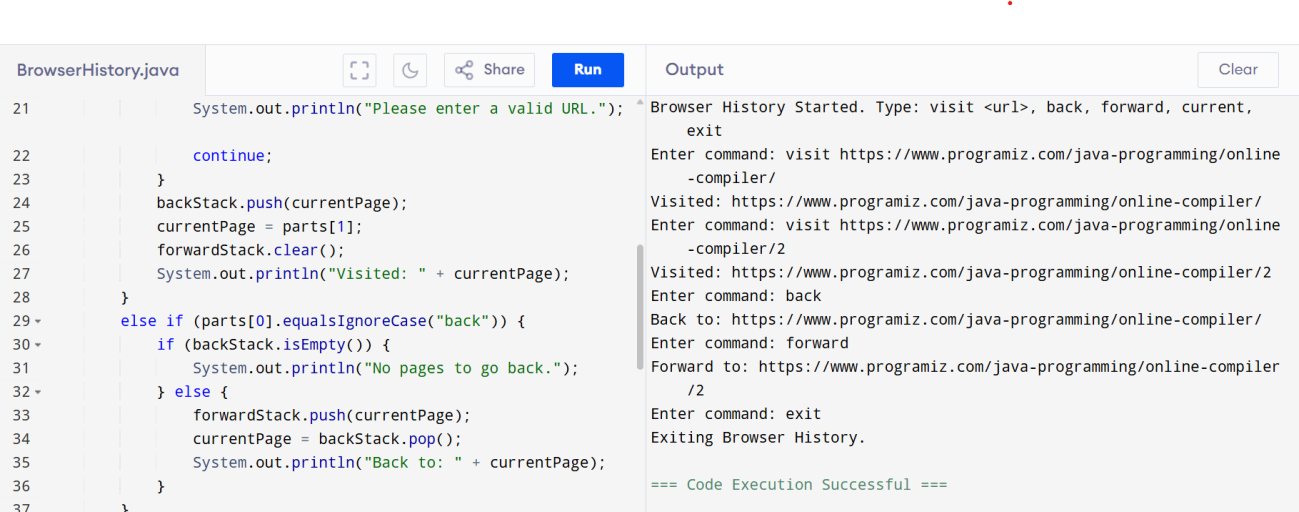
}

}

sc.close();

}

}Output:



1. Printing Queue using LinkedList

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

public class PrintQueue {

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

System.out.println("Print Queue Simulation"); System.out.println("Commands: add <job>, process, view, exit");

while (true) {

System.out.print("Enter command: "); String input = sc.nextLine();

String[] parts = input.split(" ", 2);

if (parts[0].equalsIgnoreCase("add")) { if (parts.length < 2) {

System.out.println("Please enter a job name."); continue;

}

printQueue.add(parts[1]);

System.out.println("Job added: " + parts[1]);

}

else if (parts[0].equalsIgnoreCase("process")) { if (printQueue.isEmpty()) {

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

} else {

String job = printQueue.poll(); System.out.println("Processing job: " + job);

}

}

else if (parts[0].equalsIgnoreCase("view")) { if (printQueue.isEmpty()) {

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

} else {

System.out.println("Pending print jobs:"); for (String job : printQueue) {

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

}

}

}

else if (parts[0].equalsIgnoreCase("exit")) { System.out.println("Exiting Print Queue."); break;

}

else {

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

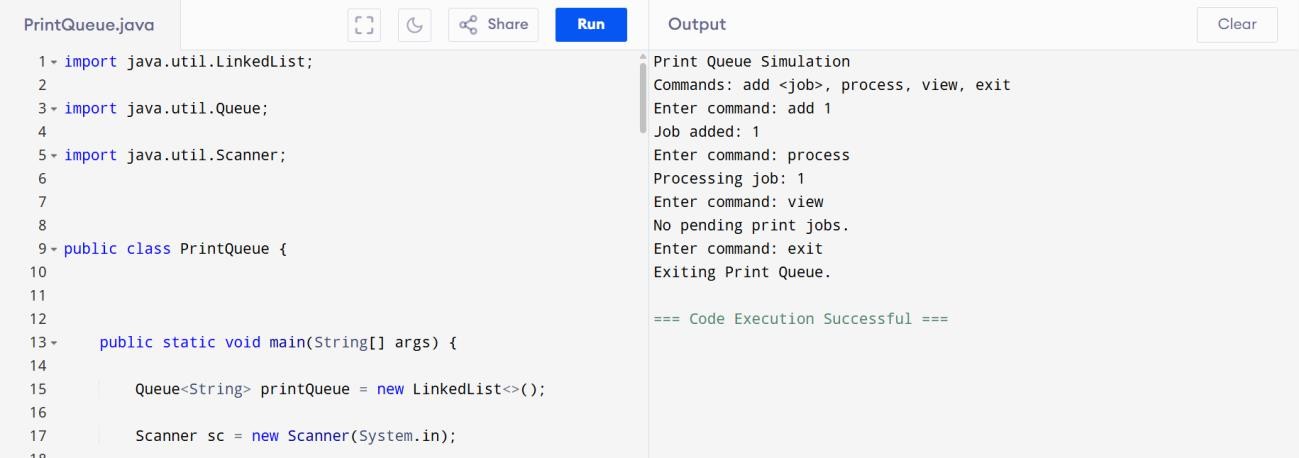
}

}

sc.close();

}

}



1. Hospital Bed Management System using LinkedList

import java.util.LinkedList; import java.util.Scanner; public class HospitalBedManagement {

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

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

System.out.println("Commands: assign <patient\_name>, discharge <patient\_name>, display, exit");

while (true) {

System.out.print("Enter command: "); String input = sc.nextLine();

String[] parts = input.split(" ", 2);

if (parts[0].equalsIgnoreCase("assign")) { if (parts.length < 2) {

System.out.println("Please enter patient name."); continue;

}

beds.add(parts[1]);

System.out.println("Bed assigned to: " + parts[1]);

}

else if (parts[0].equalsIgnoreCase("discharge")) { if (parts.length < 2) {

System.out.println("Please enter patient name to discharge."); continue;

}

String patient = parts[1];

if (beds.remove(patient)) { System.out.println("Patient discharged: " + patient);

} else {

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

}

}

else if (parts[0].equalsIgnoreCase("display")) { if (beds.isEmpty()) {

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

} else {

System.out.println("Current Occupancy:"); for (int i = 0; i < beds.size(); i++) {

System.out.println("Bed " + (i + 1) + ": " + beds.get(i));

}

}

}

else if (parts[0].equalsIgnoreCase("exit")) { System.out.println("Exiting Hospital Bed Management."); break;

}

else {

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

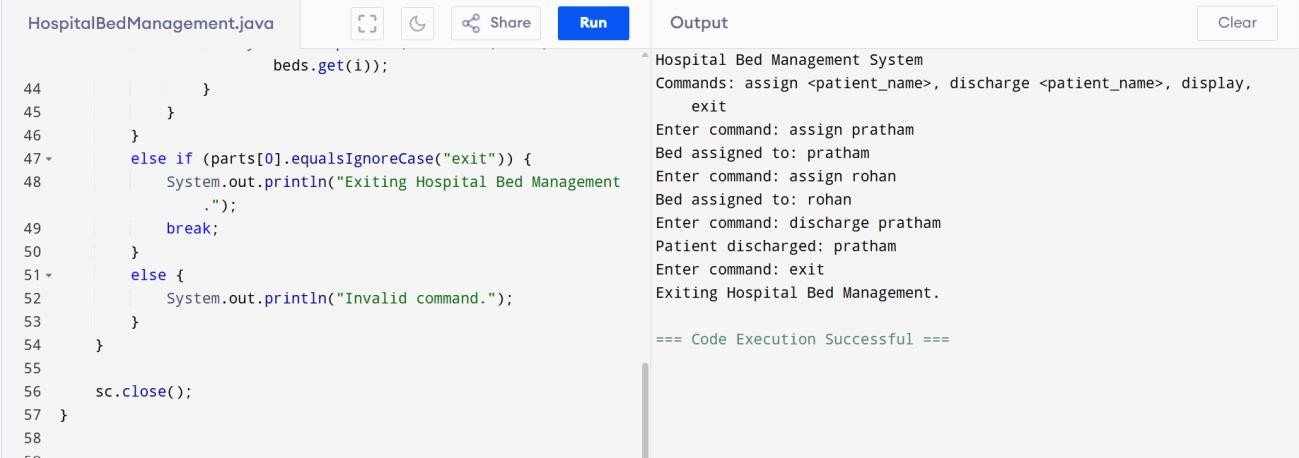
}

}

sc.close();

}

}



1. Undo-Redo Function using Stack import java.util.Scanner;

import java.util.Stack;

public class UndoRedo {

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

System.out.println("Undo-Redo System (Commands: action <task>, undo, redo, show, exit)");

while (true) {

System.out.print("Enter command: "); String input = sc.nextLine();

String[] parts = input.split(" ", 2);

if (parts[0].equalsIgnoreCase("action")) { if (parts.length < 2) {

System.out.println("Please enter a task."); continue;

}

undoStack.push(parts[1]); redoStack.clear();

System.out.println("Action done: " + parts[1]);

} else if (parts[0].equalsIgnoreCase("undo")) { if (undoStack.isEmpty()) {

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

} else {

String lastAction = undoStack.pop(); redoStack.push(lastAction); System.out.println("Undo: " + lastAction);

}

} else if (parts[0].equalsIgnoreCase("redo")) { if (redoStack.isEmpty()) {

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

} else {

String redoAction = redoStack.pop(); undoStack.push(redoAction); System.out.println("Redo: " + redoAction);

}

} else if (parts[0].equalsIgnoreCase("show")) { System.out.println("Current Actions: " + undoStack);

} else if (parts[0].equalsIgnoreCase("exit")) { break;

} else {

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

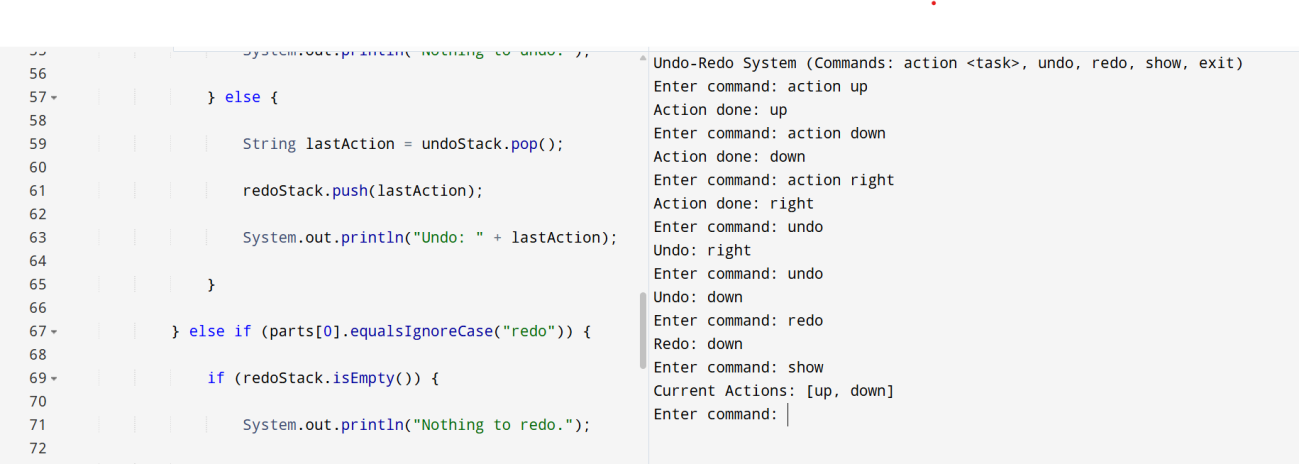
}

}

sc.close();

}

}



1. Ticket Booking System using queue

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

public class TicketBooking {

public static void main(String[] args) {

Queue<String> bookingQueue = new LinkedList<>(); Scanner sc = new Scanner(System.in);

System.out.println("Ticket Booking System (Commands: add <name>, serve, cancel

<name>, view, exit)");

while (true) {

System.out.print("Enter command: "); String input = sc.nextLine();

String[] parts = input.split(" ", 2);

if (parts[0].equalsIgnoreCase("add")) { if (parts.length < 2) {

System.out.println("Please enter a name."); continue;

}

bookingQueue.add(parts[1]);

System.out.println(parts[1] + " added to booking queue.");

} else if (parts[0].equalsIgnoreCase("serve")) { if (bookingQueue.isEmpty()) {

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

} else {

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

}

} else if (parts[0].equalsIgnoreCase("cancel")) { if (parts.length < 2) {

System.out.println("Please enter name to cancel."); continue;

}

if (bookingQueue.remove(parts[1])) { System.out.println("Cancelled ticket for: " + parts[1]);

} else {

System.out.println("Person not found in queue.");

}

} else if (parts[0].equalsIgnoreCase("view")) { System.out.println("Current Queue: " + bookingQueue);

} else if (parts[0].equalsIgnoreCase("exit")) {

System.out.println("Exiting booking system."); break;

} else {

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

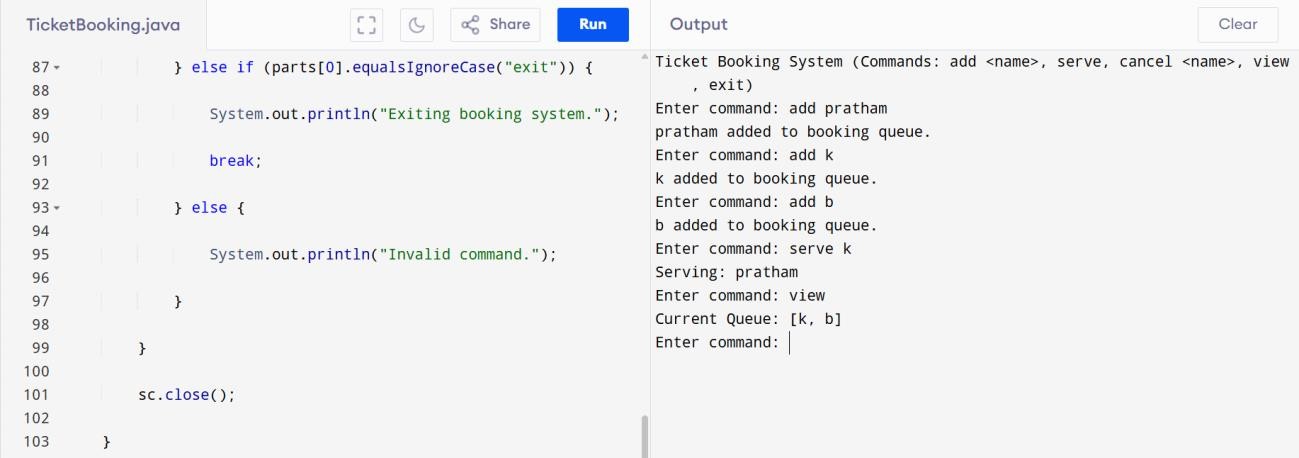
}

}

sc.close();

}

}



1. Car Wash Service using Queue import java.util.LinkedList;

import java.util.Scanner; public class CarWashQueue {

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

System.out.println("Car Wash Queue System");

System.out.println("Commands: add <car\_name>, vip <car\_name>, wash, view, exit");

while (true) {

System.out.print("Enter command: "); String input = sc.nextLine();

String[] parts = input.split(" ", 2);

if (parts[0].equalsIgnoreCase("add")) { if (parts.length < 2) {

System.out.println("Please enter car name."); continue;

}

carQueue.addLast(parts[1]); System.out.println(parts[1] + " added to the queue.");

}

else if (parts[0].equalsIgnoreCase("vip")) { if (parts.length < 2) {

System.out.println("Please enter car name."); continue;

}

carQueue.addFirst(parts[1]);

System.out.println(parts[1] + " added to the front as VIP.");

}

else if (parts[0].equalsIgnoreCase("wash")) { if (carQueue.isEmpty()) {

System.out.println("No cars to wash.");

} else {

String car = carQueue.removeFirst(); System.out.println("Washed car: " + car);

}

}

else if (parts[0].equalsIgnoreCase("view")) { System.out.println("Cars in queue: " + carQueue);

}

else if (parts[0].equalsIgnoreCase("exit")) { System.out.println("Closing Car Wash Queue."); break;

}

else {

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

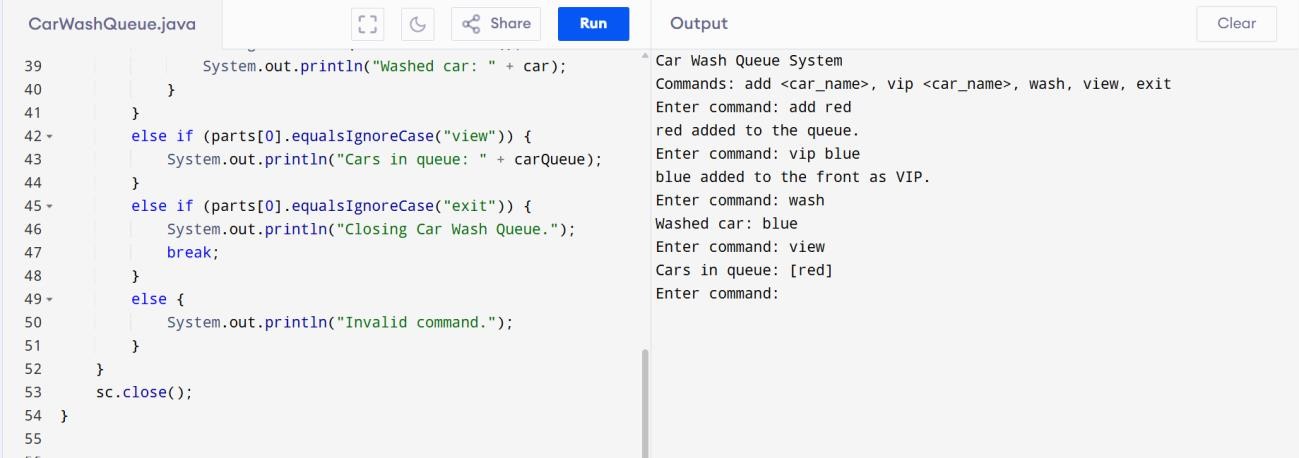
}

}

sc.close();

}

}



1. Library Book Stack import java.util.Scanner; import java.util.Stack;

public class LibraryBookStack {

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

System.out.println("Library Book Stack"); System.out.println("Commands: add <book\_name>, remove, peek, exit");

while (true) {

System.out.print("Enter command: "); String input = sc.nextLine();

String[] parts = input.split(" ", 2);

if (parts[0].equalsIgnoreCase("add")) { if (parts.length < 2) {

System.out.println("Please enter book name."); continue;

}

bookStack.push(parts[1]); System.out.println(parts[1] + " added to stack.");

}

else if (parts[0].equalsIgnoreCase("remove")) { if (bookStack.isEmpty()) {

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

} else {

String book = bookStack.pop(); System.out.println("Removed book: " + book);

}

}

else if (parts[0].equalsIgnoreCase("peek")) { if (bookStack.isEmpty()) {

System.out.println("No books in stack.");

} else {

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

}

}

else if (parts[0].equalsIgnoreCase("exit")) { System.out.println("Exiting Library System."); break;

}

else {

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

}

}

sc.close();

}

}

