5. Queue Interface

◆ Direct:

Implement a simple task queue using LinkedList as a Queue.

```
import java.util.LinkedList;
import java.util.Queue;

public class SimpleTaskQueue {

    public static void main(String[] args) {
        // Create a Queue using LinkedList
        Queue<String> taskQueue = new LinkedList<>();

        // Add tasks to the queue
        taskQueue.offer("Task 1: Write report");
        taskQueue.offer("Task 2: Email client");
        taskQueue.offer("Task 3: Prepare presentation");

        // Print all tasks in the queue
        System.out.println("Tasks in the queue:");
        for (String task : taskQueue) {
            System.out.println(task);
        }
    }
}
```

Demonstrate how to add and remove elements using offer() and poll().

```
import java.util.LinkedList;
import java.util.Queue;

public class OfferPollDemo {

    public static void main(String[] args) {
        Queue<String> taskQueue = new LinkedList<>();

        // Add tasks using offer()
        taskQueue.offer("Task 1: Write report");
        taskQueue.offer("Task 2: Email client");
        taskQueue.offer("Task 3: Prepare presentation");

        System.out.println("Processing tasks:");

        // Remove and process tasks using poll()
        while (!taskQueue.isEmpty()) {
            String currentTask = taskQueue.poll();  // retrieves and removes head
            System.out.println("Processing: " + currentTask);
        }
        // Try polling from empty queue returns null
        String noTask = taskQueue.poll();
        System.out.println("Polling empty queue returns: " + noTask);
    }
}
```

Use a PriorityQueue to order tasks by priority (integers).

```
import java.util.PriorityQueue;

public class PriorityTaskQueue {

    public static void main(String[] args) {
        // Create a PriorityQueue of Task objects
        PriorityQueue<Task> priorityQueue = new PriorityQueue<>();

        // Add tasks with different priorities (lower number = higher priority)
        priorityQueue.offer(new Task("Low priority task", 5));
        priorityQueue.offer(new Task("High priority task", 1));
        priorityQueue.offer(new Task("Medium priority task", 3));

        System.out.println("Processing tasks by priority:");

        // Process tasks according to priority order
        while (!priorityQueue.isEmpty()) {
            Task task = priorityQueue.poll();
            System.out.println("Processing: " + task.name + " (Priority: " + task.priority + ")");
        }
    }

    // Task class implements Comparable to define priority order
    static class Task implements Comparable<Task> {
        String name;
        int priority;

        Task(String name, int priority) {
            this.name = name;
            this.priority = priority;
        }

        // Compare tasks by priority (lower number = higher priority)
        @Override
        public int compareTo(Task other) {
            return Integer.compare(this.priority, other.priority);
        }
    }
}
```

◆ **Scenario-Based:**

Simulate a print queue system where print jobs are processed in order.

```
import java.util.LinkedList;
import java.util.Queue;

public class PrintQueueSystem {

    public static void main(String[] args) {
        Queue<String> printQueue = new LinkedList<>();

        // Adding print jobs to the queue
        printQueue.offer("Document1.pdf");
        printQueue.offer("Photo.png");
        printQueue.offer("Report.docx");

        System.out.println("Starting print jobs:");

        // Process print jobs in the order they were added
        while (!printQueue.isEmpty()) {
            String job = printQueue.poll();
            System.out.println("Printing: " + job);
        }

        System.out.println("All print jobs completed.");
    }
}
```

Create a ticket booking system where customer names are added to a queue and served in order.

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

public class TicketBookingSystem {

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

        System.out.print("How many customers are booking tickets? ");
        int n = scanner.nextInt();
        scanner.nextLine(); // consume leftover newline

        // Add customers to the queue
        for (int i = 0; i < n; i++) {
            System.out.print("Enter customer name: ");
            String customer = scanner.nextLine();
            ticketQueue.offer(customer);
        }
        System.out.println("\nServing customers in order:");

        // Serve customers in FIFO order
        while (!ticketQueue.isEmpty()) {
            String currentCustomer = ticketQueue.poll();
            System.out.println("Serving: " + currentCustomer);
        }

        scanner.close();
    }
}
```