**Lab 6 DSA Queue**

**L1F23BSSE0395**

Task 1 Printer Job Queue

import java.util.Scanner;  
  
class Node{  
 String job;  
 Node next;  
  
 public Node(String Job){  
 this.job=Job;  
 next=null;  
 }  
}  
class PrinterLinkedList{  
 Node front,rear;  
  
 public PrinterLinkedList(){  
 front=rear=null;  
 }  
  
 public void enqueue(String Job){  
 Node newNode=new Node(Job);  
 if(rear==null){  
 front=rear=newNode;  
 }  
 else {  
 rear.next=newNode;  
 rear=newNode;  
 }  
 }  
 public String dequeue(){  
 if(front ==null){  
 System.*out*.println("Queue is empty.");  
 return null;  
 }  
  
 String dequeued=front.job;  
 front=front.next;  
 return dequeued;  
 }  
 public void display() {  
 Node current = front;  
 if (current == null) {  
 System.*out*.println("Queue is empty!");  
 return;  
 }  
 while (current != null) {  
 System.*out*.print(current.job + " ");  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
 public String peek() {  
 if (front == null) {  
 System.*out*.println("Queue is empty.");  
 return null;  
 }  
 System.*out*.println("Current job at front: " + front.job);  
 return front.job;  
 }  
  
}  
  
  
public class Printer {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 PrinterLinkedList queue = new PrinterLinkedList();  
 int choice;  
  
 do {  
 System.*out*.println("\n--- Print Job Management ---");  
 System.*out*.println("1. Add a new print job");  
 System.*out*.println("2. Process next print job");  
 System.*out*.println("3. View current job at front");  
 System.*out*.println("4. Display all jobs");  
 System.*out*.println("5. Exit");  
 System.*out*.print("Enter your choice: ");  
 choice = scanner.nextInt();  
 scanner.nextLine();  
  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter job name: ");  
 String jobName = scanner.nextLine();  
 queue.enqueue(jobName);  
 break;  
 case 2:  
 queue.dequeue();  
 break;  
 case 3:  
 queue.peek();  
 break;  
 case 4:  
 queue.display();  
 break;  
 case 5:  
 System.*out*.println("Exiting Print Job Manager...");  
 break;  
 default:  
 System.*out*.println("Invalid choice. Try again.");  
 }  
 } while (choice != 5);  
  
 scanner.close();  
 }  
}



**Task 2 Call Center Queue**

/\*  
add(element): Adds an element to the rear of the queue. If the queue is full, it throws an exception.  
offer(element): Adds an element to the rear of the queue. If the queue is full, it returns false.  
remove(): Removes and returns the element at the front of the queue. If the queue is empty, it throws an exception.  
poll(): Removes and returns the element at the front of the queue. If the queue is empty, it returns null.  
element(): Returns the element at the front of the queue without removing it. If the queue is empty, it throws an exception.  
peek(): Returns the element at the front of the queue without removing it. If the queue is empty, it returns null.  
\*/  
  
/\*  
 Why use % capacity?  
 In a circular queue, the end of the array "wraps around" to the beginning.  
 This allows you to reuse space when elements are dequeued instead of shifting everything forward like in a linear queue.  
 The % capacity ensures that the index always stays within bounds of the array. Once you hit the end, it loops back to 0.  
\*/  
  
  
import java.util.Scanner;  
  
class CallCenterCircularArrayQueue{  
 int capacity,rear,front,size;  
 String[] callers;  
 public CallCenterCircularArrayQueue(int capacity){  
 this.capacity=capacity;  
 callers=new String[capacity];  
 rear=-1;  
 front=size=0;  
  
 }  
  
 public void addCaller( String str){  
 if(capacity==size){  
 System.*out*.println("Queue is full. Cannot add Caller :"+str);  
 return;  
 }  
 rear=(rear+1)%capacity;  
 callers[rear]=str;  
 size++;  
  
 System.*out*.println("Caller \""+str+"\" added to the queue successfully.");  
 }  
 public String connectCaller(){  
 if(size==0){  
 System.*out*.println("No caller in the queue.");  
 return null;  
 }  
 String dequeued=callers[front];  
 System.*out*.println("Connecting to caller: "+dequeued);  
 front=(front+1)%capacity;  
 size--;  
 return dequeued;  
 }  
 public String nextCaller() {  
 if (size == 0) {  
 System.*out*.println("No caller in the queue.");  
 return null;  
 }  
 System.*out*.println("Next caller: " + callers[front]);  
 return callers[front];  
 }  
 public void displayCallers(){  
 if (size == 0) {  
 System.*out*.println("No waiting caller in the queue.");  
 return ;  
 }  
 int i=front;  
 for (int j = 0; j < size; j++) {  
 System.*out*.println(j+1+"- " + callers[i]);  
 i=(i+1)%capacity;  
   
 }  
 }  
}  
  
  
public class CallCenter {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 CallCenterCircularArrayQueue queue = new CallCenterCircularArrayQueue(5);  
 int choice;  
  
 do {  
 System.*out*.println("\n--- Customer Support Call Center ---");  
 System.*out*.println("1. Add a new caller");  
 System.*out*.println("2. Connect next caller to agent");  
 System.*out*.println("3. Check who is next in line");  
 System.*out*.println("4. Display all waiting callers");  
 System.*out*.println("5. Exit");  
 System.*out*.print("Enter your choice: ");  
 choice = scanner.nextInt();  
 scanner.nextLine();  
  
 switch (choice) {  
 case 1:  
 System.*out*.print("Enter caller name: ");  
 String name = scanner.nextLine();  
 queue.addCaller(name);  
 break;  
 case 2:  
 queue.connectCaller();  
 break;  
 case 3:  
 queue.nextCaller();  
 break;  
 case 4:  
 queue.displayCallers();  
 break;  
 case 5:  
 System.*out*.println("Exiting Call Center Manager...");  
 break;  
 default:  
 System.*out*.println("Invalid choice. Try again.");  
 }  
 } while (choice != 5);  
  
 scanner.close();  
 }  
}

