|  |  |
| --- | --- |
| ***Roll No*** | ***22SW040*** |
| ***Section*** | ***01*** |
| ***Name*** | ***Farooque Sajjad*** |
| ***LAB No*** | ***08(Stacks)*** |

***Task N0 : 01***

***public class ArrayStack <F>{  
 int size ;  
 F [] arrayQueue ;  
  
 public ArrayStack(int capacity) {  
 this.size = 0;  
 arrayQueue = (F[]) new Object[capacity];  
 }  
  
 public void push(F e ){  
 arrayQueue[size]=e;  
 size++;  
  
 }  
 public int getSize(){  
 return size;  
 }  
 public boolean isEmpty(){  
 return size==0;  
 }  
 public boolean search(F target){  
  
 for(int i=0;i<size;i++){  
 if(arrayQueue[i].equals(target)){  
 return true;  
 }  
 }  
 return false;  
  
 }  
 public F pop(){  
  
 if(isEmpty()){  
 throw new IllegalStateException("Stack is Empty");  
 }  
  
 F value = arrayQueue[size-1];  
 --size;  
  
 return value;  
 }  
  
 public void display(){  
 System.out.print("Stack : ");  
 for(int i=0;i<size;i++){  
 System.out.print(arrayQueue[i]+" ");  
 }  
 System.out.println();  
  
 }  
  
 public static void main(String[] args) {  
 ArrayStack<Integer> stack = new ArrayStack<>(10);  
  
 for (int i = 1; i <= 10; i++) {  
 stack.push(i \* 10);  
 }  
  
 System.out.println("Is stack empty? " + stack.isEmpty());  
 System.out.println("Stack size: " + stack.getSize());  
 System.out.println("Search for 50: " + stack.search(50));  
  
 System.out.println("Popping and displaying stack:");  
 while (!stack.isEmpty()) {  
 System.out.println("Popped: " + stack.pop());  
 stack.display();  
 }  
 }  
}***

******

***Task N0 : 02***

***public class LinkedStack<F> {  
 int size = 0;  
  
 public class Node {  
 F data;  
 Node next;  
  
 public Node(F data) {  
 this.data = data;  
 }  
 }  
  
 Node top ;  
  
 public void push(F e) {  
 Node newNode = new Node(e);  
 newNode.next = top;  
 top = newNode;  
 size++;  
 }  
  
 public int getSize() {  
 return size;  
 }  
  
 public boolean isEmpty() {  
 return size == 0;  
 }  
  
 public boolean search(F target) {  
 Node current = top;  
  
 while (current != null) {  
 if (current.data.equals(target)) {  
 return true;  
 }  
 current = current.next;  
 }  
  
 return false;  
 }  
  
 public F pop() {  
 if (isEmpty()) {  
 throw new IllegalStateException("Stack is Empty");  
 }  
  
 F value = top.data;  
 top = top.next;  
 size--;  
  
  
  
  
 return value;  
 }  
  
  
  
 public void display() {  
 System.out.print("Stack: ");  
 Node current = top;  
 LinkedStack<F> tempStack = new LinkedStack<>();  
  
 while (current != null) {  
 tempStack.push(current.data);  
 current = current.next;  
 }  
  
 while (!tempStack.isEmpty()) {  
 System.out.print(tempStack.pop() + " ");  
 }  
  
 System.out.println();  
 }  
  
  
 public static void main(String[] args) {  
 LinkedStack<Integer> stack = new LinkedStack<>();  
  
 for (int i = 1; i <= 10; i++) {  
 stack.push(i \* 10);  
 }  
  
 System.out.println("Is stack empty? " + stack.isEmpty());  
 System.out.println("Stack size: " + stack.getSize());  
 System.out.println("Search for 50: " + stack.search(50));  
  
 System.out.println("Popping and displaying stack:");  
 while (!stack.isEmpty()) {  
 System.out.println("Popped: " + stack.pop());  
 stack.display();  
 }  
 }  
}***

******

***Task N0 : 03***

***public class Browser {  
 static class Node {  
 String url;  
 Node next;  
 Node prev;  
  
 public Node(String url) {  
 this.url = url;  
 this.next = null;  
 this.prev = null;  
 }  
 }  
  
 public static class BrowserHistory {  
 private Node current;  
 private Node head;  
  
 public BrowserHistory(String initialUrl) {  
 this.head = new Node(initialUrl);  
 this.current = head;  
 }  
  
 public void visit(String url) {  
 Node newNode = new Node(url);  
 current.next = newNode;  
 newNode.prev = current;  
 current = newNode;  
 }  
  
 public String getCurrentPage() {  
 return current.url;  
 }  
  
 public boolean canGoBack() {  
 return current.prev != null;  
 }  
  
 public boolean canGoForward() {  
 return current.next != null;  
 }  
  
 public String goBack() {  
 if (canGoBack()) {  
 current = current.prev;  
 return current.url;  
 } else {  
 return "Cannot go back. At the beginning of history.";  
 }  
 }  
  
 public String goForward() {  
 if (canGoForward()) {  
 current = current.next;  
 return current.url;  
 } else {  
 return "Cannot go forward. At the end of history.";  
 }  
 }  
  
 public static void main(String[] args) {  
 BrowserHistory browser = new BrowserHistory("https://www.example.com");  
  
 browser.visit("https://azure.microsoft.com/en-us/?wt.mc\_id=studentamb\_348493");  
 browser.visit("https://cloudblogs.miscrosoft.com/?wt.mc\_id=studentamb\_348493");  
 browser.visit("https://code.visualstudio.com/?wt.mc\_id=studentamb\_348493");  
 browser.visit("https://developer.microsoft.com/en-us/?wt.mc\_id=studentamb\_348493");  
 System.out.println("Current Page: " + browser.getCurrentPage());  
  
 System.out.println("Going back: " + browser.goBack());  
 System.out.println("Current Page: " + browser.getCurrentPage());  
  
 System.out.println("Going forward: " + browser.goForward());  
 System.out.println("Current Page: " + browser.getCurrentPage());  
 }  
 }  
}***

******