Data Structure and Algorithms (JAVA)



**Assignment #02**

**Submitted by:** *L1F23BSSE0391 – Abdullah Maqbool*

**Section:** *P4*

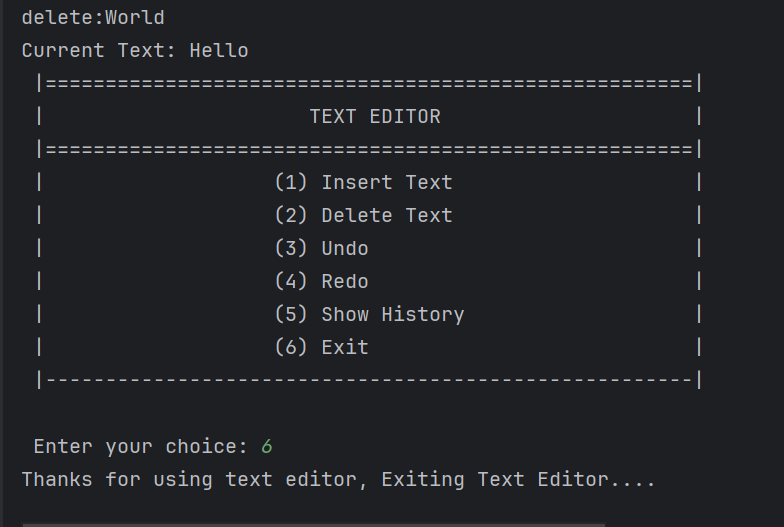
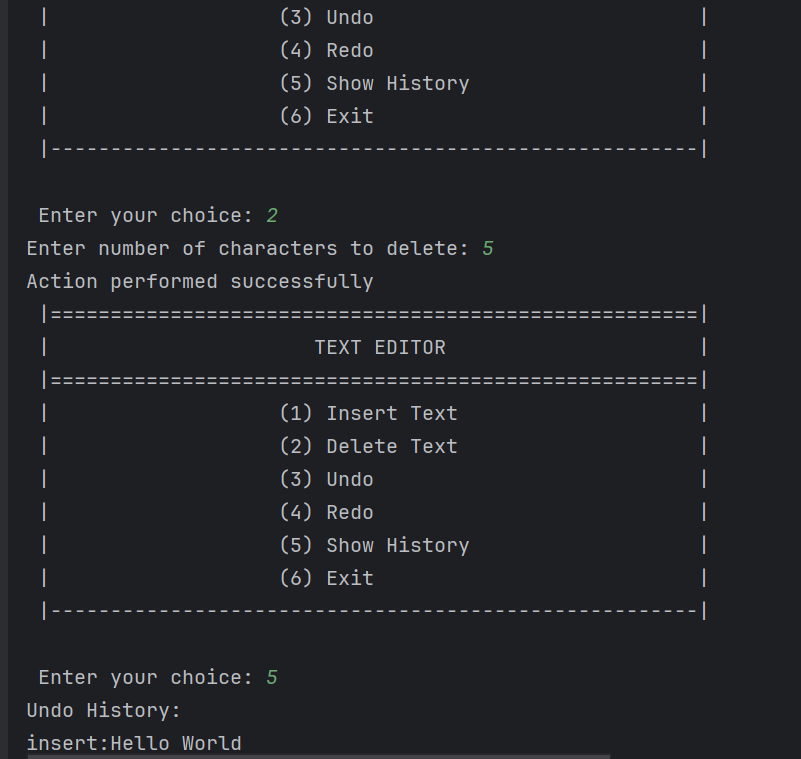
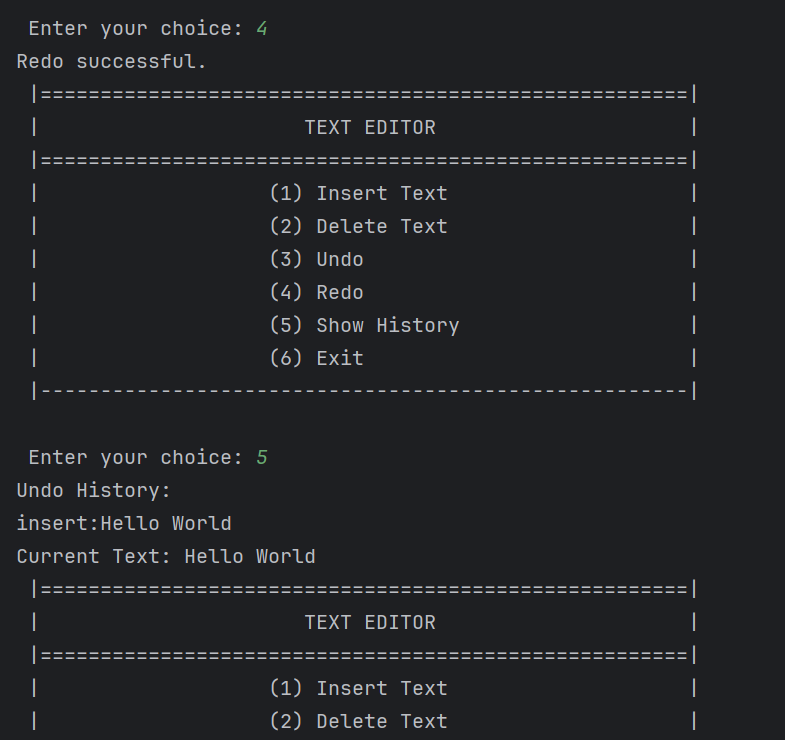
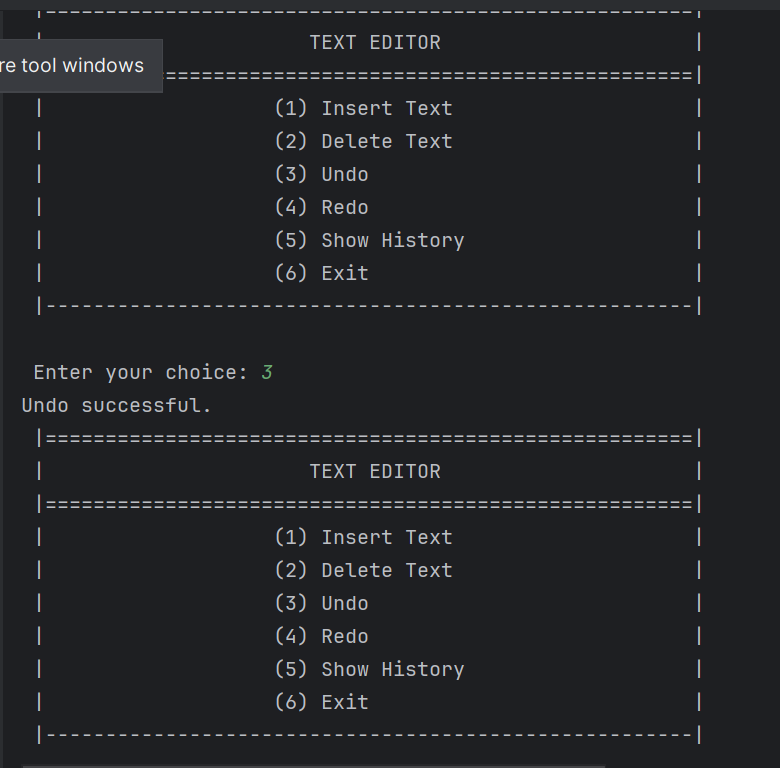
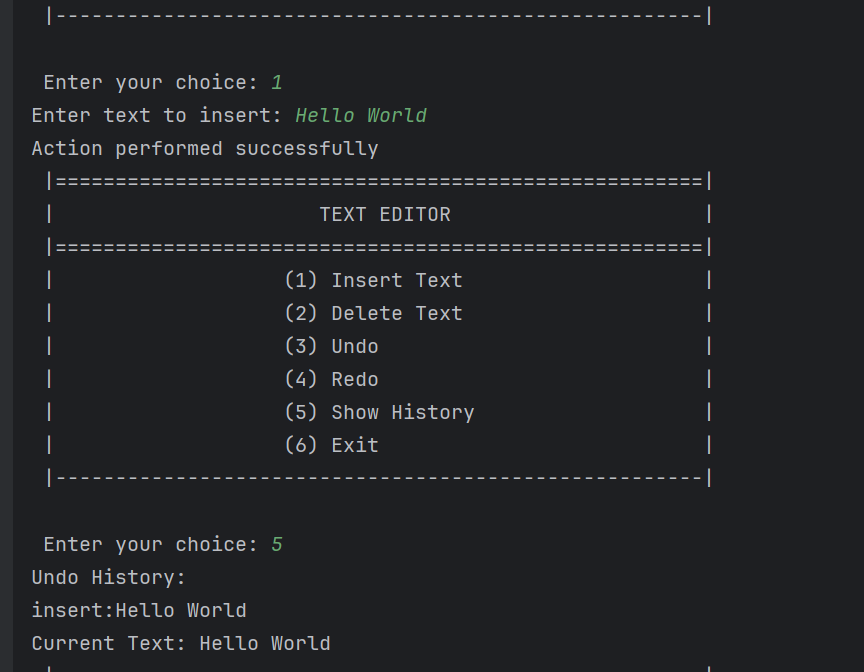
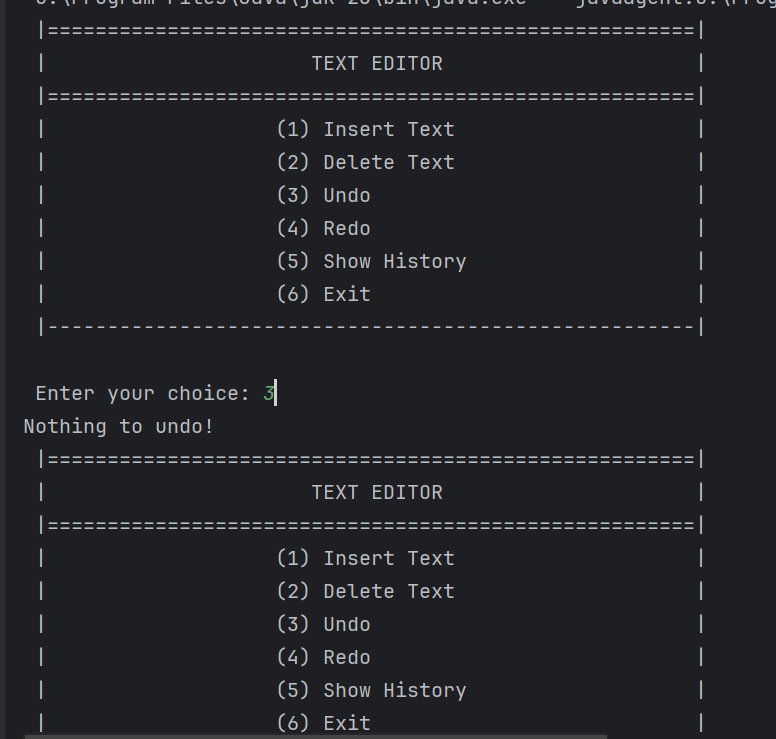
**Submitted to:** *Ma’am Afifa Hameed*

# Scenario 1: Text Editor

## CODE

// L1F23BSSE0391 - Abdullah Maqbool - P4  
  
  
import java.util.Scanner;  
import java.util.Stack;  
  
class TextEditor {  
 Stack<String> undoStack = new Stack<>();  
 Stack<String> redoStack = new Stack<>();  
 StringBuilder currentText = new StringBuilder();  
  
 public void performAction(String type, String text) {  
 if (type.equalsIgnoreCase("insert")) {  
 currentText.append(text);  
 undoStack.push("insert:" + text);  
 } else if (type.equalsIgnoreCase("delete")) {  
 int len = Integer.*parseInt*(text); // converting text(num) to integer  
 if (len > currentText.length()) {  
 len = currentText.length();  
 }  
 String deleted = currentText.substring(currentText.length() - len);  
 currentText.delete(currentText.length() - len, currentText.length());  
 undoStack.push("delete:" + deleted);  
 } else {  
 System.*out*.println("Unknown action type.");  
 return;  
 }  
 redoStack.clear();  
 System.*out*.println("Action performed successfully");  
 }  
  
 public void undo() {  
 if (undoStack.isEmpty()) {  
 System.*out*.println("Nothing to undo!");  
 return;  
 }  
 String action = undoStack.pop();  
 redoStack.push(action);  
  
 String[] parts = action.split(":");  
 if (parts[0].equals("insert")) {  
 currentText.delete(currentText.length() - parts[1].length(), currentText.length());  
 } else if (parts[0].equals("delete")) {  
 currentText.append(parts[1]);  
 }  
 System.*out*.println("Undo successful.");  
 }  
  
 public void redo() {  
 if (redoStack.isEmpty()) {  
 System.*out*.println("Nothing to redo.");  
 return;  
 }  
 String action = redoStack.pop();  
 undoStack.push(action);  
  
 String[] parts = action.split(":");  
 if (parts[0].equals("insert")) {  
 currentText.append(parts[1]);  
 } else if (parts[0].equals("delete")) {  
 currentText.delete(currentText.length() - parts[1].length(), currentText.length());  
 }  
 System.*out*.println("Redo successful.");  
 }  
  
 public void showHistory() {  
 System.*out*.println("Undo History:");  
 for (String act : undoStack) {  
 System.*out*.println(act);  
 }  
 System.*out*.println("Current Text: " + currentText.toString());  
 }  
}  
  
public class Scenario01 {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 TextEditor TE = new TextEditor();  
 while (true) {  
 System.*out*.println(" |======================================================|");  
 System.*out*.println(" | TEXT EDITOR |");  
 System.*out*.println(" |======================================================|");  
 System.*out*.println(" | (1) Insert Text |");  
 System.*out*.println(" | (2) Delete Text |");  
 System.*out*.println(" | (3) Undo |");  
 System.*out*.println(" | (4) Redo |");  
 System.*out*.println(" | (5) Show History |");  
 System.*out*.println(" | (6) Exit |");  
 System.*out*.println(" |----------------------------------------------------------------------------------------|");  
  
 System.*out*.print("\n Enter your choice: ");  
 int choice = sc.nextInt();  
 sc.nextLine();  
  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter text to insert: ");  
 String text = sc.nextLine();  
 TE.performAction("insert", text);  
 break;  
 case 2:  
 System.*out*.print("Enter number of characters to delete: ");  
 String num = sc.nextLine();  
 TE.performAction("delete", num);  
 break;  
 case 3:  
 TE.undo();  
 break;  
 case 4:  
 TE.redo();  
 break;  
 case 5:  
 TE.showHistory();  
 break;  
 case 6:  
 System.*out*.println("Thanks for using text editor, Exiting Text Editor....");  
 return;  
 default:  
 System.*out*.println("Invalid choice.Pls Try Again");  
 }  
 }  
 }  
}

## Output Screenshot

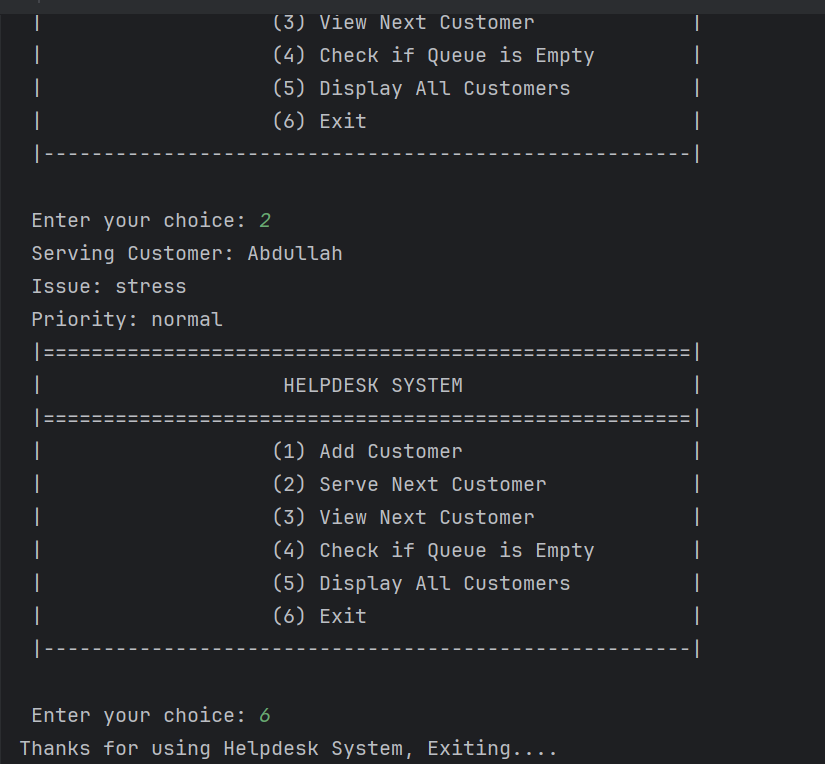
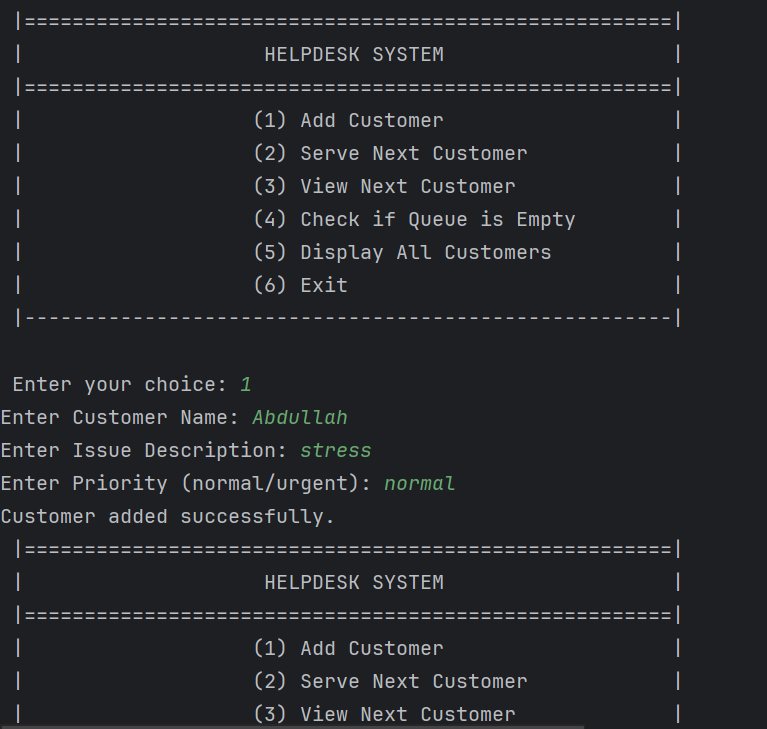


# Scenario 2: Helpdesk System

## CODE

// L1F23BSSE0391 - Abdullah Maqbool - P4  
  
import java.util.LinkedList;  
import java.util.Scanner;  
  
class Customer {  
 String name;  
 String issue;  
 String priority;  
  
 public Customer(String name, String issue, String priority) {  
 this.name = name;  
 this.issue = issue;  
 this.priority = priority;  
 }  
}  
  
class HelpDesk {  
 LinkedList<Customer> queue = new LinkedList<>();  
  
 public void enqueue(String name, String issue, String priority) {  
 Customer newCustomer = new Customer(name, issue, priority);  
 if (priority.equalsIgnoreCase("urgent")) {  
 queue.addFirst(newCustomer);  
 } else {  
 queue.addLast(newCustomer);  
 }  
 System.*out*.println("Customer added successfully.");  
 }  
  
 public void dequeue() {  
 if (queue.isEmpty()) {  
 System.*out*.println("No customers in the queue to serve.");  
 return;  
 }  
 Customer servedCustomer = queue.removeFirst();  
 System.*out*.println(" Serving Customer: " + servedCustomer.name + " \n Issue: " + servedCustomer.issue + " \n Priority: " + servedCustomer.priority);  
 }  
  
 public void peek() {  
 if (queue.isEmpty()) {  
 System.*out*.println("Queue is empty.");  
 return;  
 }  
 Customer nextCustomer = queue.peekFirst();  
 System.*out*.println(" Next Customer: " + nextCustomer.name + " \n Issue: " + nextCustomer.issue + " \n Priority: " + nextCustomer.priority);  
 }  
  
 public void isEmpty() {  
 if (queue.isEmpty()) {  
 System.*out*.println("The queue is empty.");  
 } else {  
 System.*out*.println("There are customers waiting in the queue.");  
 }  
 }  
  
 public void displayQueue() {  
 if (queue.isEmpty()) {  
 System.*out*.println("No customers are currently in the queue.");  
 return;  
 }  
 System.*out*.println("Customers currently in queue:");  
 int count = 1;  
 for (Customer c : queue) {  
 System.*out*.println(count++ + ") " + c.name + " ~ Issue: " + c.issue + " ~ Priority: " + c.priority);  
 }  
 }  
}  
  
public class Scenario02 {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 HelpDesk HD = new HelpDesk();  
  
 while (true) {  
 System.*out*.println(" |======================================================|");  
 System.*out*.println(" | HELPDESK SYSTEM |");  
 System.*out*.println(" |======================================================|");  
 System.*out*.println(" | (1) Add Customer |");  
 System.*out*.println(" | (2) Serve Next Customer |");  
 System.*out*.println(" | (3) View Next Customer |");  
 System.*out*.println(" | (4) Check if Queue is Empty |");  
 System.*out*.println(" | (5) Display All Customers |");  
 System.*out*.println(" | (6) Exit |");  
 System.*out*.println(" |----------------------------------------------------------------------------------------|");  
  
 System.*out*.print("\n Enter your choice: ");  
 int choice = sc.nextInt();  
 sc.nextLine();  
  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter Customer Name: ");  
 String name = sc.nextLine();  
 System.*out*.print("Enter Issue Description: ");  
 String issue = sc.nextLine();  
 System.*out*.print("Enter Priority (normal/urgent): ");  
 String priority = sc.nextLine();  
 HD.enqueue(name, issue, priority);  
 break;  
 case 2:  
 HD.dequeue();  
 break;  
 case 3:  
 HD.peek();  
 break;  
 case 4:  
 HD.isEmpty();  
 break;  
 case 5:  
 HD.displayQueue();  
 break;  
 case 6:  
 System.*out*.println("Thanks for using Helpdesk System, Exiting....");  
 return;  
 default:  
 System.*out*.println("Invalid choice,Pls try again.");  
 }  
 }  
 }  
}

## Output Screenshot

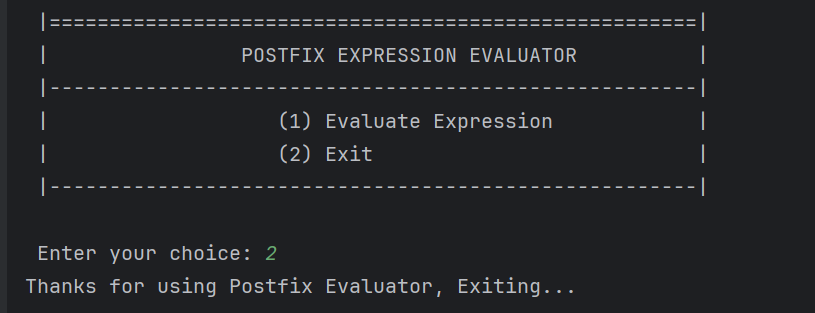
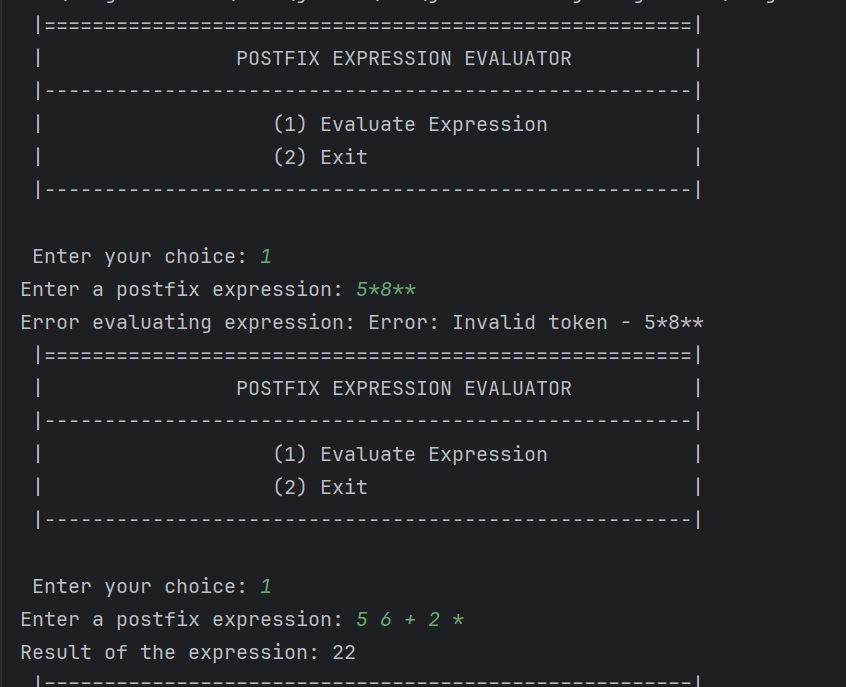


# Scenario 3: Postfix Evaluator

## CODE

// L1F23BSSE0391 - Abdullah Maqbool - P4  
  
  
import java.util.Scanner;  
import java.util.Stack;  
  
class PostfixEvaluator {  
 Stack<Integer> stack = new Stack<>();  
  
 public int evaluate(String expression) {  
 String[] tokens = expression.trim().split(" ");  
  
 for (String token : tokens) {  
 if (isNumber(token)) {  
 stack.push(Integer.*parseInt*(token));  
 } else if (isOperator(token)) {  
 if (stack.size() < 2) {  
 throw new IllegalArgumentException("Error: Not enough operands.");  
 }  
 int b = stack.pop();  
 int a = stack.pop();  
 int result = applyOperation(a, b, token);  
 stack.push(result);  
 } else {  
 throw new IllegalArgumentException("Error: Invalid token - " + token);  
 }  
 }  
  
 if (stack.size() != 1) {  
 throw new IllegalStateException("Error: Invalid postfix expression.");  
 }  
  
 return stack.pop();  
 }  
  
 private boolean isNumber(String token) {  
 try {  
 Integer.*parseInt*(token);  
 return true;  
 } catch (Exception e) {  
 return false;  
 }  
 }  
  
 private boolean isOperator(String token) {  
 return token.equals("+") || token.equals("-") || token.equals("\*") || token.equals("/");  
 }  
  
 private int applyOperation(int a, int b, String operator) {  
 switch (operator) {  
 case "+": return a + b;  
 case "-": return a - b;  
 case "\*": return a \* b;  
 case "/":  
 if (b == 0) {  
 throw new ArithmeticException("Division by zero not possible.");  
 }  
 return a / b;  
 default:  
 throw new IllegalArgumentException("Unknown operator: " + operator);  
 }  
 }  
}  
public class Scenario03 {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 PostfixEvaluator PE = new PostfixEvaluator();  
  
 while (true) {  
 System.*out*.println(" |======================================================|");  
 System.*out*.println(" | POSTFIX EXPRESSION EVALUATOR |");  
 System.*out*.println(" |----------------------------------------------------------------------------------------|");  
 System.*out*.println(" | (1) Evaluate Expression |");  
 System.*out*.println(" | (2) Exit |");  
 System.*out*.println(" |---------------------------------------------------------------------------------------|");  
  
 System.*out*.print("\n Enter your choice: ");  
 int choice = sc.nextInt();  
 sc.nextLine();  
  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter a postfix expression: ");  
 String expression = sc.nextLine();  
 try {  
 int result = PE.evaluate(expression);  
 System.*out*.println("Result of the expression: " + result);  
 } catch (Exception e) {  
 System.*out*.println("Error evaluating expression: " + e.getMessage());  
 }  
 break;  
 case 2:  
 System.*out*.println("Thanks for using Postfix Evaluator, Exiting...");  
 return;  
 default:  
 System.*out*.println("Invalid choice, Pls try again.");  
 }  
 }  
 }  
}

## Output Screenshot

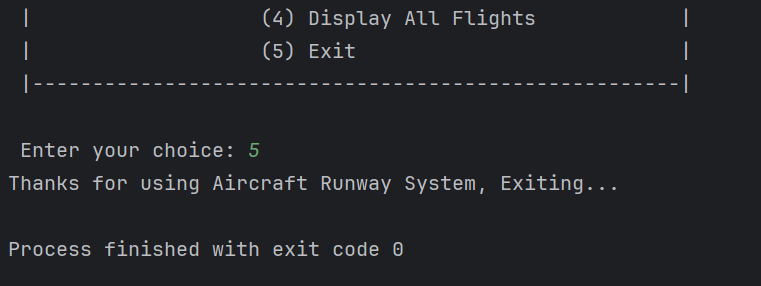
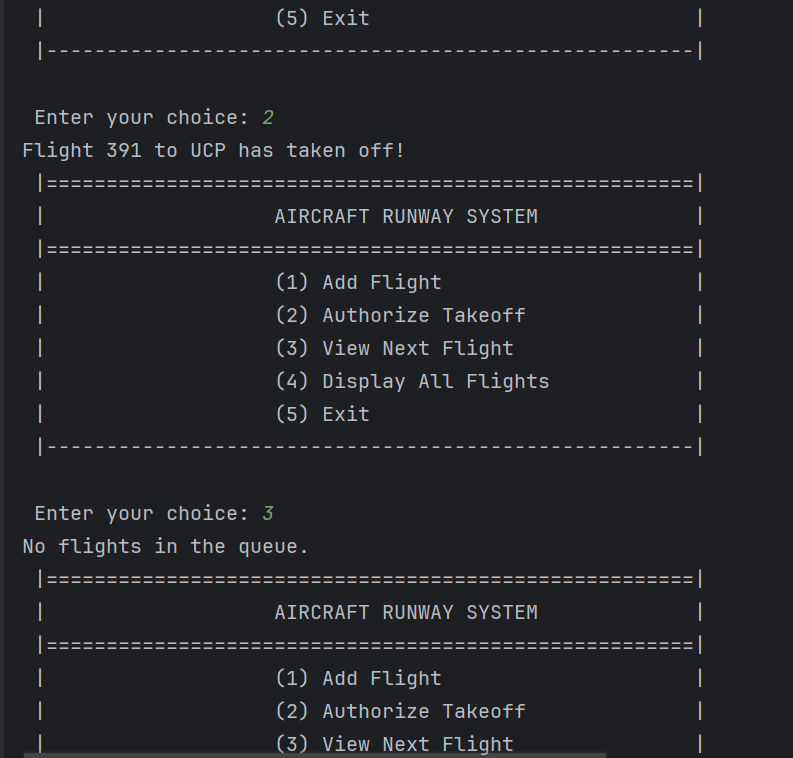
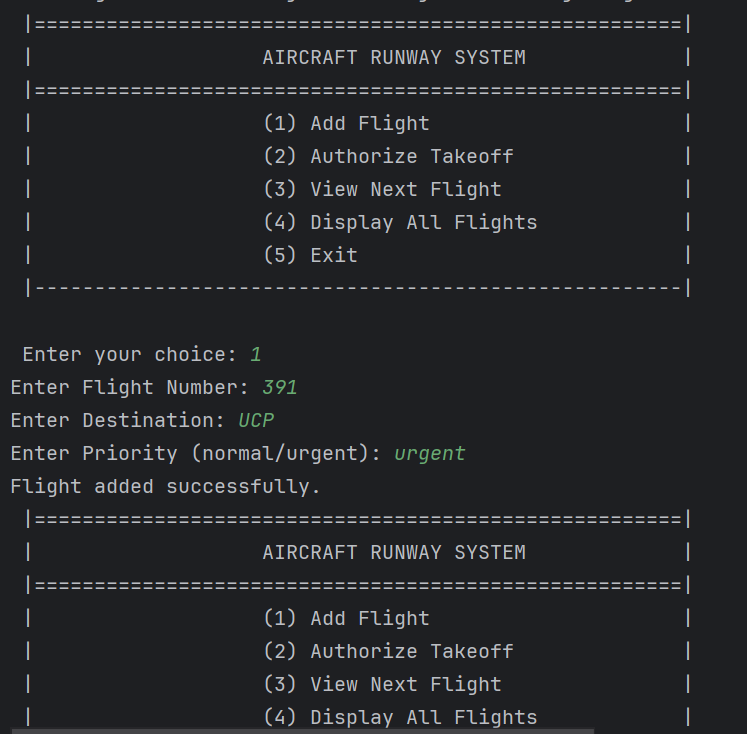


# Scenario 4: Aircraft Runway System

## CODE

// L1F23BSSE0391 - Abdullah Maqbool - P4  
  
import java.util.\*;  
  
class Aircraft {  
 String flightNumber;  
 String destination;  
 String priority;  
  
 public Aircraft(String flightNumber, String destination, String priority) {  
 this.flightNumber = flightNumber;  
 this.destination = destination;  
 this.priority = priority;  
 }  
}  
  
class RunwaySystem {  
 Queue<Aircraft> queue = new LinkedList<>();  
 Set<String> flightNumbers = new HashSet<>();  
  
 public void addFlight(String flightNumber, String destination, String priority) {  
 if (flightNumbers.contains(flightNumber)) {  
 System.*out*.println("Error: Flight number already exists!");  
 return;  
 }  
 Aircraft newFlight = new Aircraft(flightNumber, destination, priority);  
  
 if (priority.equalsIgnoreCase("urgent")) {  
 LinkedList<Aircraft> tempList = new LinkedList<>(queue);  
 tempList.addFirst(newFlight);  
 queue = new LinkedList<>(tempList);  
 } else {  
 queue.add(newFlight);  
 }  
  
 flightNumbers.add(flightNumber);  
 System.*out*.println("Flight added successfully.");  
 }  
  
 public void authorizeTakeoff() {  
 if (queue.isEmpty()) {  
 System.*out*.println("No flights waiting for takeoff.");  
 return;  
 }  
 Aircraft flight = queue.poll();  
 flightNumbers.remove(flight.flightNumber);  
 System.*out*.println("Flight " + flight.flightNumber + " to " + flight.destination + " has taken off!");  
 }  
  
 public void peekNextFlight() {  
 if (queue.isEmpty()) {  
 System.*out*.println("No flights in the queue.");  
 return;  
 }  
 Aircraft next = queue.peek();  
 System.*out*.println("Next Flight: " + next.flightNumber + " ~ Destination: " + next.destination + " ~ Priority: " + next.priority);  
 }  
  
 public void displayQueue() {  
 if (queue.isEmpty()) {  
 System.*out*.println("No flights waiting in the queue.");  
 return;  
 }  
 System.*out*.println("Flights waiting in queue:");  
 int count = 1;  
 for (Aircraft f : queue) {  
 System.*out*.println(count++ + ") " + f.flightNumber + " -> " + f.destination + " (" + f.priority + ")");  
 }  
 }  
}  
  
public class Scenario04 {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 RunwaySystem RS = new RunwaySystem();  
  
 while (true) {  
 System.*out*.println(" |======================================================|");  
 System.*out*.println(" | AIRCRAFT RUNWAY SYSTEM |");  
 System.*out*.println(" |======================================================|");  
 System.*out*.println(" | (1) Add Flight |");  
 System.*out*.println(" | (2) Authorize Takeoff |");  
 System.*out*.println(" | (3) View Next Flight |");  
 System.*out*.println(" | (4) Display All Flights |");  
 System.*out*.println(" | (5) Exit |");  
 System.*out*.println(" |------------------------------------------------------|");  
  
 System.*out*.print("\n Enter your choice: ");  
 int choice = sc.nextInt();  
 sc.nextLine(); // consume newline  
  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter Flight Number: ");  
 String flightNumber = sc.nextLine();  
 System.*out*.print("Enter Destination: ");  
 String destination = sc.nextLine();  
 System.*out*.print("Enter Priority (normal/urgent): ");  
 String priority = sc.nextLine();  
 RS.addFlight(flightNumber, destination, priority);  
 break;  
 case 2:  
 RS.authorizeTakeoff();  
 break;  
 case 3:  
 RS.peekNextFlight();  
 break;  
 case 4:  
 RS.displayQueue();  
 break;  
 case 5:  
 System.*out*.println("Thanks for using Aircraft Runway System, Exiting...");  
 return;  
 default:  
 System.*out*.println("Invalid choice,pls try again.");  
 }  
 }  
 }  
}

## Output Screenshot



----------------------------------------------------End of Assignment! ---------------------------------------------------