05/08/24

1)Create a stack using linked list

```
class Node {
    int data;
    Node next;

    public Node(int data) {
        this.data = data;
        this.next = null;
    }
}

class Stack {
    Node top;
    public Stack() {
        this.top = null;
    }


    public void push(int data) {
        Node newNode = new Node(data);
        if (this.top == null) {
            this.top = newNode;
        } else {
```

```
            newNode.next = this.top;
            this.top = newNode;
        }
        System.out.println("Pushed element: " + data);
    }
    public void pop() {
        if (this.top == null) {
            System.out.println("Stack is empty");
        } else {
            int data = this.top.data;
            this.top = this.top.next;
            System.out.println("Popped element: " + data);
        }
    }
}
public class Main {
    public static void main(String[] args) {
        Stack stack = new Stack();
        stack.push(1);
        stack.push(2);
        stack.push(3);
        stack.push(4);
        stack.push(5);
        stack.pop();
        stack.pop();
```

```
        stack.pop();
        stack.pop();
        stack.pop();
        stack.pop();

    }
}
```

Output

```
java -cp /tmp/YtJvfu8CPO/Main
Pushed element: 1
Pushed element: 2
Pushed element: 3
Pushed element: 4
Pushed element: 5
Popped element: 5
Popped element: 4
Popped element: 3
Popped element: 2
Popped element: 1
Stack is empty

=== Code Execution Successful ===
```

2)create queue using linked list

```
class Node {
    int data;
    Node next;

    public Node(int data) {
```

```
        this.data = data;
        this.next = null;
    }
}
class Queue {
    Node front;
    Node rear;

    public Queue() {
        this.front = null;
        this.rear = null;
    }
    public void enqueue(int data) {
        Node newNode = new Node(data);

        if (rear == null) {
            front = newNode;
            rear = newNode;
        } else {
            rear.next = newNode;
            rear = newNode;
        }
    }
    public int dequeue() {
        if (front == null) {
```

```
            throw new RuntimeException("Queue is empty");
        }
        int data = front.data;
        front = front.next;

        if (front == null) {
            rear = null;
        }
        return data;
    }
    public int peek() {
        if (front == null) {
            throw new RuntimeException("Queue is empty");
        }
        return front.data;
    }
    public boolean isEmpty() {
        return front == null;
    }
}
public class Main {
    public static void main(String[] args) {
        Queue queue = new Queue();
        queue.enqueue(1);
        queue.enqueue(2);
```

```
        queue.enqueue(3);
        System.out.println(queue.dequeue());
        System.out.println(queue.dequeue());
        System.out.println(queue.dequeue());
        System.out.println(queue.isEmpty());
    }
}
```

2)Mobile class

```
import java.util.Arrays;
import java.util.Comparator;
class Mobile {
    private double price;
    private String brand;
    private String model;
    private int ram;
```

```
public Mobile(double price, String brand, String model, int ram) {
    this.price = price;
    this.brand = brand;
    this.model = model;
    this.ram = ram;
}
public double getPrice() {
    return price;
}

public void setPrice(double price) {
    this.price = price;
}
public String getBrand() {
    return brand;
}
public void setBrand(String brand) {
    this.brand = brand;
}
public String getModel() {
    return model;
}

public void setModel(String model) {
    this.model = model;
```

```
    }
    public int getRam() {
        return ram;
    }
    public void setRam(int ram) {
        this.ram = ram;
    }
    @Override
    public String toString() {
        return "Mobile{" +
                "price=" + price +
                ", brand='" + brand + '\'' +
                ", model='" + model + '\'' +
                ", ram=" + ram +
                '}';
    }
}

public class Main {
    public static void main(String[] args) {
        Mobile[] mobiles = new Mobile[] {
                new Mobile(15000, "Samsung", "Galaxy M31", 6),
                new Mobile(20000, "Apple", "iPhone 12", 4),
                new Mobile(10000, "Xiaomi", "Redmi 9", 4),
                new Mobile(25000, "OnePlus", "Nord", 8),
```

```
            new Mobile(12000, "Realme", "6 Pro", 6)
        };
        System.out.println("Unsorted Mobiles:");
        Arrays.stream(mobiles).forEach(System.out::println);
        Arrays.sort(mobiles, Comparator.comparingDouble(Mobile::getPrice));
        System.out.println("\nSorted Mobiles by Price:");
        Arrays.stream(mobiles).forEach(System.out::println);
    }
}
```

Output

```
java -cp /tmp/hNX8V4tm51/Main
Unsorted Mobiles:
Mobile{price=15000.0, brand='Samsung', model='Galaxy M31', ram=6}
Mobile{price=20000.0, brand='Apple', model='iPhone 12', ram=4}
Mobile{price=10000.0, brand='Xiaomi', model='Redmi 9', ram=4}
Mobile{price=25000.0, brand='OnePlus', model='Nord', ram=8}
Mobile{price=12000.0, brand='Realme', model='6 Pro', ram=6}

Sorted Mobiles by Price:
Mobile{price=10000.0, brand='Xiaomi', model='Redmi 9', ram=4}
Mobile{price=12000.0, brand='Realme', model='6 Pro', ram=6}
Mobile{price=15000.0, brand='Samsung', model='Galaxy M31', ram=6}
Mobile{price=20000.0, brand='Apple', model='iPhone 12', ram=4}
Mobile{price=25000.0, brand='OnePlus', model='Nord', ram=8}

=== Code Execution Successful ===
```