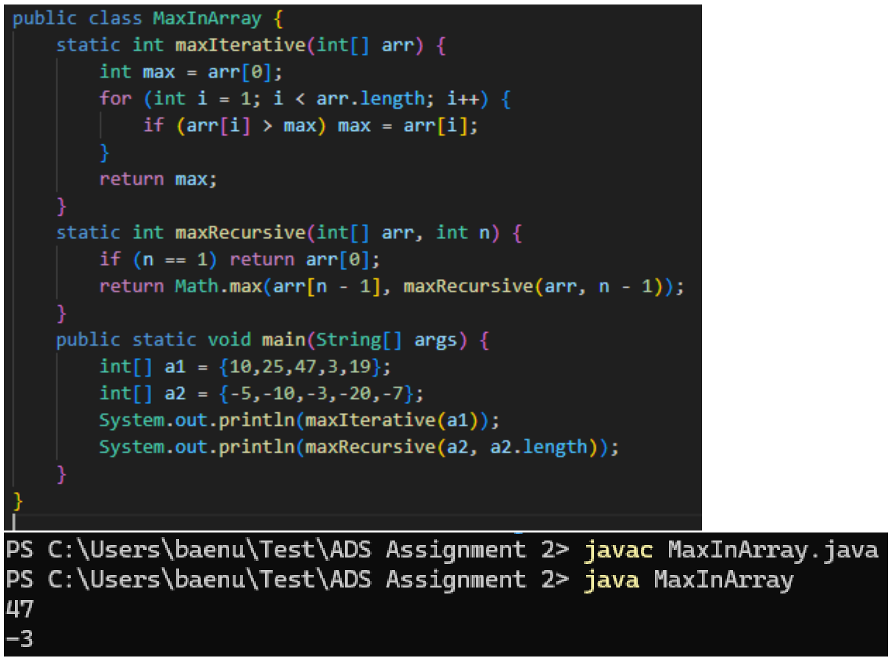
Assignment 2

Submission Date: 20/09/2025

ANSWERS

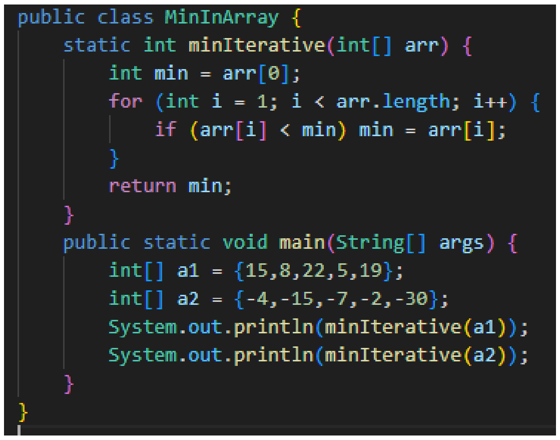
1. Find the maximum element in an array. Solve this problem using iterative and recursion method
Testcase1:
Input: [10, 25, 47, 3, 19]
Expected Output: 47
Testcase2:
Input: [-5, -10, -3, -20, -7]
Expected Output: -3



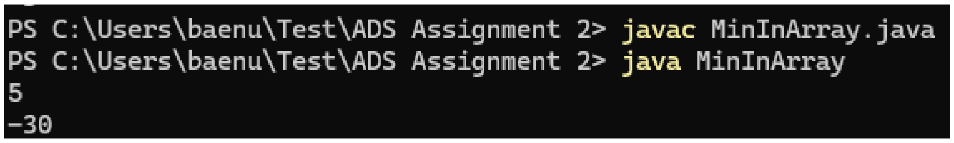
```java
public class MaxInArray {
    static int maxIterative(int[] arr) {
        int max = arr[0];
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] > max) max = arr[i];
        }
        return max;
    }
    static int maxRecursive(int[] arr, int n) {
        if (n == 1) return arr[0];
        return Math.max(arr[n - 1], maxRecursive(arr, n - 1));
    }
    public static void main(String[] args) {
        int[] a1 = {10,25,47,3,19};
        int[] a2 = {-5,-10,-3,-20,-7};
        System.out.println(maxIterative(a1));
        System.out.println(maxRecursive(a2, a2.length));
    }
}
```

```
PS C:\Users\baenu\Test\ADS Assignment 2> javac MaxInArray.java
PS C:\Users\baenu\Test\ADS Assignment 2> java MaxInArray
47
-3
```

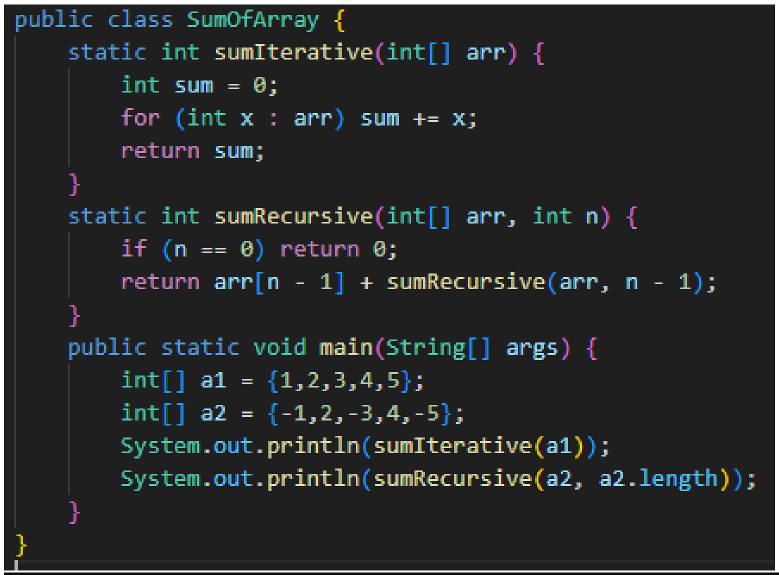
2. Find the minimum element in an array.
Testcases1:
Input: [15, 8, 22, 5, 19]
Expected Output: 5
Testcase2:
Input: [-4, -15, -7, -2, -30]
Expected Output: -30

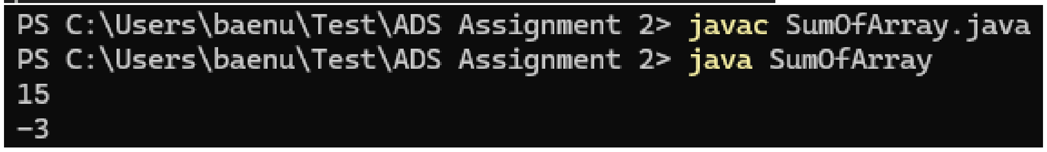


```java
public class MinInArray {
    static int minIterative(int[] arr) {
        int min = arr[0];
        for (int i = 1; i < arr.length; i++) {
            if (arr[i] < min) min = arr[i];
        }
        return min;
    }
    public static void main(String[] args) {
        int[] a1 = {15,8,22,5,19};
        int[] a2 = {-4,-15,-7,-2,-30};
        System.out.println(minIterative(a1));
        System.out.println(minIterative(a2));
    }
}
```

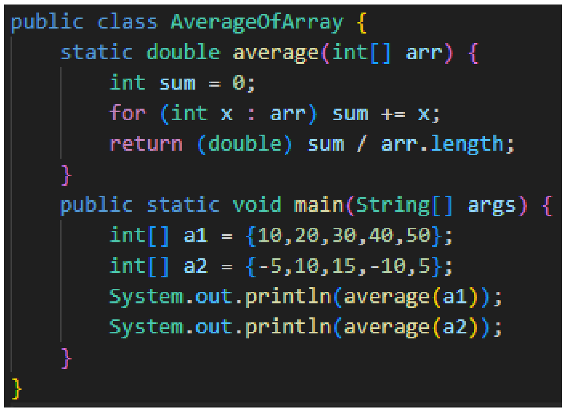

```
PS C:\Users\baenu\Test\ADS Assignment 2> javac MinInArray.java
PS C:\Users\baenu\Test\ADS Assignment 2> java MinInArray
5
-30
```

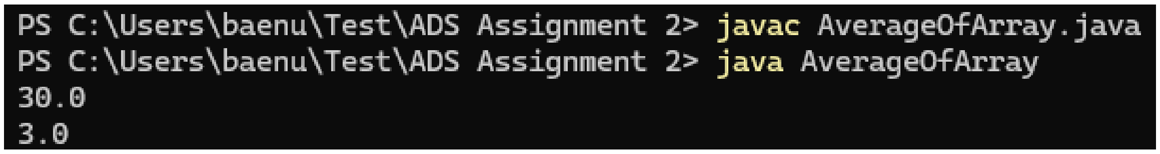
3. Calculate the sum of all array elements. Solve this problem using iterative and recursion method
Testcase1:
Input: [1, 2, 3, 4, 5]
Expected Output: 15
Testcase2:
Input: [-1, 2, -3, 4, -5]
Expected Output: -3


```java
public class SumOfArray {
    static int sumIterative(int[] arr) {
        int sum = 0;
        for (int x : arr) sum += x;
        return sum;
    }
    static int sumRecursive(int[] arr, int n) {
        if (n == 0) return 0;
        return arr[n - 1] + sumRecursive(arr, n - 1);
    }
    public static void main(String[] args) {
        int[] a1 = {1,2,3,4,5};
        int[] a2 = {-1,2,-3,4,-5};
        System.out.println(sumIterative(a1));
        System.out.println(sumRecursive(a2, a2.length));
    }
}
```


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac SumOfArray.java
PS C:\Users\baenu\Test\ADS Assignment 2> java SumOfArray
15
-3
```

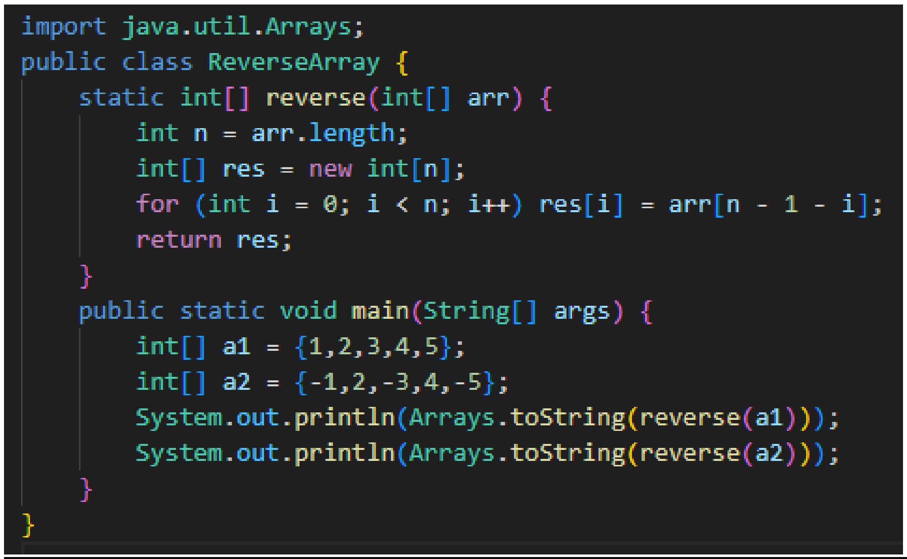
4. Find the average of array elements
Testcase1:
Input: [10, 20, 30, 40, 50]
Expected Output: 30.0
Testcase2:
Input: [-5, 10, 15, -10, 5]
Expected Output: 3.0


```java
public class AverageOfArray {
    static double average(int[] arr) {
        int sum = 0;
        for (int x : arr) sum += x;
        return (double) sum / arr.length;
    }
    public static void main(String[] args) {
        int[] a1 = {10,20,30,40,50};
        int[] a2 = {-5,10,15,-10,5};
        System.out.println(average(a1));
        System.out.println(average(a2));
    }
}
```

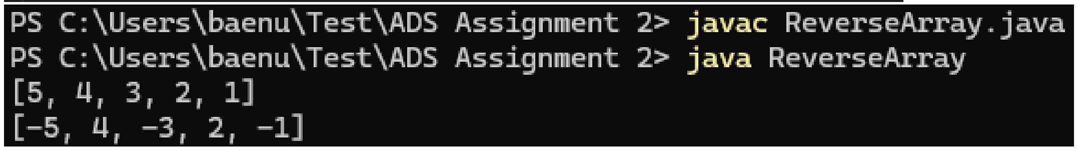


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac AverageOfArray.java
PS C:\Users\baenu\Test\ADS Assignment 2> java AverageOfArray
30.0
3.0
```

5. Print array elements in reverse order.
Testcase1:
Input: [1, 2, 3, 4, 5]
Expected Output: [5, 4, 3, 2, 1]
Testcase2:
Input: [-1, 2, -3, 4, -5]
Expected Output: [-5, 4, -3, 2, -1]

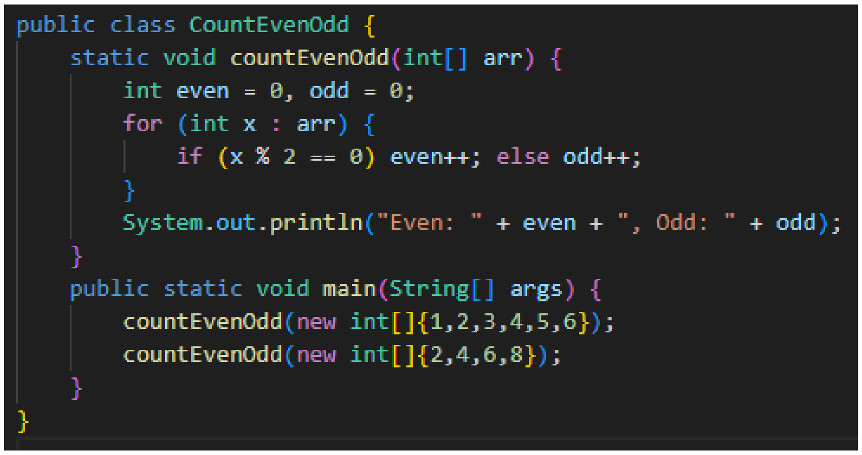


```java
import java.util.Arrays;
public class ReverseArray {
    static int[] reverse(int[] arr) {
        int n = arr.length;
        int[] res = new int[n];
        for (int i = 0; i < n; i++) res[i] = arr[n - 1 - i];
        return res;
    }
    public static void main(String[] args) {
        int[] a1 = {1,2,3,4,5};
        int[] a2 = {-1,2,-3,4,-5};
        System.out.println(Arrays.toString(reverse(a1)));
        System.out.println(Arrays.toString(reverse(a2)));
    }
}
```

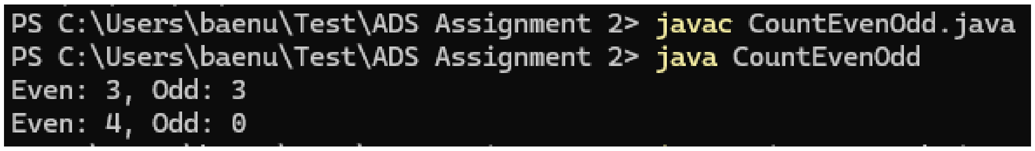


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac ReverseArray.java
PS C:\Users\baenu\Test\ADS Assignment 2> java ReverseArray
[5, 4, 3, 2, 1]
[-5, 4, -3, 2, -1]
```

6. Count even and odd elements in an array.
Testcase1:
Input: [1, 2, 3, 4, 5, 6]
Expected Output: Even: 3, Odd: 3
Input: [2, 4, 6, 8]
Expected Output: Even: 4, Odd: 0

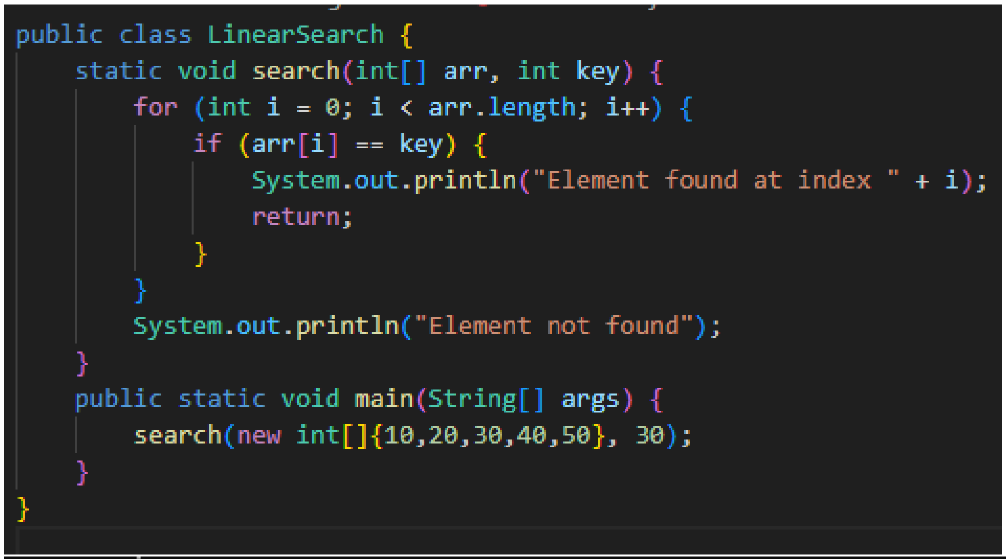


```java
public class CountEvenOdd {
    static void countEvenOdd(int[] arr) {
        int even = 0, odd = 0;
        for (int x : arr) {
            if (x % 2 == 0) even++; else odd++;
        }
        System.out.println("Even: " + even + ", Odd: " + odd);
    }
    public static void main(String[] args) {
        countEvenOdd(new int[]{1,2,3,4,5,6});
        countEvenOdd(new int[]{2,4,6,8});
    }
}
```

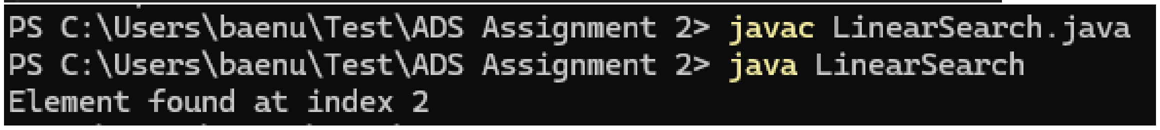


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac CountEvenOdd.java
PS C:\Users\baenu\Test\ADS Assignment 2> java CountEvenOdd
Even: 3, Odd: 3
Even: 4, Odd: 0
```

7. Search for an element in the array (linear search).
Testcase1:
Input: [10, 20, 30, 40, 50], Search Element: 30
Expected Output: Element found at index 2

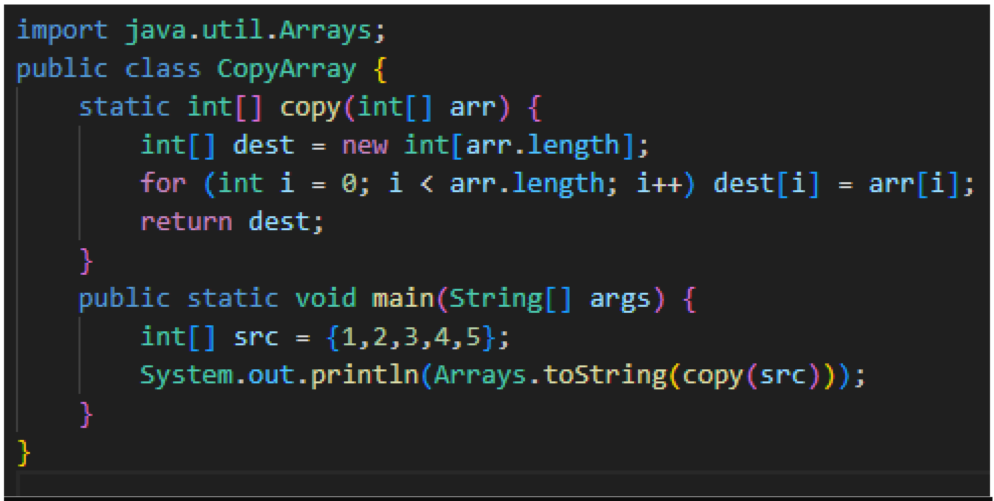


```java
public class LinearSearch {
    static void search(int[] arr, int key) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == key) {
                System.out.println("Element found at index " + i);
                return;
            }
        }
        System.out.println("Element not found");
    }
    public static void main(String[] args) {
        search(new int[]{10,20,30,40,50}, 30);
    }
}
```

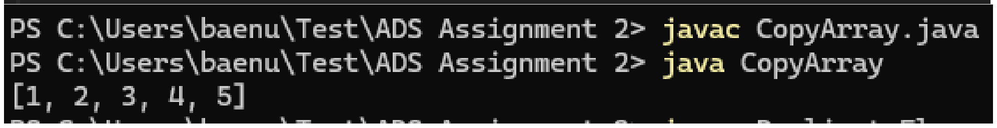


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac LinearSearch.java
PS C:\Users\baenu\Test\ADS Assignment 2> java LinearSearch
Element found at index 2
```

8. Copy elements of one array into another.
Testcase:
Input: Source Array: [1, 2, 3, 4, 5]
Expected Output: Destination Array: [1, 2, 3, 4, 5]

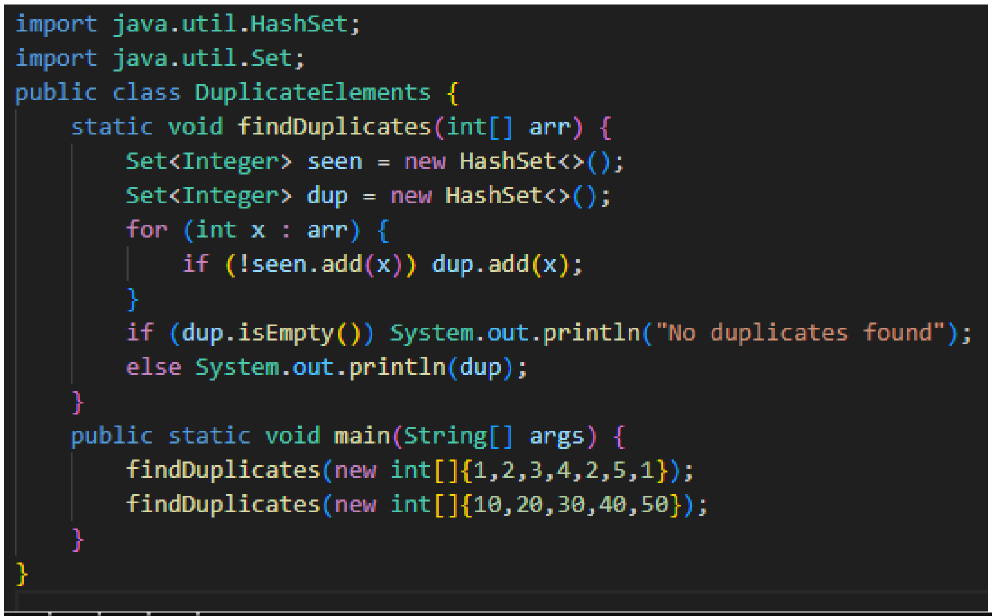


```java
import java.util.Arrays;
public class CopyArray {
    static int[] copy(int[] arr) {
        int[] dest = new int[arr.length];
        for (int i = 0; i < arr.length; i++) dest[i] = arr[i];
        return dest;
    }
    public static void main(String[] args) {
        int[] src = {1,2,3,4,5};
        System.out.println(Arrays.toString(copy(src)));
    }
}
```

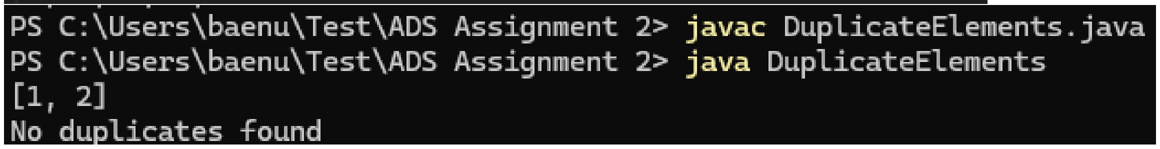


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac CopyArray.java
PS C:\Users\baenu\Test\ADS Assignment 2> java CopyArray
[1, 2, 3, 4, 5]
```

9. Display duplicate elements from an array.
Testcase1:
Input: [1, 2, 3, 4, 2, 5, 1]
Expected Output: 1, 2
Testcase2:
Input: [10, 20, 30, 40, 50]
Expected Output: No duplicates found

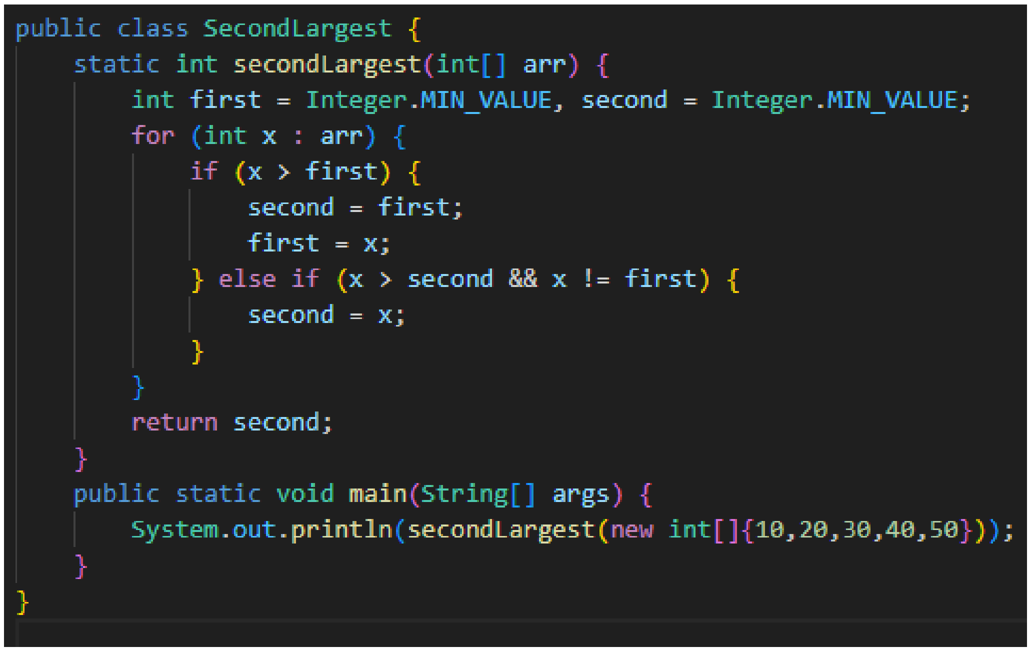


```java
import java.util.HashSet;
import java.util.Set;
public class DuplicateElements {
    static void findDuplicates(int[] arr) {
        Set<Integer> seen = new HashSet<>();
        Set<Integer> dup = new HashSet<>();
        for (int x : arr) {
            if (!seen.add(x)) dup.add(x);
        }
        if (dup.isEmpty()) System.out.println("No duplicates found");
        else System.out.println(dup);
    }
    public static void main(String[] args) {
        findDuplicates(new int[]{1,2,3,4,2,5,1});
        findDuplicates(new int[]{10,20,30,40,50});
    }
}
```

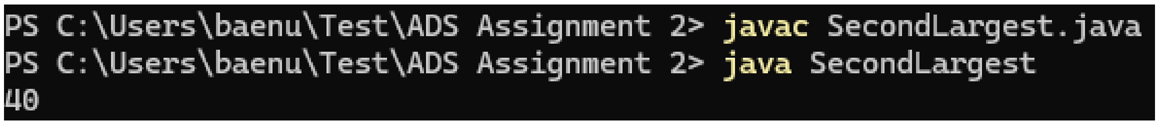


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac DuplicateElements.java
PS C:\Users\baenu\Test\ADS Assignment 2> java DuplicateElements
[1, 2]
No duplicates found
```

10. Find the second largest element in the array.
Testcase:
Input: [10, 20, 30, 40, 50]
Expected Output: 40

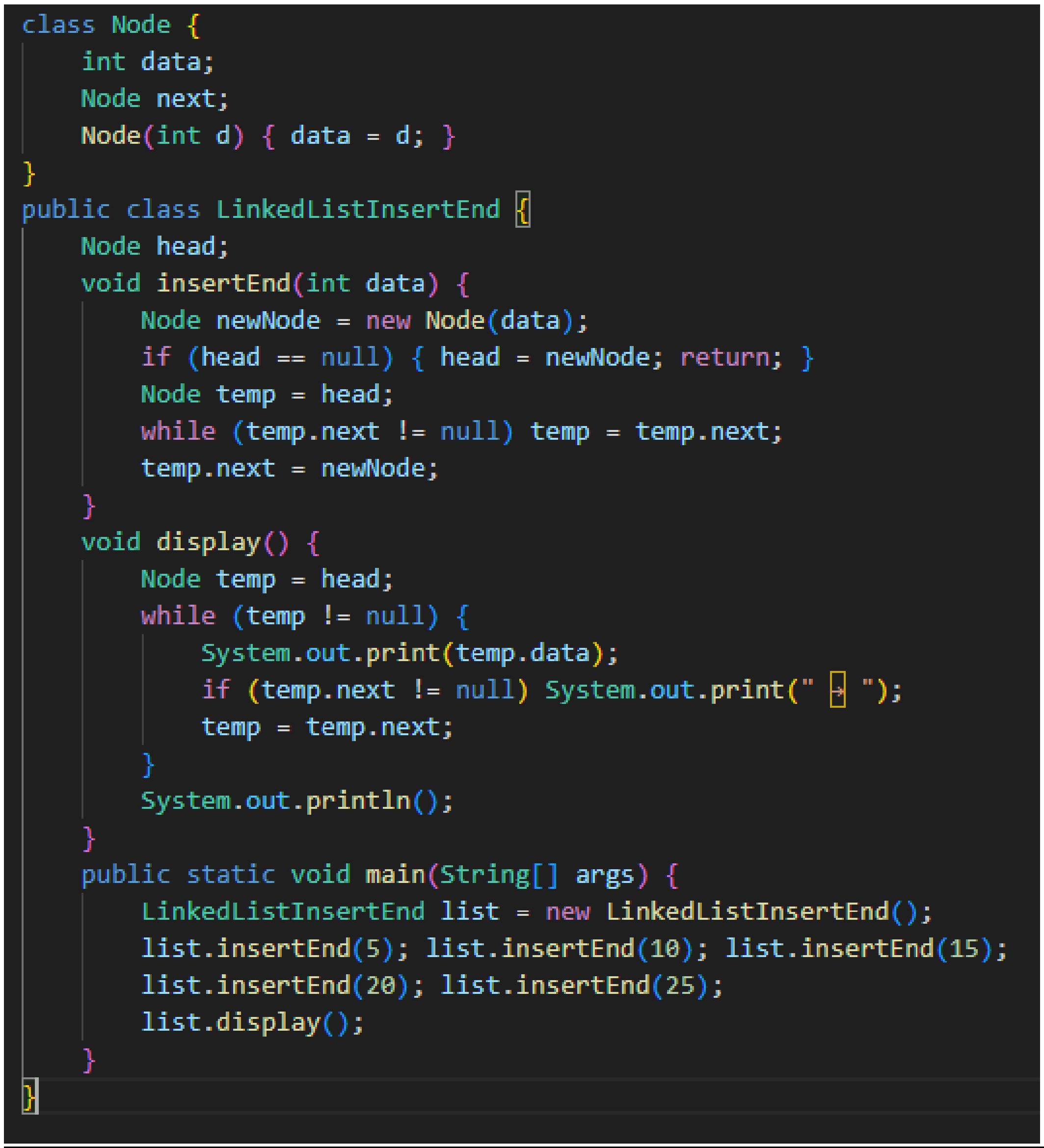


```java
public class SecondLargest {
    static int secondLargest(int[] arr) {
        int first = Integer.MIN_VALUE, second = Integer.MIN_VALUE;
        for (int x : arr) {
            if (x > first) {
                second = first;
                first = x;
            } else if (x > second && x != first) {
                second = x;
            }
        }
        return second;
    }
    public static void main(String[] args) {
        System.out.println(secondLargest(new int[]{10,20,30,40,50}));
    }
}
```

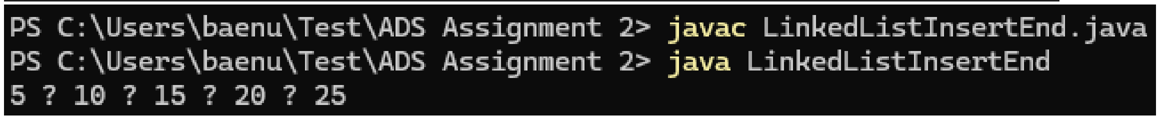


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac SecondLargest.java
PS C:\Users\baenu\Test\ADS Assignment 2> java SecondLargest
40
```

11. Create a LinkedList and insert elements at the end.
Testcase:
Existing LinkedList: [5, 10, 15]
Elements to insert: [20, 25]
Expected Output: LinkedList: 5 → 10 → 15 → 20 → 25

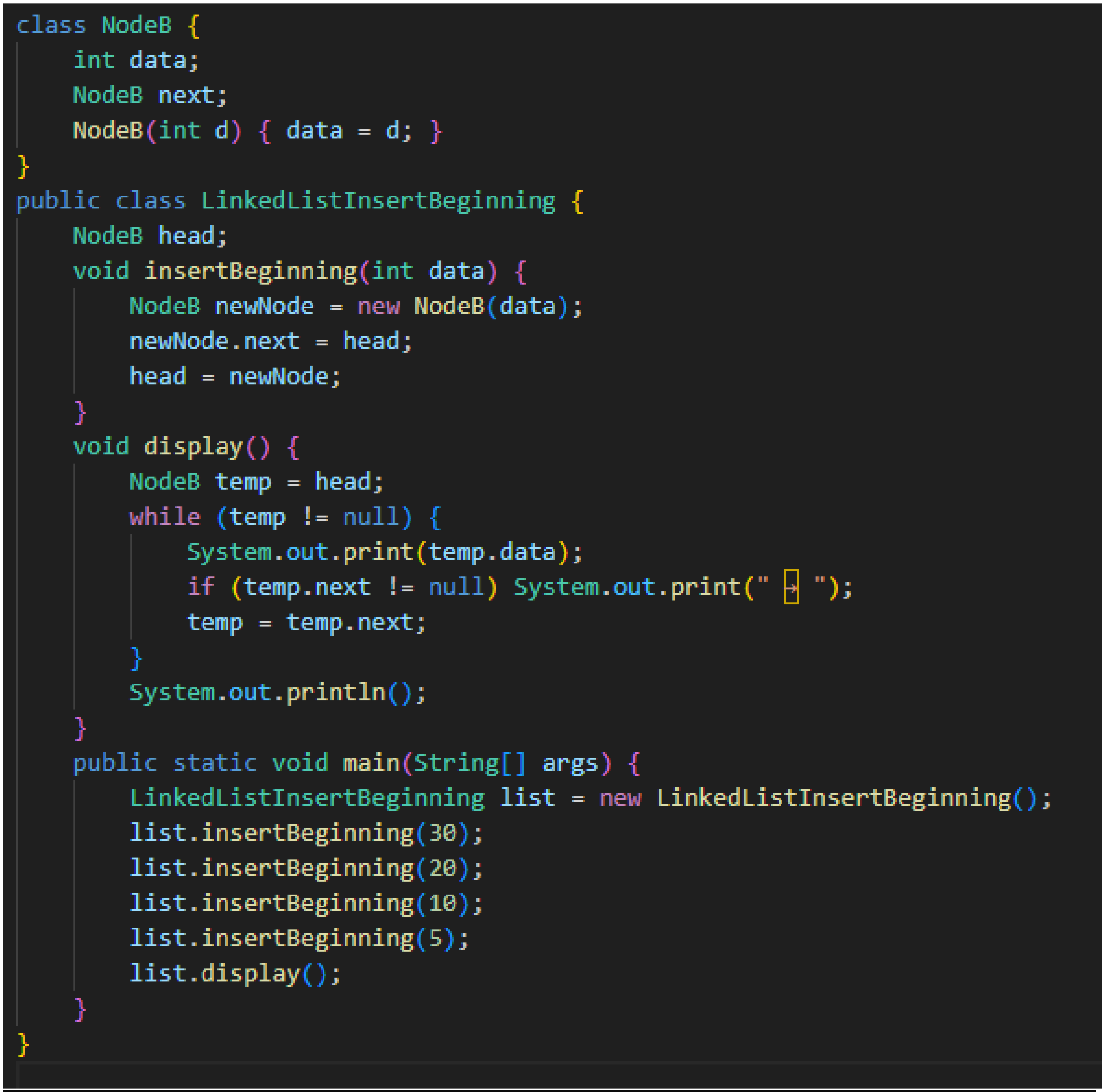


```java
class Node {
    int data;
    Node next;
    Node(int d) { data = d; }
}
public class LinkedListInsertEnd {
    Node head;
    void insertEnd(int data) {
        Node newNode = new Node(data);
        if (head == null) { head = newNode; return; }
        Node temp = head;
        while (temp.next != null) temp = temp.next;
        temp.next = newNode;
    }
    void display() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.data);
            if (temp.next != null) System.out.print(" → ");
            temp = temp.next;
        }
        System.out.println();
    }
    public static void main(String[] args) {
        LinkedListInsertEnd list = new LinkedListInsertEnd();
        list.insertEnd(5); list.insertEnd(10); list.insertEnd(15);
        list.insertEnd(20); list.insertEnd(25);
        list.display();
    }
}
```

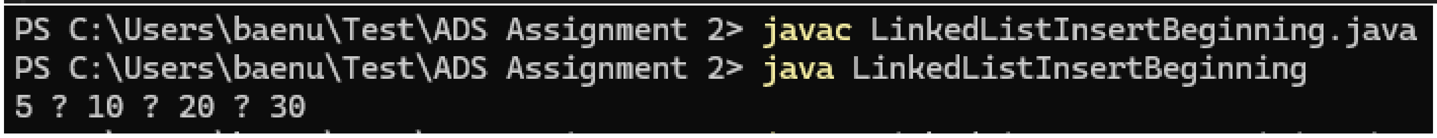


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac LinkedListInsertEnd.java
PS C:\Users\baenu\Test\ADS Assignment 2> java LinkedListInsertEnd
5 ? 10 ? 15 ? 20 ? 25
```

12. Insert a new node at the beginning of a LinkedList.
Testcase:
Existing LinkedList: [10, 20, 30]
Node to insert: 5
Expected Output: LinkedList: 5 → 10 → 20 → 30

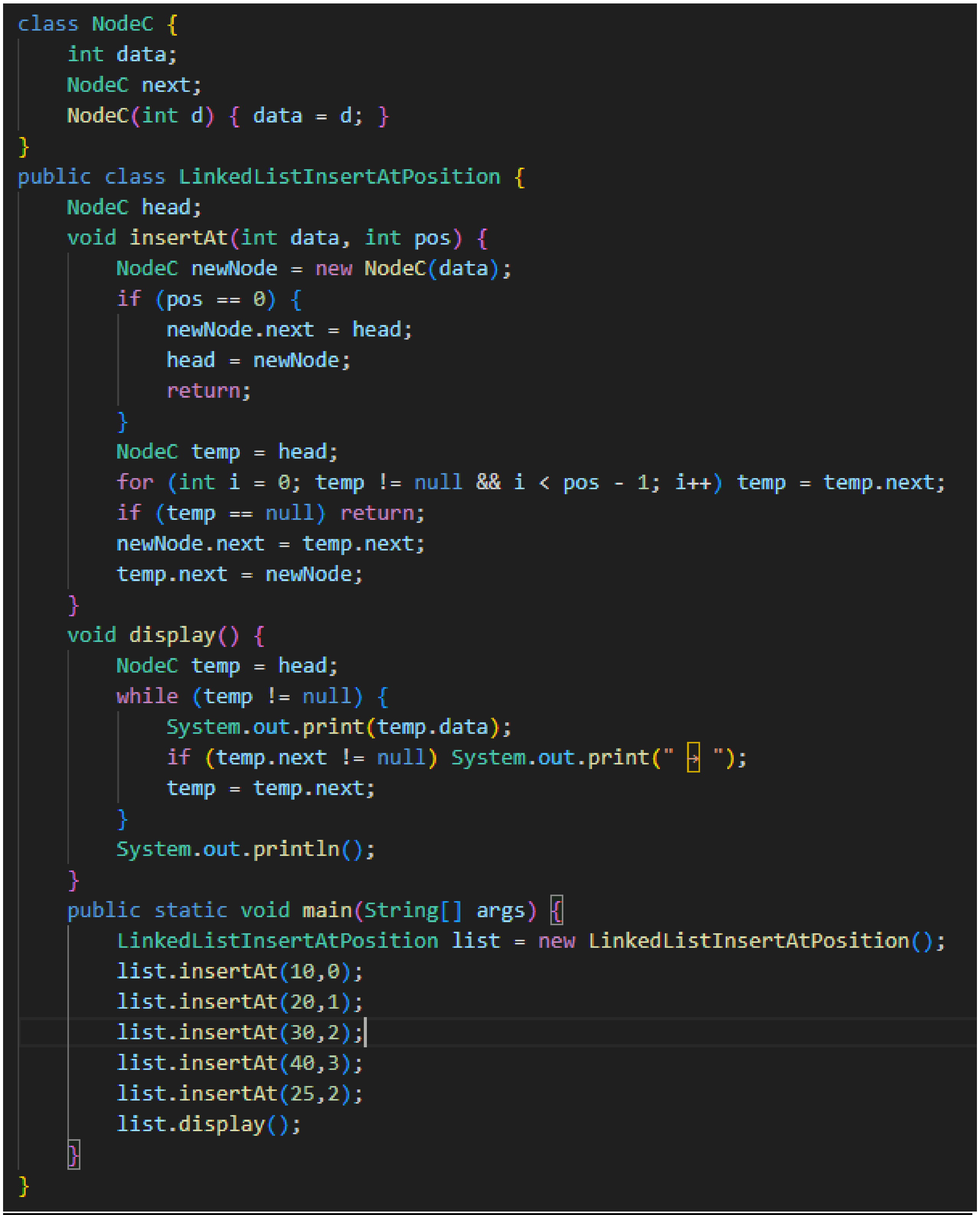


```java
class NodeB {
    int data;
    NodeB next;
    NodeB(int d) { data = d; }
}
public class LinkedListInsertBeginning {
    NodeB head;
    void insertBeginning(int data) {
        NodeB newNode = new NodeB(data);
        newNode.next = head;
        head = newNode;
    }
    void display() {
        NodeB temp = head;
        while (temp != null) {
            System.out.print(temp.data);
            if (temp.next != null) System.out.print(" → ");
            temp = temp.next;
        }
        System.out.println();
    }
    public static void main(String[] args) {
        LinkedListInsertBeginning list = new LinkedListInsertBeginning();
        list.insertBeginning(30);
        list.insertBeginning(20);
        list.insertBeginning(10);
        list.insertBeginning(5);
        list.display();
    }
}
```

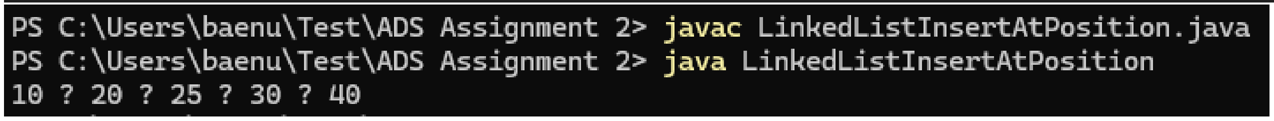


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac LinkedListInsertBeginning.java
PS C:\Users\baenu\Test\ADS Assignment 2> java LinkedListInsertBeginning
5 ? 10 ? 20 ? 30
```

13. Insert a new node at a given position in a LinkedList.
Testcase:
Existing LinkedList: [10, 20, 30, 40]
Node to insert: 25 at position 2
Expected Output: LinkedList: 10 → 20 → 25 → 30 → 40

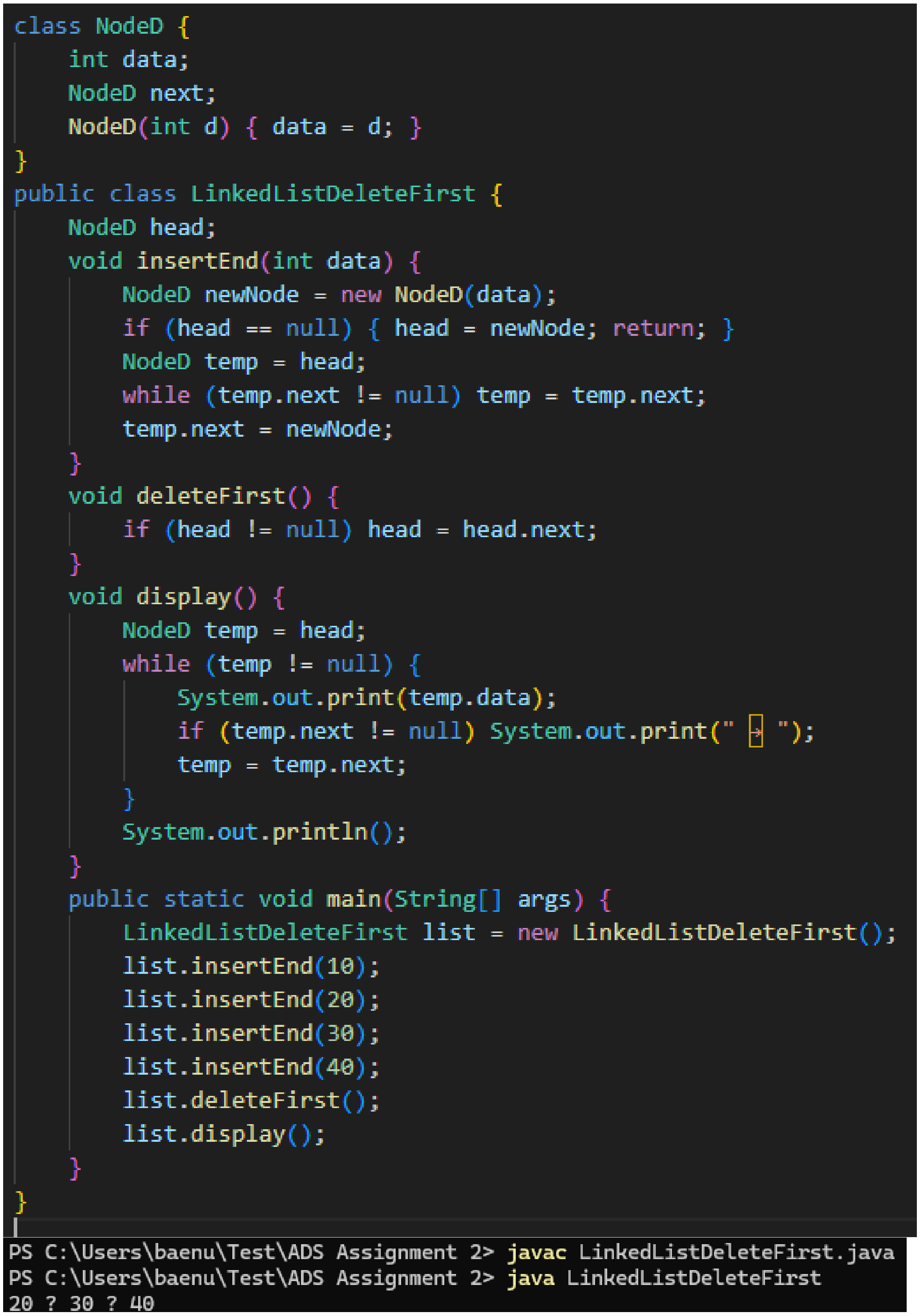


```java
class NodeC {
    int data;
    NodeC next;
    NodeC(int d) { data = d; }
}
public class LinkedListInsertAtPosition {
    NodeC head;
    void insertAt(int data, int pos) {
        NodeC newNode = new NodeC(data);
        if (pos == 0) {
            newNode.next = head;
            head = newNode;
            return;
        }
        NodeC temp = head;
        for (int i = 0; temp != null && i < pos - 1; i++) temp = temp.next;
        if (temp == null) return;
        newNode.next = temp.next;
        temp.next = newNode;
    }
    void display() {
        NodeC temp = head;
        while (temp != null) {
            System.out.print(temp.data);
            if (temp.next != null) System.out.print(" → ");
            temp = temp.next;
        }
        System.out.println();
    }
    public static void main(String[] args) {
        LinkedListInsertAtPosition list = new LinkedListInsertAtPosition();
        list.insertAt(10,0);
        list.insertAt(20,1);
        list.insertAt(30,2);
        list.insertAt(40,3);
        list.insertAt(25,2);
        list.display();
    }
}
```



```
PS C:\Users\baenu\Test\ADS Assignment 2> javac LinkedListInsertAtPosition.java
PS C:\Users\baenu\Test\ADS Assignment 2> java LinkedListInsertAtPosition
10 ? 20 ? 25 ? 30 ? 40
```

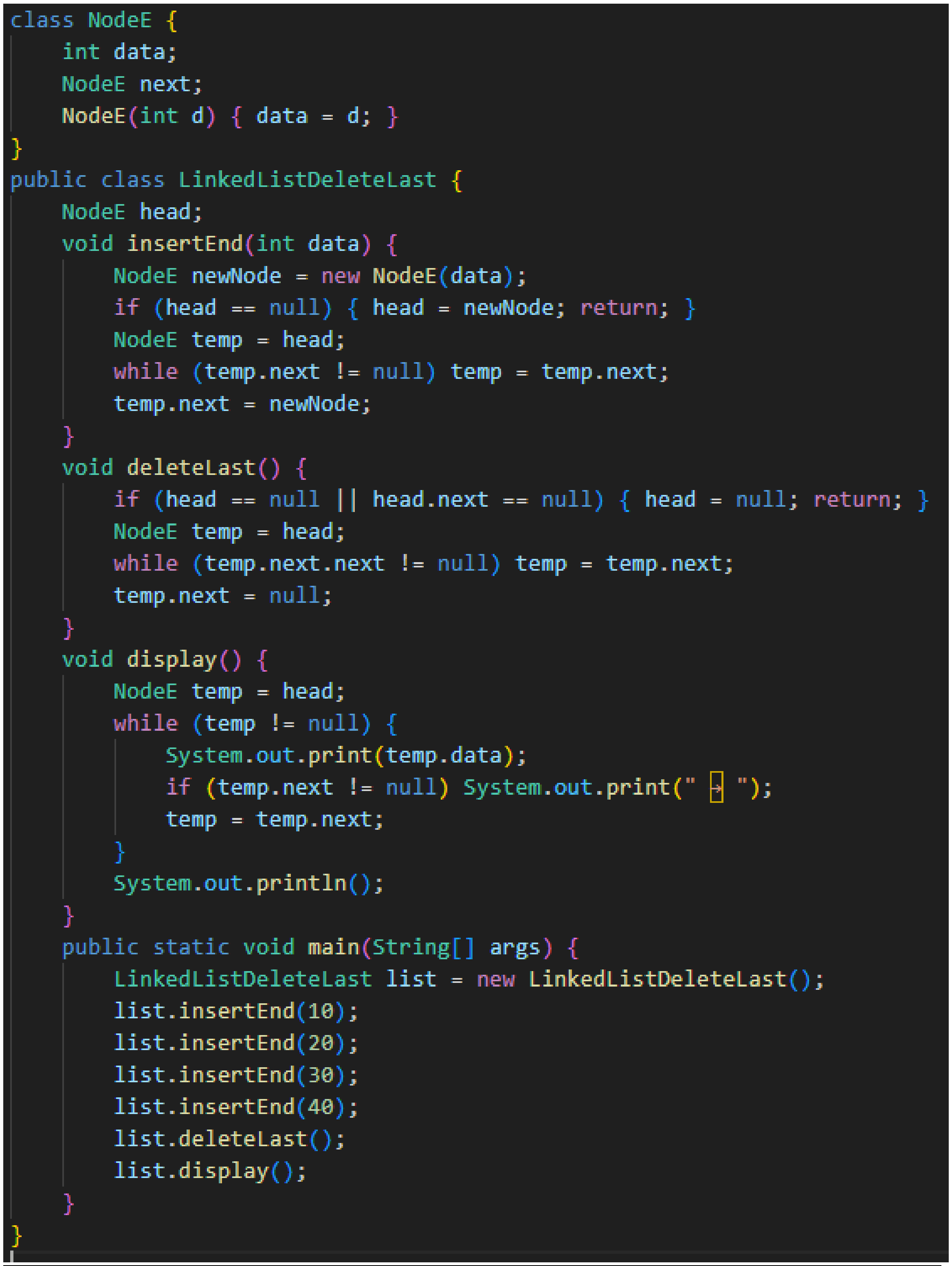
14. Delete the first node of a LinkedList
Testcase:
Existing LinkedList: [10, 20, 30, 40]
Expected Output: LinkedList: 20 → 30 → 40



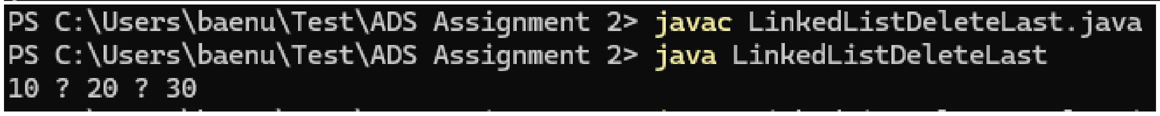
```
class NodeD {
    int data;
    NodeD next;
    NodeD(int d) { data = d; }
}
public class LinkedListDeleteFirst {
    NodeD head;
    void insertEnd(int data) {
        NodeD newNode = new NodeD(data);
        if (head == null) { head = newNode; return; }
        NodeD temp = head;
        while (temp.next != null) temp = temp.next;
        temp.next = newNode;
    }
    void deleteFirst() {
        if (head != null) head = head.next;
    }
    void display() {
        NodeD temp = head;
        while (temp != null) {
            System.out.print(temp.data);
            if (temp.next != null) System.out.print(" → ");
            temp = temp.next;
        }
        System.out.println();
    }
    public static void main(String[] args) {
        LinkedListDeleteFirst list = new LinkedListDeleteFirst();
        list.insertEnd(10);
        list.insertEnd(20);
        list.insertEnd(30);
        list.insertEnd(40);
        list.deleteFirst();
        list.display();
    }
}
```

```
PS C:\Users\baenu\Test\ADS Assignment 2> javac LinkedListDeleteFirst.java
PS C:\Users\baenu\Test\ADS Assignment 2> java LinkedListDeleteFirst
20 ? 30 ? 40
```

15. Delete the last node of a LinkedList.
Testcase:
Existing LinkedList: [10, 20, 30, 40]
Expected Output: LinkedList: 10 → 20 → 30

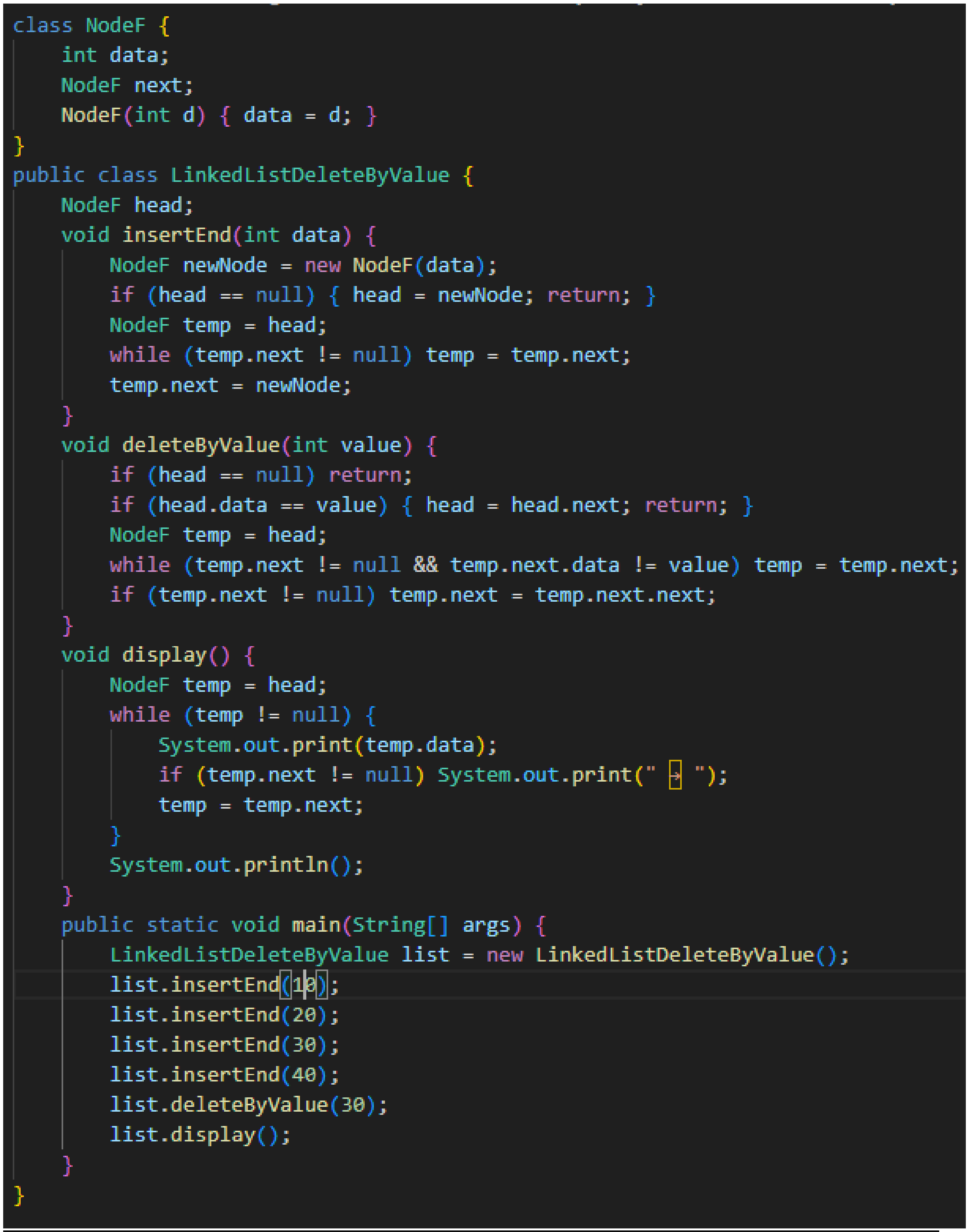


```java
class NodeE {
    int data;
    NodeE next;
    NodeE(int d) { data = d; }
}
public class LinkedListDeleteLast {
    NodeE head;
    void insertEnd(int data) {
        NodeE newNode = new NodeE(data);
        if (head == null) { head = newNode; return; }
        NodeE temp = head;
        while (temp.next != null) temp = temp.next;
        temp.next = newNode;
    }
    void deleteLast() {
        if (head == null || head.next == null) { head = null; return; }
        NodeE temp = head;
        while (temp.next.next != null) temp = temp.next;
        temp.next = null;
    }
    void display() {
        NodeE temp = head;
        while (temp != null) {
            System.out.print(temp.data);
            if (temp.next != null) System.out.print(" → ");
            temp = temp.next;
        }
        System.out.println();
    }
    public static void main(String[] args) {
        LinkedListDeleteLast list = new LinkedListDeleteLast();
        list.insertEnd(10);
        list.insertEnd(20);
        list.insertEnd(30);
        list.insertEnd(40);
        list.deleteLast();
        list.display();
    }
}
```

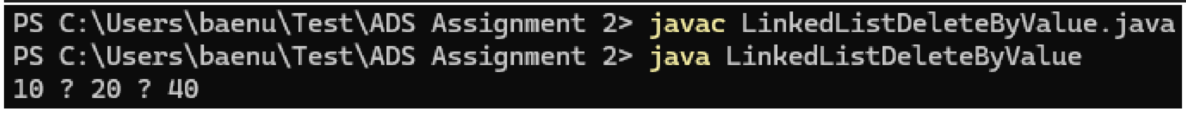


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac LinkedListDeleteLast.java
PS C:\Users\baenu\Test\ADS Assignment 2> java LinkedListDeleteLast
10 ? 20 ? 30
```

16. Delete a node by its value in a LinkedList.
Testcase:
Existing LinkedList: [10, 20, 30, 40]
Node to delete: 30
Expected Output: LinkedList: 10 → 20 → 40

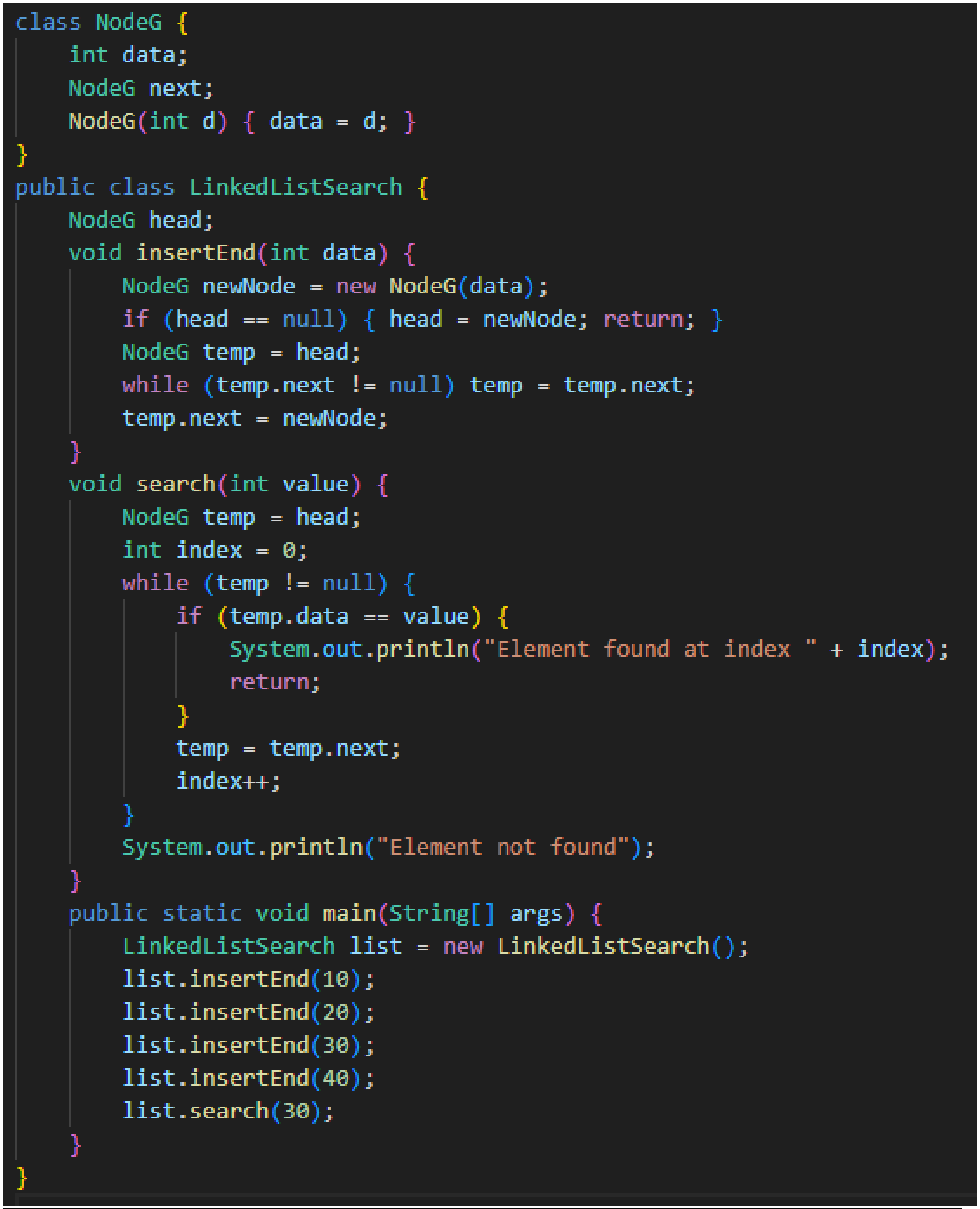


```java
class NodeF {
    int data;
    NodeF next;
    NodeF(int d) { data = d; }
}
public class LinkedListDeleteByValue {
    NodeF head;
    void insertEnd(int data) {
        NodeF newNode = new NodeF(data);
        if (head == null) { head = newNode; return; }
        NodeF temp = head;
        while (temp.next != null) temp = temp.next;
        temp.next = newNode;
    }
    void deleteByValue(int value) {
        if (head == null) return;
        if (head.data == value) { head = head.next; return; }
        NodeF temp = head;
        while (temp.next != null && temp.next.data != value) temp = temp.next;
        if (temp.next != null) temp.next = temp.next.next;
    }
    void display() {
        NodeF temp = head;
        while (temp != null) {
            System.out.print(temp.data);
            if (temp.next != null) System.out.print(" → ");
            temp = temp.next;
        }
        System.out.println();
    }
    public static void main(String[] args) {
        LinkedListDeleteByValue list = new LinkedListDeleteByValue();
        list.insertEnd(10);
        list.insertEnd(20);
        list.insertEnd(30);
        list.insertEnd(40);
        list.deleteByValue(30);
        list.display();
    }
}
```

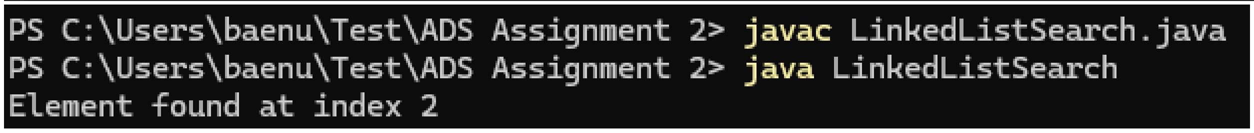


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac LinkedListDeleteByValue.java
PS C:\Users\baenu\Test\ADS Assignment 2> java LinkedListDeleteByValue
10 ? 20 ? 40
```

17. Search for an element in a LinkedList.
Testcase:
Existing LinkedList: [10, 20, 30, 40]
Element to search: 30
Expected Output: Element found at index 2

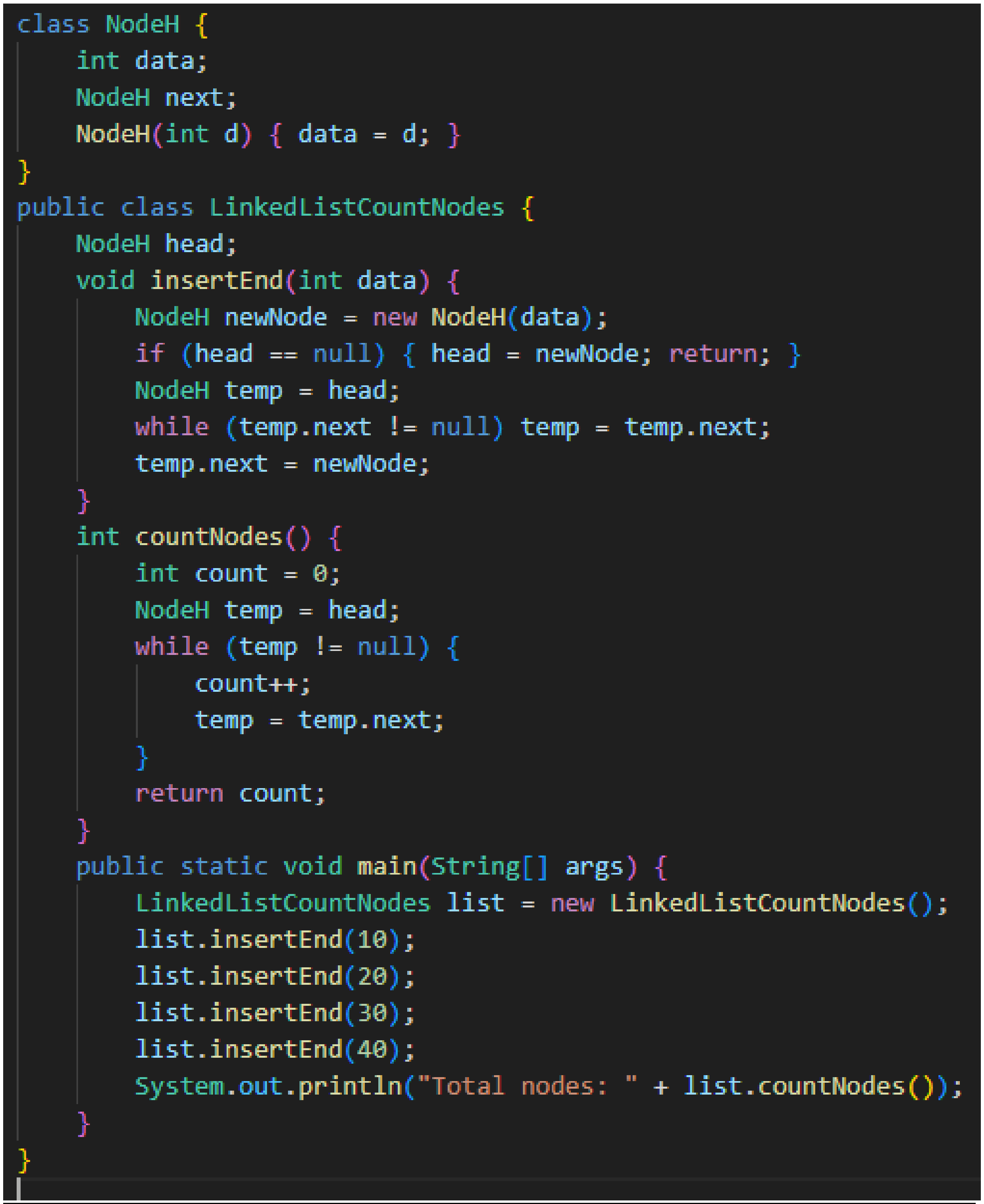


```java
class NodeG {
    int data;
    NodeG next;
    NodeG(int d) { data = d; }
}
public class LinkedListSearch {
    NodeG head;
    void insertEnd(int data) {
        NodeG newNode = new NodeG(data);
        if (head == null) { head = newNode; return; }
        NodeG temp = head;
        while (temp.next != null) temp = temp.next;
        temp.next = newNode;
    }
    void search(int value) {
        NodeG temp = head;
        int index = 0;
        while (temp != null) {
            if (temp.data == value) {
                System.out.println("Element found at index " + index);
                return;
            }
            temp = temp.next;
            index++;
        }
        System.out.println("Element not found");
    }
    public static void main(String[] args) {
        LinkedListSearch list = new LinkedListSearch();
        list.insertEnd(10);
        list.insertEnd(20);
        list.insertEnd(30);
        list.insertEnd(40);
        list.search(30);
    }
}
```

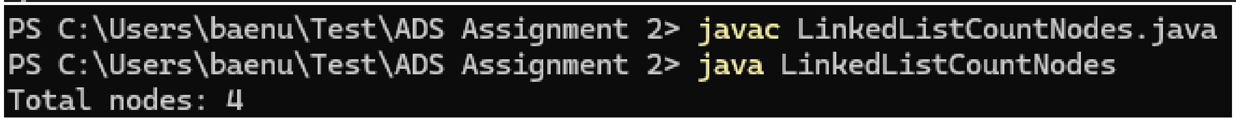


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac LinkedListSearch.java
PS C:\Users\baenu\Test\ADS Assignment 2> java LinkedListSearch
Element found at index 2
```

18. Count the total number of nodes in a LinkedList.
Testcase:
Existing LinkedList: [10, 20, 30, 40]
Expected Output: Total nodes: 4

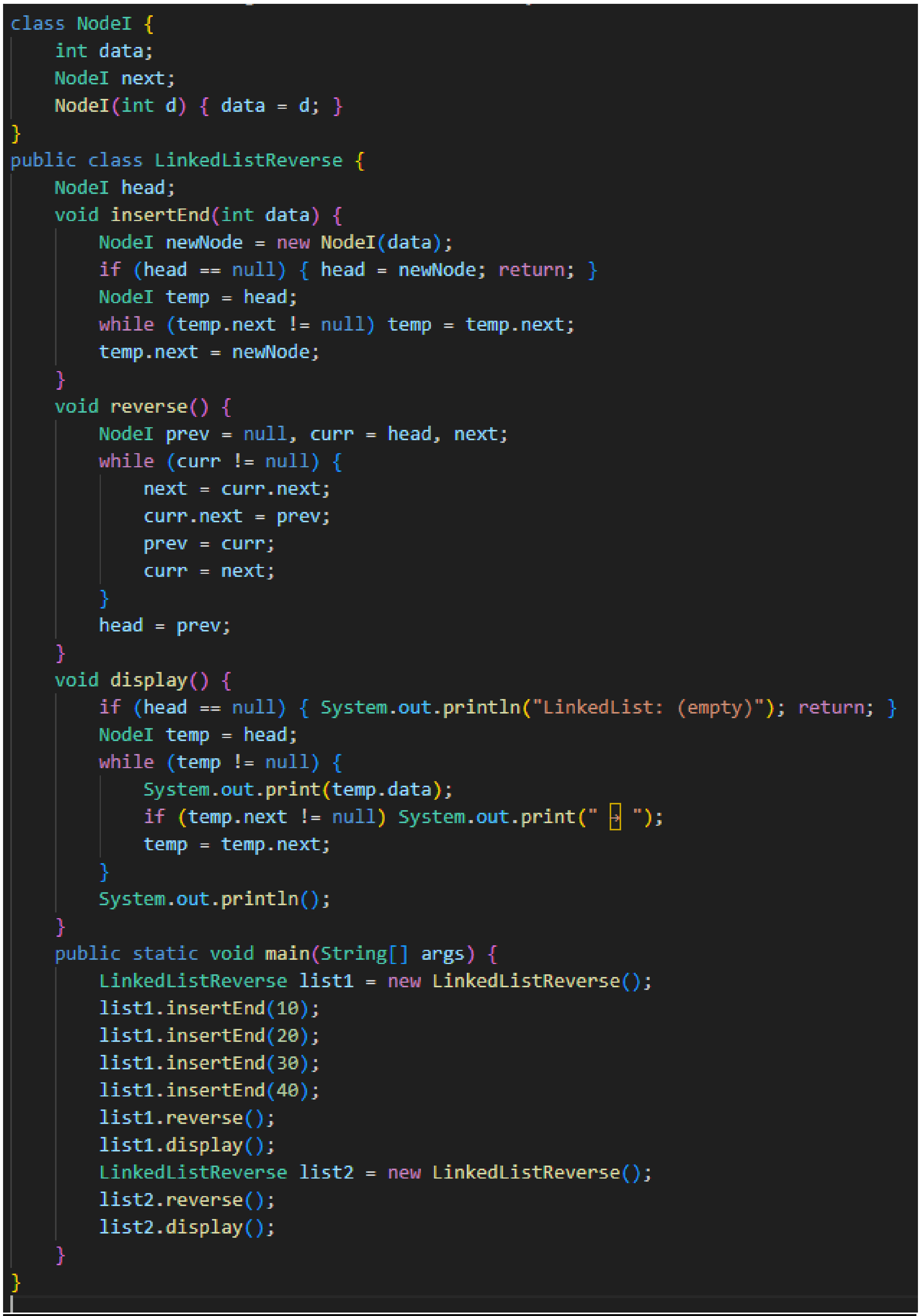


```java
class NodeH {
    int data;
    NodeH next;
    NodeH(int d) { data = d; }
}
public class LinkedListCountNodes {
    NodeH head;
    void insertEnd(int data) {
        NodeH newNode = new NodeH(data);
        if (head == null) { head = newNode; return; }
        NodeH temp = head;
        while (temp.next != null) temp = temp.next;
        temp.next = newNode;
    }
    int countNodes() {
        int count = 0;
        NodeH temp = head;
        while (temp != null) {
            count++;
            temp = temp.next;
        }
        return count;
    }
    public static void main(String[] args) {
        LinkedListCountNodes list = new LinkedListCountNodes();
        list.insertEnd(10);
        list.insertEnd(20);
        list.insertEnd(30);
        list.insertEnd(40);
        System.out.println("Total nodes: " + list.countNodes());
    }
}
```

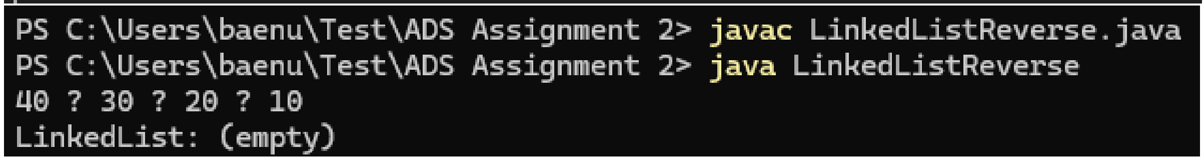


```
PS C:\Users\baenu\Test\ADS Assignment 2> javac LinkedListCountNodes.java
PS C:\Users\baenu\Test\ADS Assignment 2> java LinkedListCountNodes
Total nodes: 4
```

19. Reverse a LinkedList.
Testcase:
Existing LinkedList: [10, 20, 30, 40]
Expected Output: LinkedList: 40 → 30 → 20 → 10
Existing LinkedList: []
Expected Output: LinkedList: (empty)



```java
class NodeI {
    int data;
    NodeI next;
    NodeI(int d) { data = d; }
}
public class LinkedListReverse {
    NodeI head;
    void insertEnd(int data) {
        NodeI newNode = new NodeI(data);
        if (head == null) { head = newNode; return; }
        NodeI temp = head;
        while (temp.next != null) temp = temp.next;
        temp.next = newNode;
    }
    void reverse() {
        NodeI prev = null, curr = head, next;
        while (curr != null) {
            next = curr.next;
            curr.next = prev;
            prev = curr;
            curr = next;
        }
        head = prev;
    }
    void display() {
        if (head == null) { System.out.println("LinkedList: (empty)"); return; }
        NodeI temp = head;
        while (temp != null) {
            System.out.print(temp.data);
            if (temp.next != null) System.out.print(" → ");
            temp = temp.next;
        }
        System.out.println();
    }
    public static void main(String[] args) {
        LinkedListReverse list1 = new LinkedListReverse();
        list1.insertEnd(10);
        list1.insertEnd(20);
        list1.insertEnd(30);
        list1.insertEnd(40);
        list1.reverse();
        list1.display();
        LinkedListReverse list2 = new LinkedListReverse();
        list2.reverse();
        list2.display();
    }
}
```



```
PS C:\Users\baenu\Test\ADS Assignment 2> javac LinkedListReverse.java
PS C:\Users\baenu\Test\ADS Assignment 2> java LinkedListReverse
40 ? 30 ? 20 ? 10
LinkedList: (empty)
```