**1. Browser History**

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;

public void visit(String page) {

if (currentPage != null) {

backStack.push(currentPage);

}

currentPage = page;

forwardStack.clear();

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

}

public void back() {

if (!backStack.isEmpty()) {

forwardStack.push(currentPage);

currentPage = backStack.pop();

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

} else {

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

}

}

public void forward() {

if (!forwardStack.isEmpty()) {

backStack.push(currentPage);

currentPage = forwardStack.pop();

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

} else {

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

}

}

public void current() {

if (currentPage != null) {

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

} else {

System.out.println("No page visited yet.");

}

}

public void showHistory() {

System.out.println("Back Stack: " + backStack);

System.out.println("Forward Stack: " + forwardStack);

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

}

public static void main(String[] args) {

BrowserHistory browser = new BrowserHistory();

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n=== Browser History 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. Show History");

System.out.println("6. Exit");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.print("Enter URL to visit: ");

String url = sc.nextLine();

browser.visit(url);

break;

case 2:

browser.back();

break;

case 3:

browser.forward();

break;

case 4:

browser.current();

break;

case 5:

browser.showHistory();

break;

case 6:

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

break;

default:

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

}

} while (choice != 6);

sc.close();

}

}

A screenshot of a computer program

AI-generated content may be incorrect.

**2. Print Queue**

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 print job: " + job);

}

public void processJob() {

if (!queue.isEmpty()) {

System.out.println("Processing: " + queue.poll());

} else {

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

}

}

public void viewJobs() {

if (!queue.isEmpty()) {

System.out.println("Pending jobs: " + queue);

} else {

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

}

}

public static void main(String[] args) {

PrintQueue printer = new PrintQueue();

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- Print Queue Menu ---");

System.out.println("1. Add new 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 = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.print("Enter job name: ");

String job = sc.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.");

}

} while (choice != 4);

sc.close();

}

}

A screenshot of a computer screen

AI-generated content may be incorrect.

**3. Hospital Bed Management**

import java.util.LinkedList;

import java.util.Scanner;

public class HospitalBedManagement {

private LinkedList<String> beds = new LinkedList<>();

public void assignBed(String patient) {

beds.add(patient);

System.out.println("Assigned bed to: " + patient);

}

public void discharge(String patient) {

if (beds.remove(patient)) {

System.out.println("Discharged: " + patient);

} else {

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

}

}

public void viewBeds() {

if (!beds.isEmpty()) {

System.out.println("Current Occupancy: " + beds);

} else {

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

}

}

public static void main(String[] args) {

HospitalBedManagement hospital = new HospitalBedManagement();

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- Hospital Bed Management Menu ---");

System.out.println("1. Assign bed to new patient");

System.out.println("2. Discharge patient");

System.out.println("3. View current occupancy");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.print("Enter patient name: ");

String patient = sc.nextLine();

hospital.assignBed(patient);

break;

case 2:

System.out.print("Enter patient name to discharge: ");

String name = sc.nextLine();

hospital.discharge(name);

break;

case 3:

hospital.viewBeds();

break;

case 4:

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

break;

default:

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

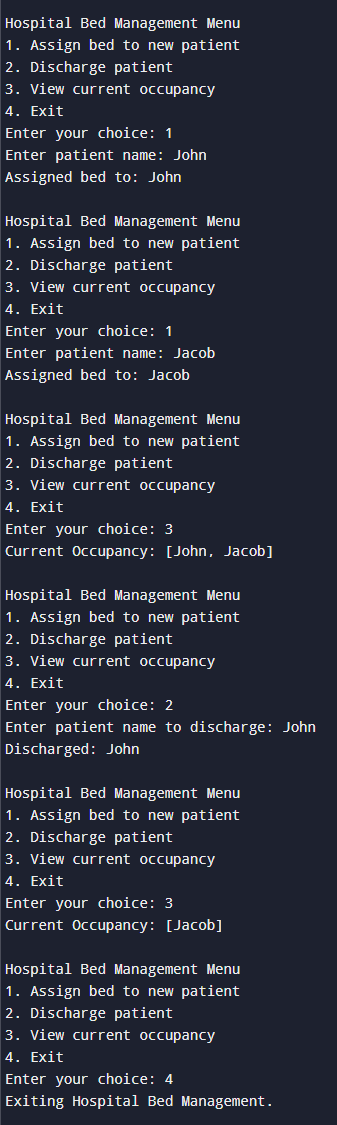
}

} while (choice != 4);

sc.close();

}

}



**4. Undo-Redo Function**

import java.util.Scanner;

import java.util.Stack;

public class UndoRedoFunction {

private Stack<String> undoStack = new Stack<>();

private Stack<String> redoStack = new Stack<>();

public void performAction(String action) {

undoStack.push(action);

redoStack.clear();

System.out.println("Performed: " + action);

}

public void undo() {

if (!undoStack.isEmpty()) {

String action = undoStack.pop();

redoStack.push(action);

System.out.println("Undo: " + action);

} else {

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

}

}

public void redo() {

if (!redoStack.isEmpty()) {

String action = redoStack.pop();

undoStack.push(action);

System.out.println("Redo: " + action);

} else {

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

}

}

public void showStacks() {

System.out.println("Undo Stack: " + undoStack);

System.out.println("Redo Stack: " + redoStack);

}

public static void main(String[] args) {

UndoRedoFunction ur = new UndoRedoFunction();

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- Undo/Redo Menu ---");

System.out.println("1. Perform Action");

System.out.println("2. Undo");

System.out.println("3. Redo");

System.out.println("4. View Stacks");

System.out.println("5. Exit");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

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

String action = sc.nextLine();

ur.performAction(action);

break;

case 2:

ur.undo();

break;

case 3:

ur.redo();

break;

case 4:

ur.showStacks();

break;

case 5:

System.out.println("Exiting Undo/Redo Function.");

break;

default:

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

}

} while (choice != 5);

sc.close();

}

}

A screen shot of a computer

AI-generated content may be incorrect.

**5. Ticket Booking System**

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("Added to booking queue: " + name);

}

public void serveNext() {

if (!bookingQueue.isEmpty()) {

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

} else {

System.out.println("No person in the queue.");

}

}

public void cancelTicket(String name) {

if (bookingQueue.remove(name)) {

System.out.println("Cancelled ticket for: " + name);

} else {

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

}

}

public void viewQueue() {

System.out.println("Current Booking Queue: " + bookingQueue);

}

public static void main(String[] args) {

TicketBookingSystem system = new TicketBookingSystem();

Scanner sc = new Scanner(System.in);

int choice;

do {

System.out.println("\n--- Ticket Booking System Menu ---");

System.out.println("1. Add person to queue");

System.out.println("2. Serve next person");

System.out.println("3. Cancel ticket");

System.out.println("4. View queue");

System.out.println("5. Exit");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

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

String name = sc.nextLine();

system.addPerson(name);

break;

case 2:

system.serveNext();

break;

case 3:

System.out.print("Enter name to cancel ticket: ");

String cancelName = sc.nextLine();

system.cancelTicket(cancelName);

break;

case 4:

system.viewQueue();

break;

case 5:

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

break;

default:

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

}

} while (choice != 5);

sc.close();

}

}

A screen shot of a computer program

AI-generated content may be incorrect.

**6. Car Wash Service**