public class LinkedList {

// Node class to represent each element in the list

static class Node {

int data;

Node next;

// Constructor to initialize the node

public Node(int data) {

this.data = data;

this.next = null;

}

}

// Method to check if the linked list has a cycle using Floyd's Tortoise and Hare algorithm

public static boolean hasCycle(Node head) {

if (head == null) return false;

Node slowPointer = head;

Node fastPointer = head;

while (fastPointer != null && fastPointer.next != null) {

slowPointer = slowPointer.next; // Move slow pointer one step

fastPointer = fastPointer.next.next; // Move fast pointer two steps

if (slowPointer == fastPointer) {

return true; // Cycle detected

}

}

return false; // No cycle detected

}

public static void main(String[] args) {

// Example usage

LinkedList list = new LinkedList();

// Creating a simple linked list with no cycle: 1 -> 2 -> 3 -> 4 -> null

Node head = new Node(1);

head.next = new Node(2);

head.next.next = new Node(3);

head.next.next.next = new Node(4);

System.out.println("Has cycle? " + hasCycle(head)); // Expected: false

// Creating a linked list with a cycle: 1 -> 2 -> 3 -> 4 -> 2 (cycle)

head.next.next.next.next = head.next; // Creating a cycle

System.out.println("Has cycle? " + hasCycle(head)); // Expected: true

}

}