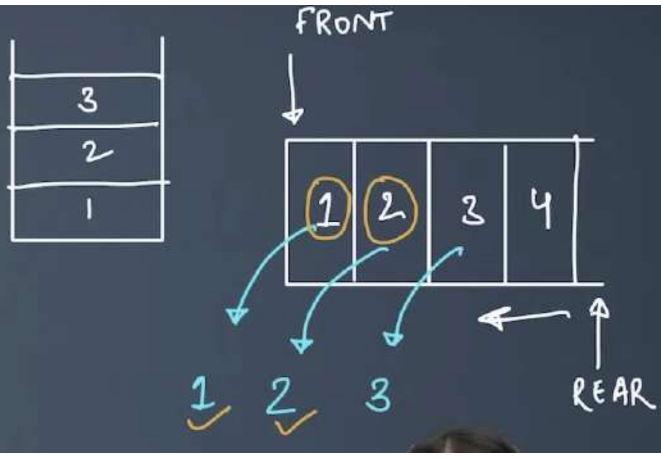
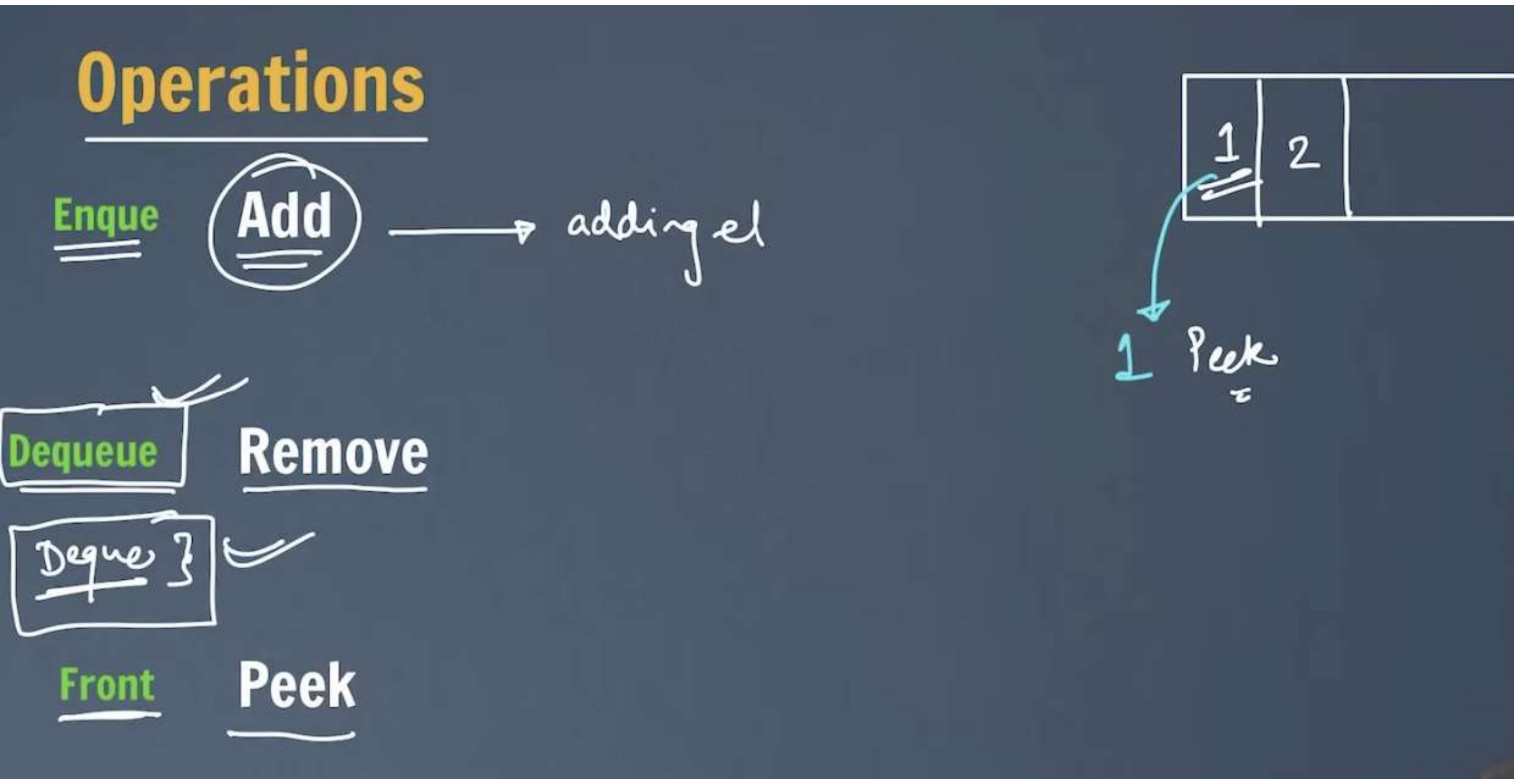
QUEUE

FIFO

First In First Out





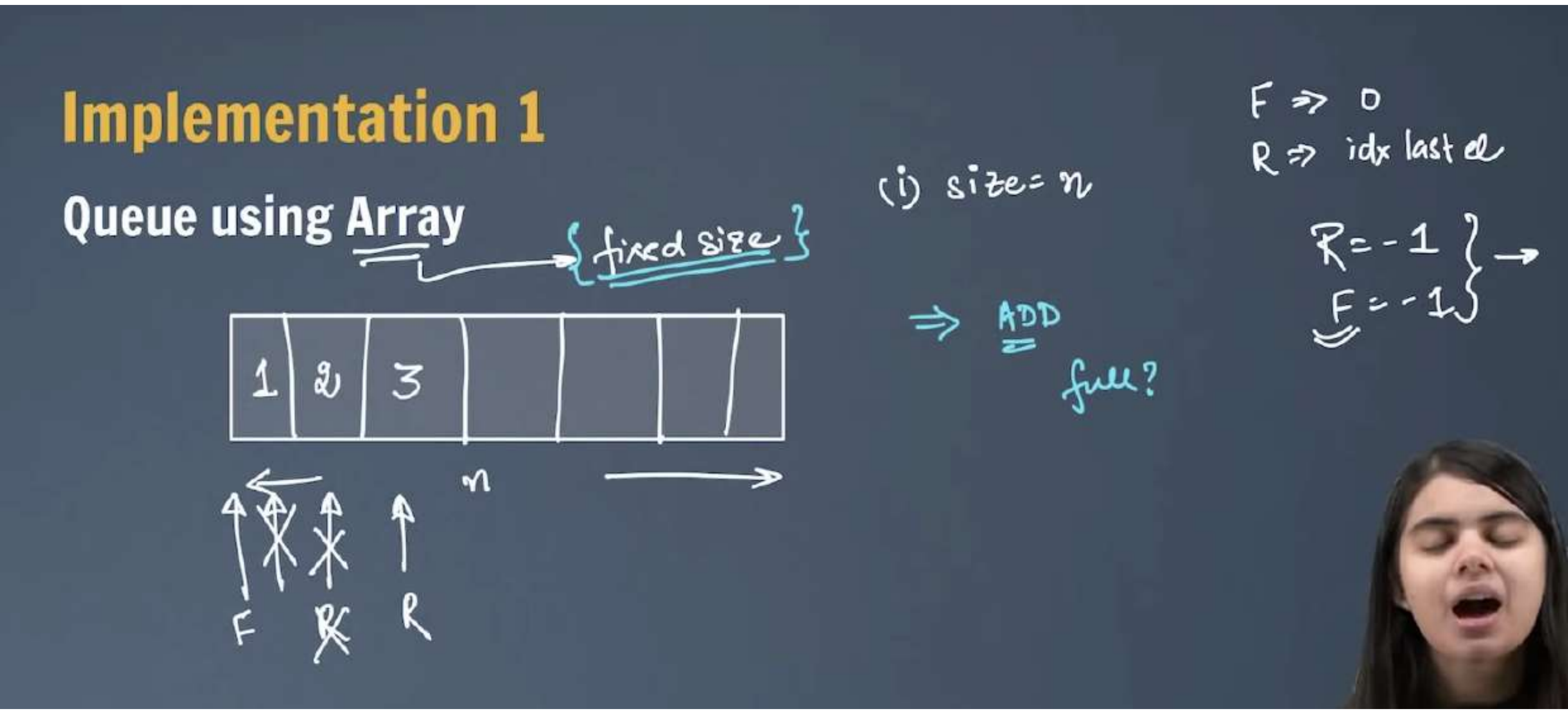
Operations

Enque — $\boxed{\text{Add}}$ $\longrightarrow$ adding el

$\boxed{\text{Dequeue}}$ — Remove

$\boxed{\text{Deque }\}}$ ✓

Front — Peek



Implementation 1

Queue using Array

{fixed size}

(i) size = n

⇒ ADD

full?

F ⇒ 0

R ⇒ idx last el

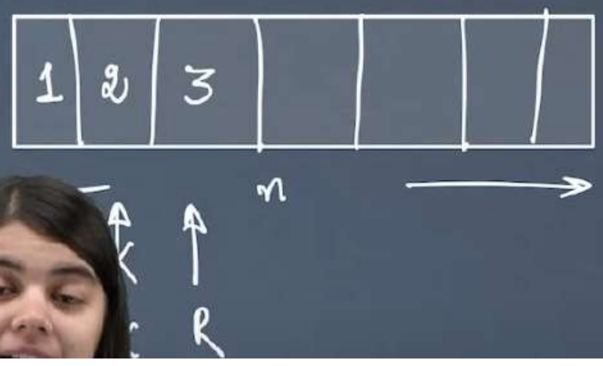
R = -1

F = -1

Implementation 1

Queue using Array

{fixed size}



(i) size = n

$\Rightarrow$ ADD

full?

F ⇒ 0

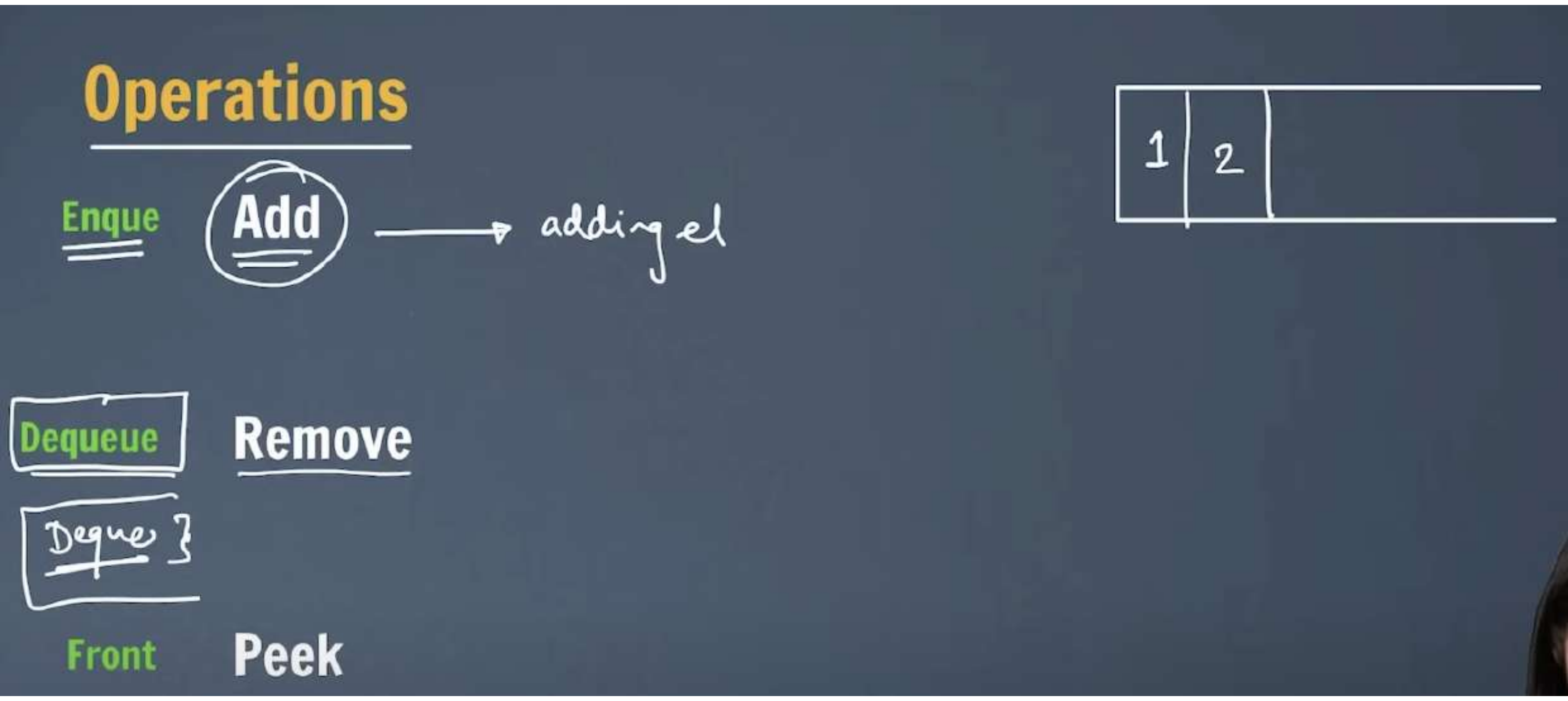
R ⇒ idx last el

$\left.\begin{array}{l} R = -1 \\ F = -1 \end{array}\right\} \rightarrow$

add - $O(1)$

remove - $O(n)$

+ peak



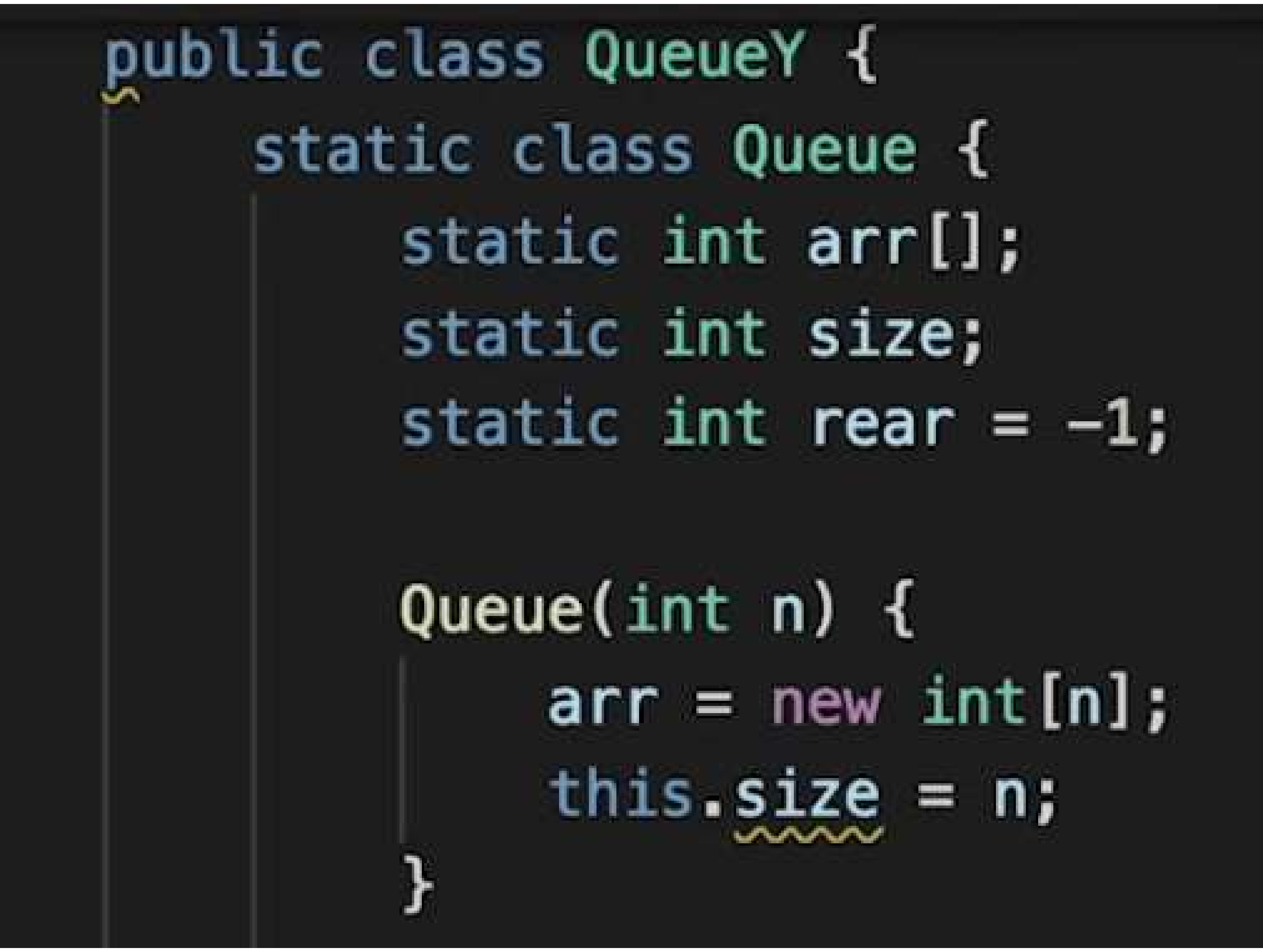
Operations

Enque Add $\longrightarrow$ adding el

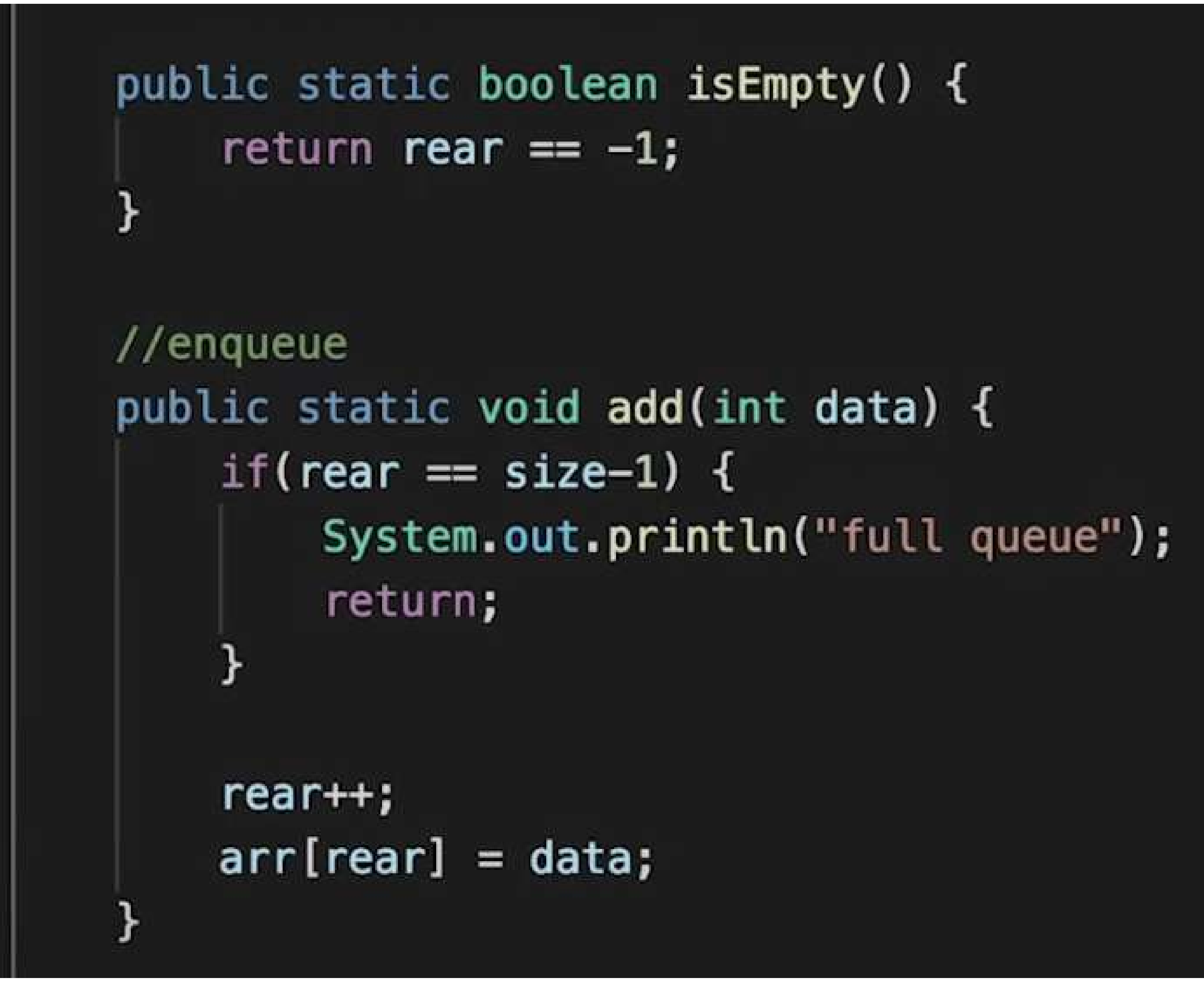
Dequeue Remove

Deque }

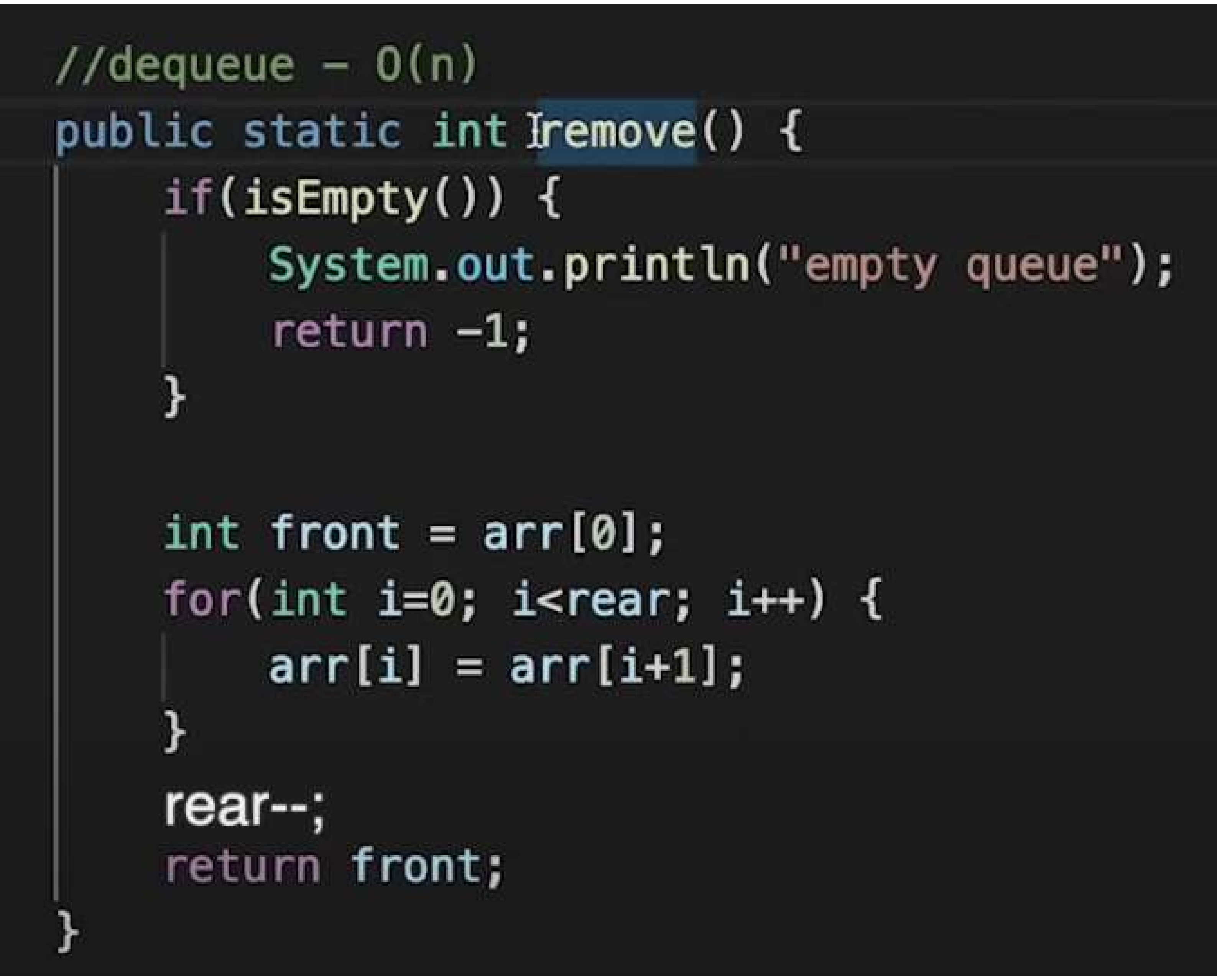
Front Peek



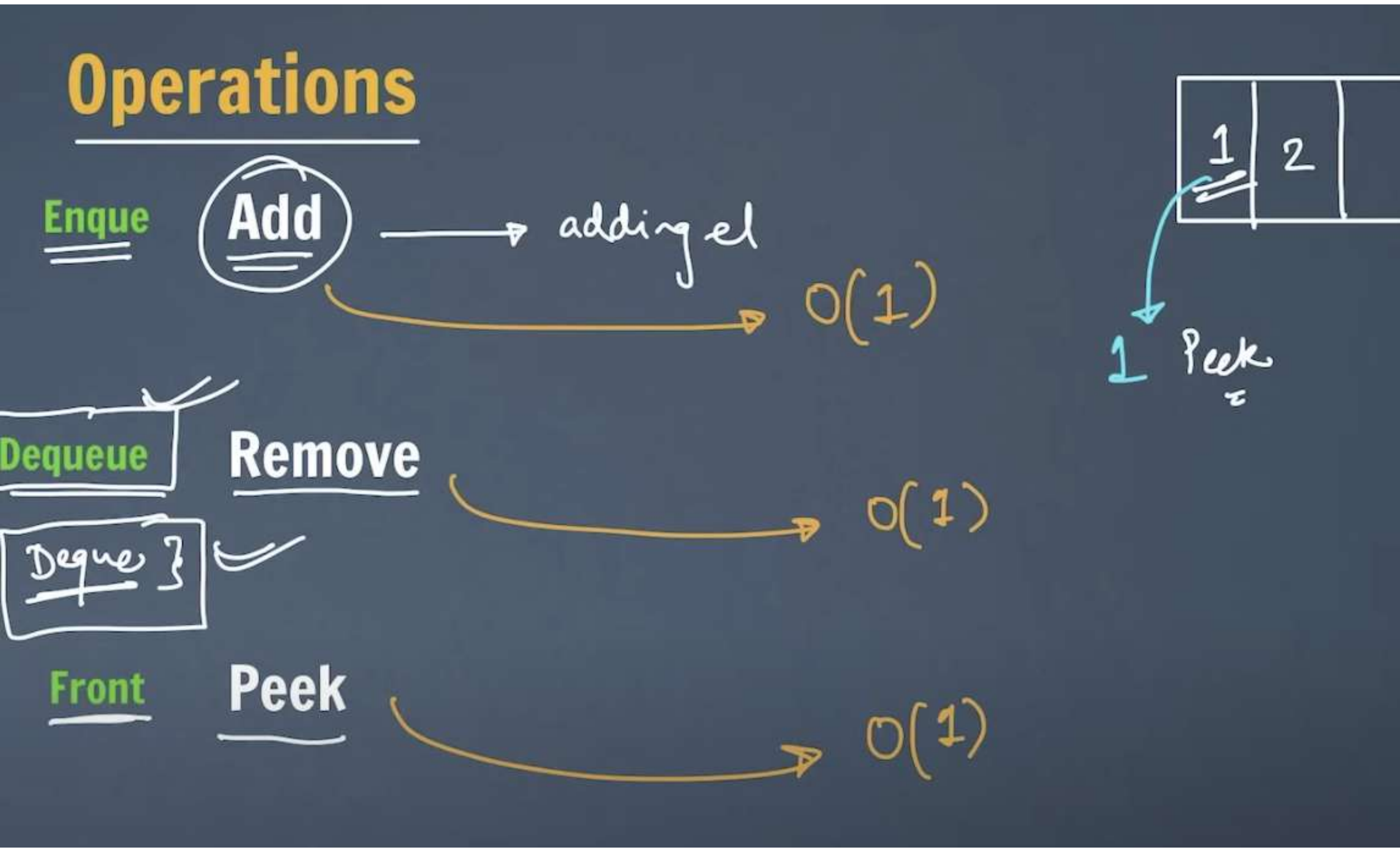
```
public class QueueY {
    static class Queue {
        static int arr[];
        static int size;
        static int rear = -1;

        Queue(int n) {
            arr = new int[n];
            this.size = n;
        }
```



```java
public static boolean isEmpty() {
    return rear == -1;
}

//enqueue
public static void add(int data) {
    if(rear == size-1) {
        System.out.println("full queue");
        return;
    }

    rear++;
    arr[rear] = data;
}
```



```
//dequeue - O(n)
public static int remove() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }

    int front = arr[0];
    for(int i=0; i<rear; i++) {
        arr[i] = arr[i+1];
    }
    rear--;
    return front;
}
```



Operations

Enque Add → adding el

O(1)

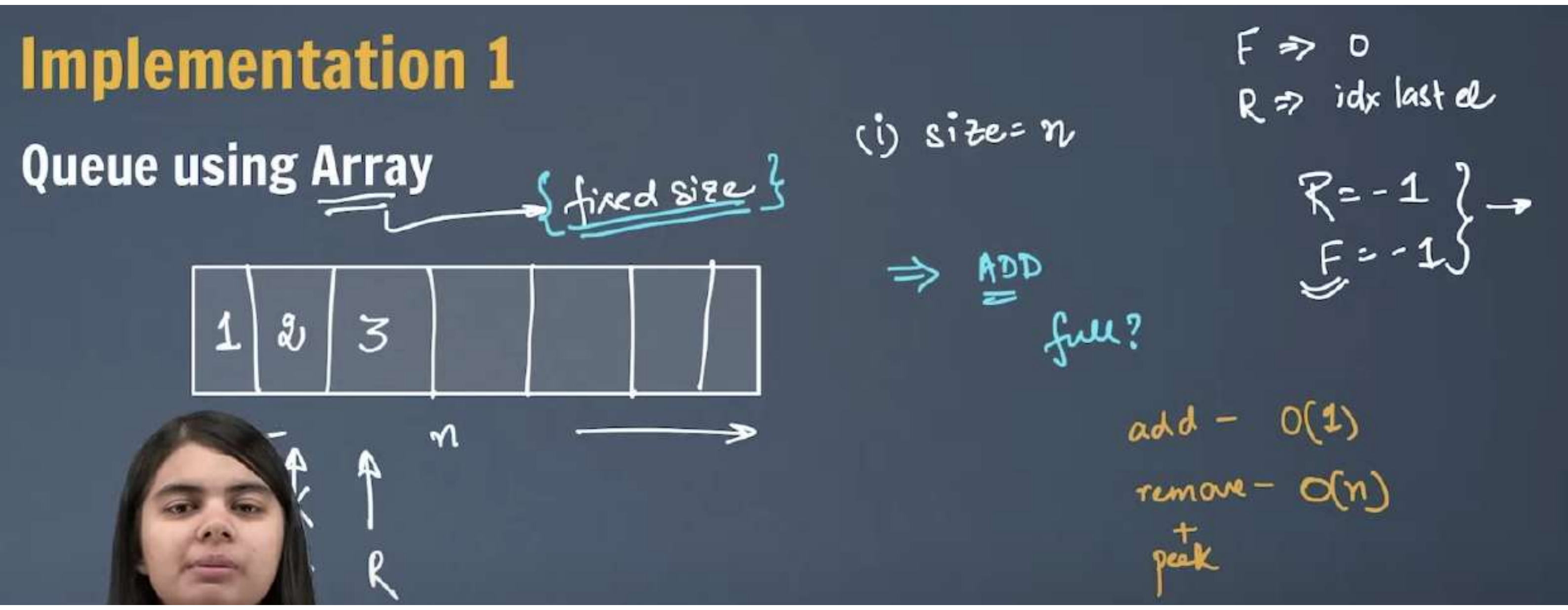
Dequeue Remove

O(1)

Deque }

Front Peek

O(1)



Implementation 1

Queue using Array { fixed size }

(i) size = n

⇒ ADD

full?

F ⇒ 0

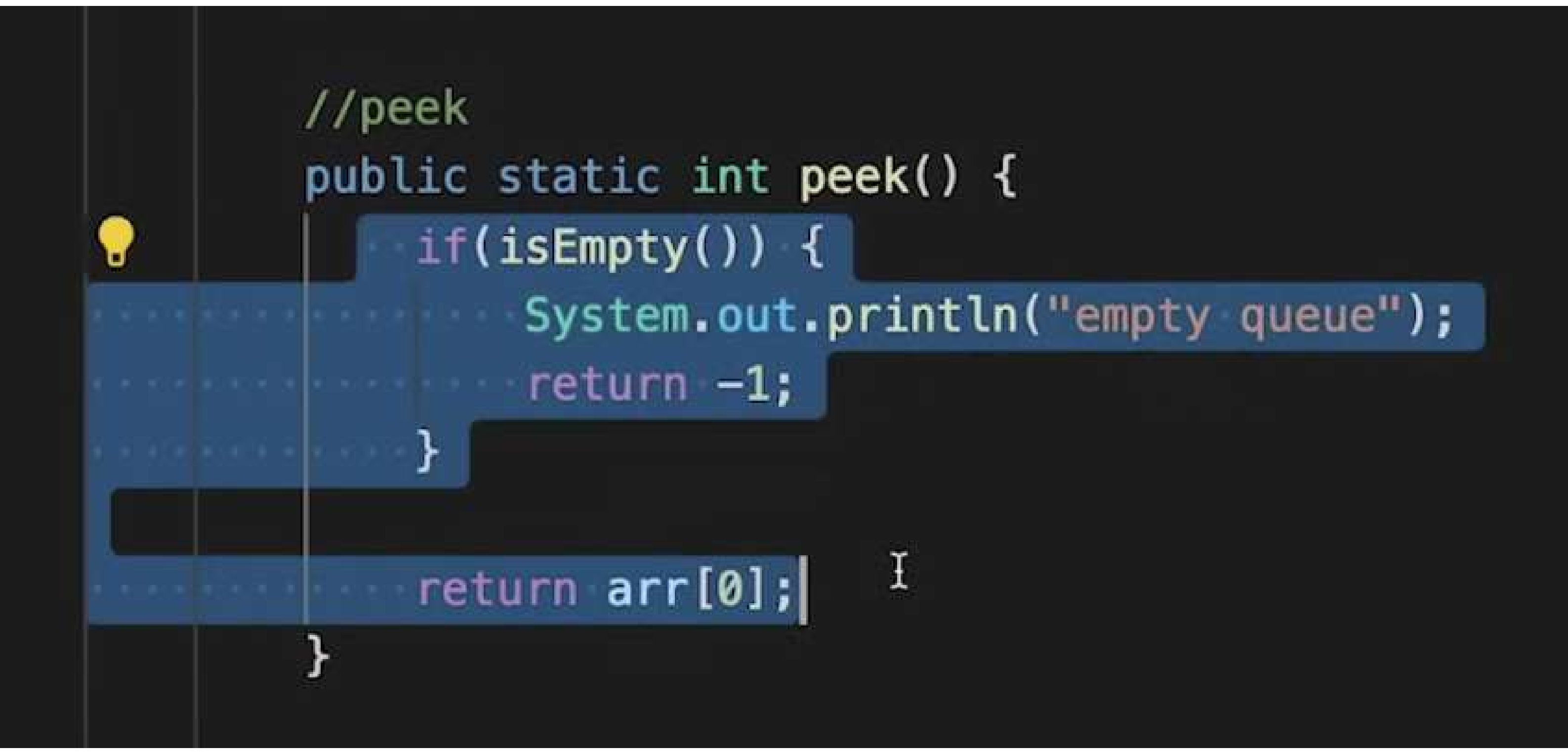
R ⇒ idx last el

$\left.\begin{array}{l} R = -1 \\ F = -1 \end{array}\right\} \rightarrow$

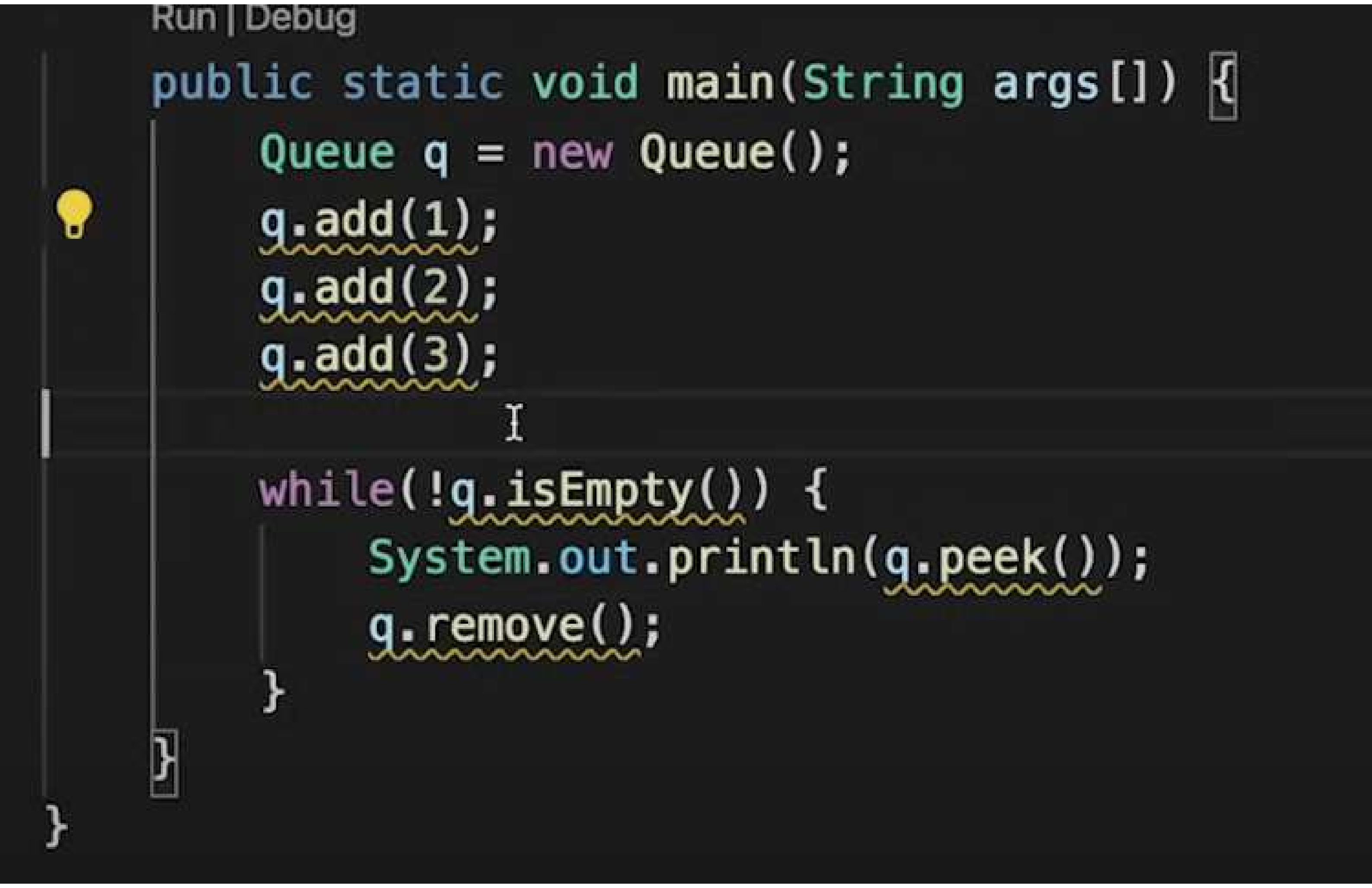
add - O(1)

remove - O(n)

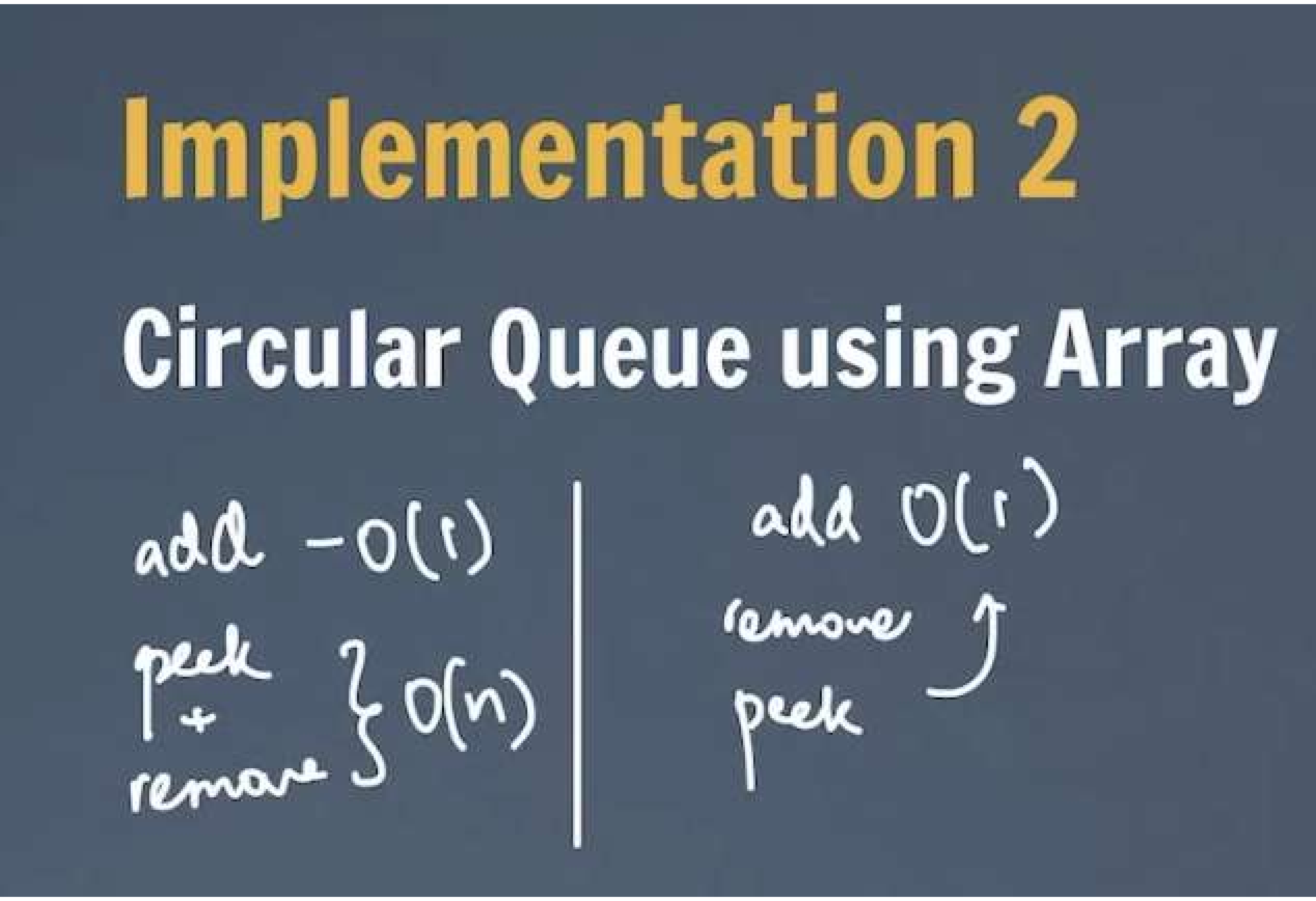
+ peek



```java
//peek
public static int peek() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }

    return arr[0];
}
```



```java
public static void main(String args[]) {
    Queue q = new Queue();
    q.add(1);
    q.add(2);
    q.add(3);

    while(!q.isEmpty()) {
        System.out.println(q.peek());
        q.remove();
    }
}
}
```



Implementation 2

Circular Queue using Array

add – O(1)

peek + remove $\}$ O(n)

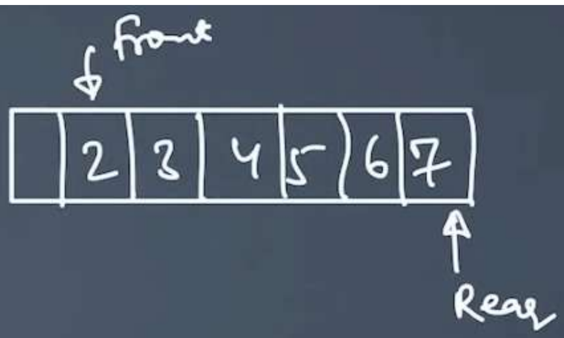
add O(1)

remove ↑

peek

Implementation 2

Circular Queue using Array



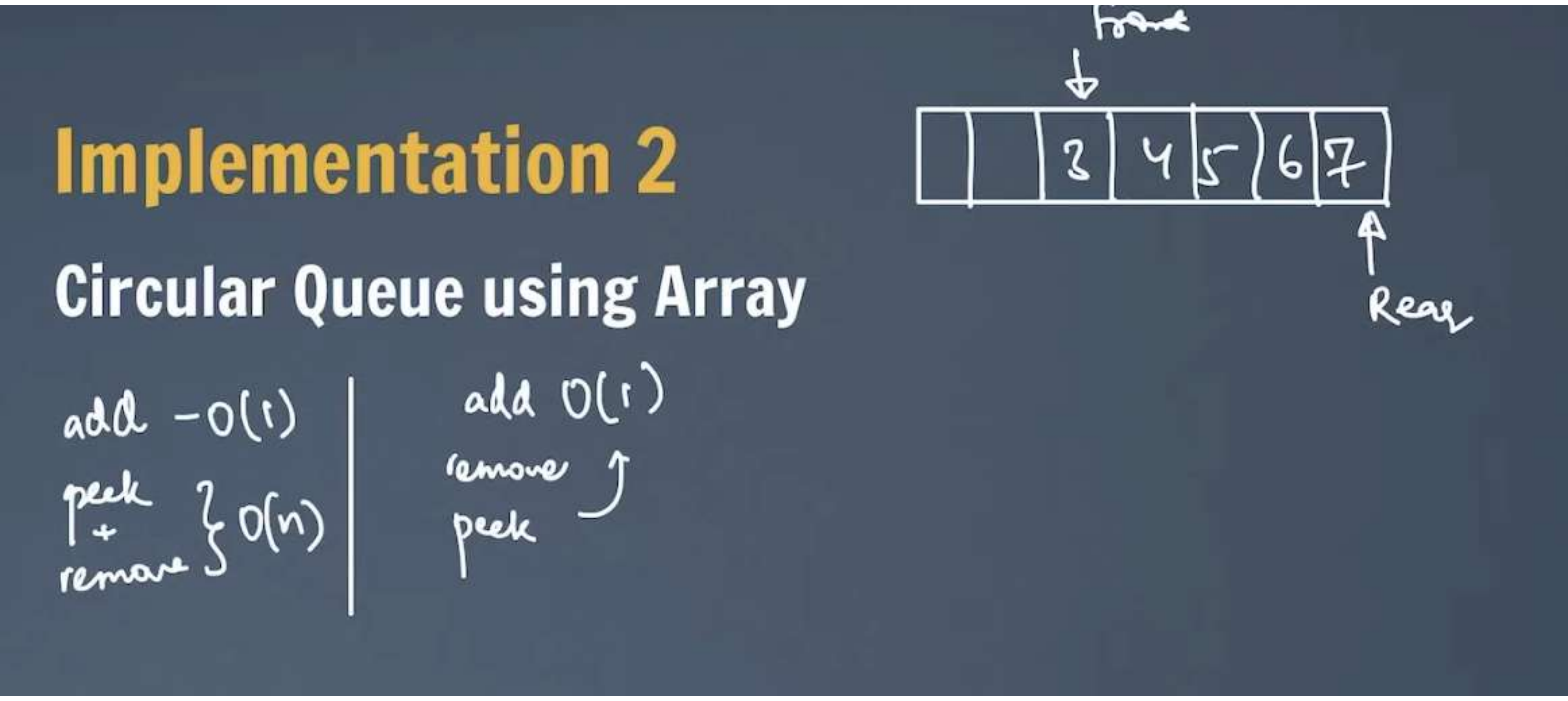
add −O(1)

peek + remove } O(n)

add O(1)

remove ↑

peek



Implementation 2

Circular Queue using Array

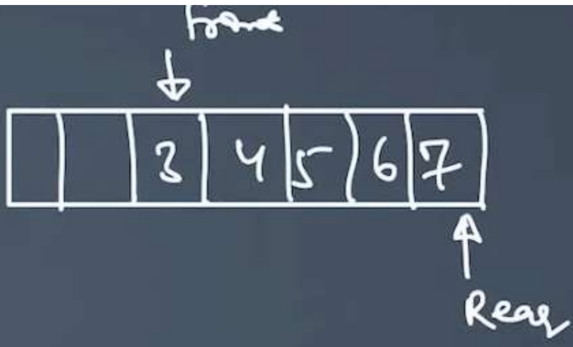
$$\text{add} - O(1)$$

$$\left.\begin{array}{l}\text{peek} \\ + \\ \text{remove}\end{array}\right\} O(n)$$

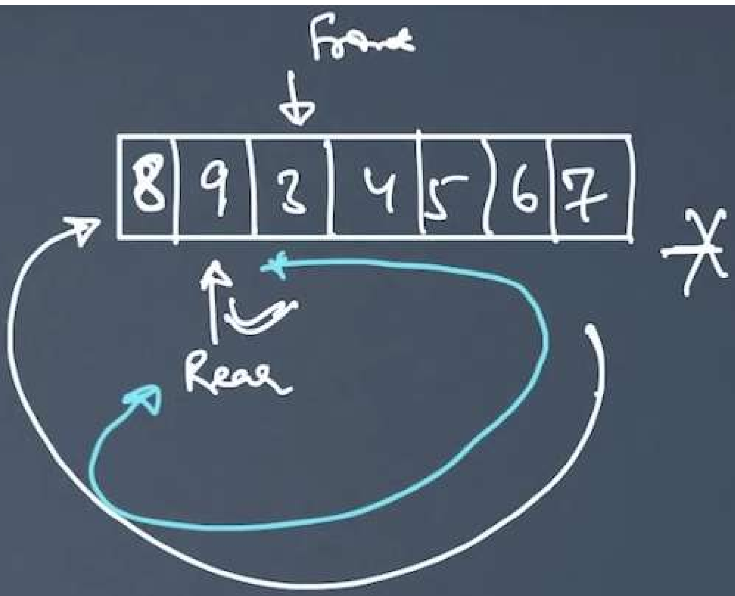
add $O(1)$

remove

peek



ay



8

9

full

$$\boxed{Rear + 1 == front}$$

Rear = -1

Front = -1

Rear++

$$\boxed{Rear = (Rear + 1) \% size}$$

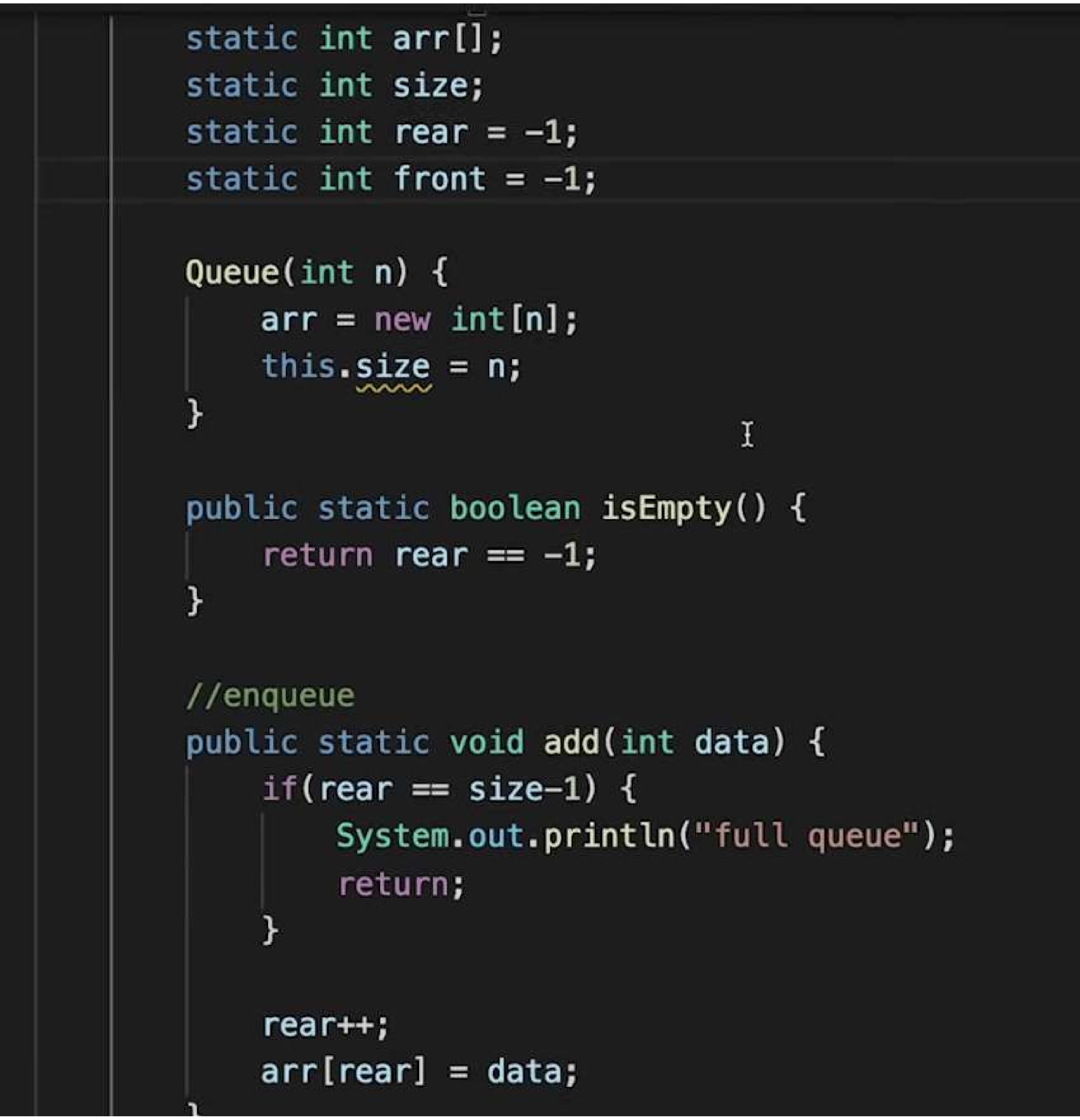
✓ Rear++ X

5

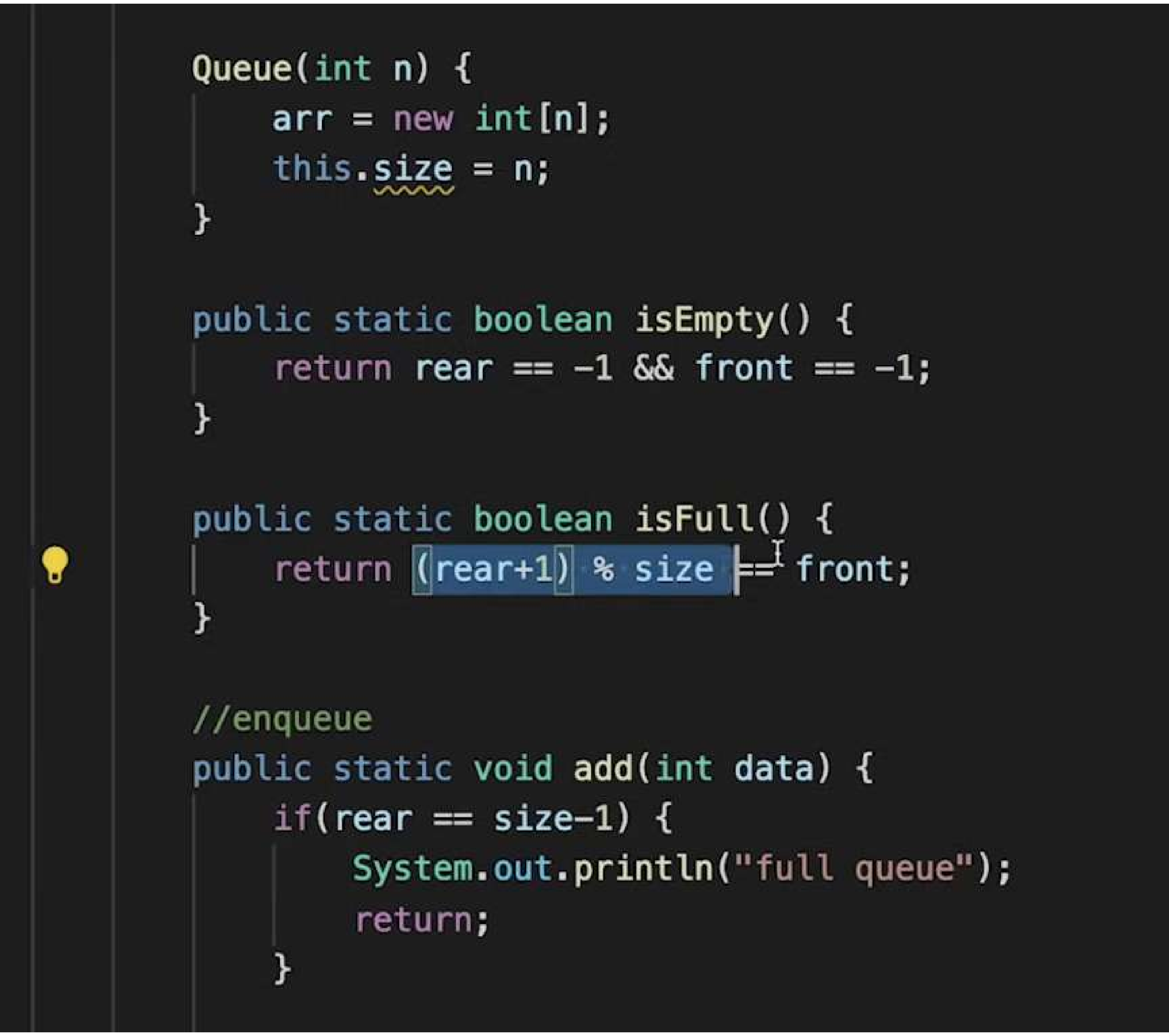
$(5+1) \% 6 \Rightarrow 6 \% 6$

$\Rightarrow 0$

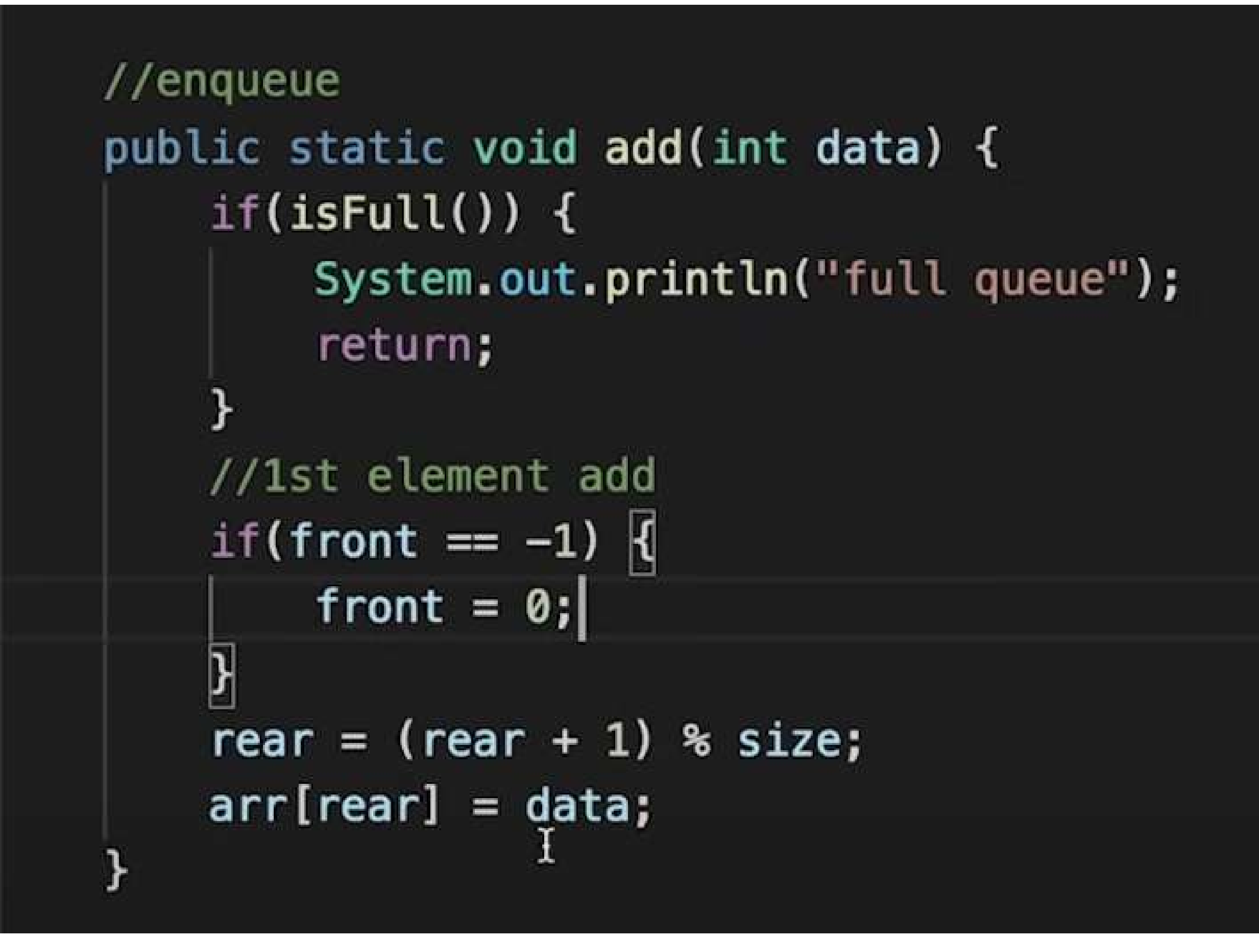
$$\boxed{(Rear + 1) \% size == front}$$



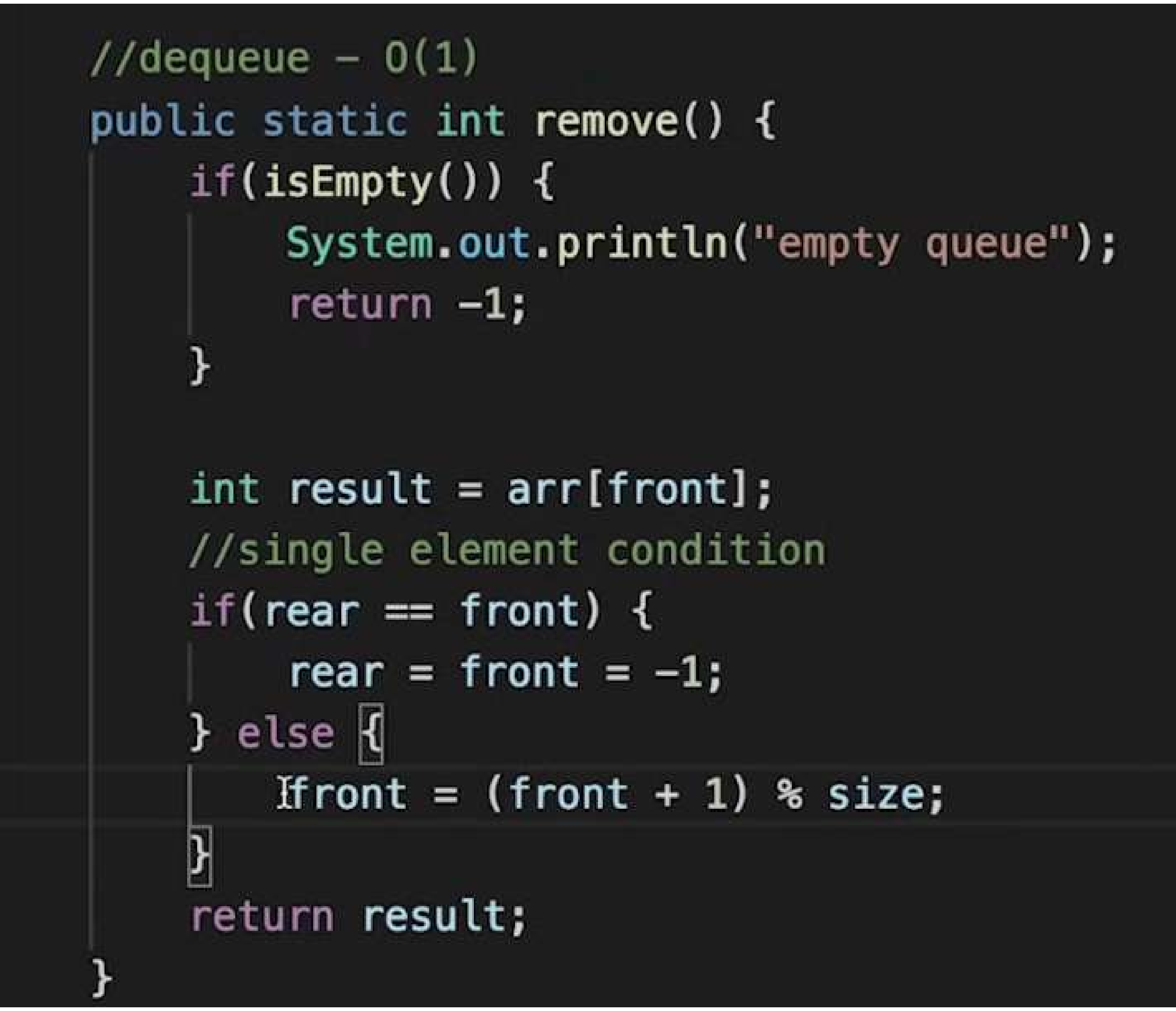
```
static int arr[];
static int size;
static int rear = -1;
static int front = -1;

Queue(int n) {
    arr = new int[n];
    this.size = n;
}

public static boolean isEmpty() {
    return rear == -1;
}

//enqueue
public static void add(int data) {
    if(rear == size-1) {
        System.out.println("full queue");
        return;
    }

    rear++;
    arr[rear] = data;
```



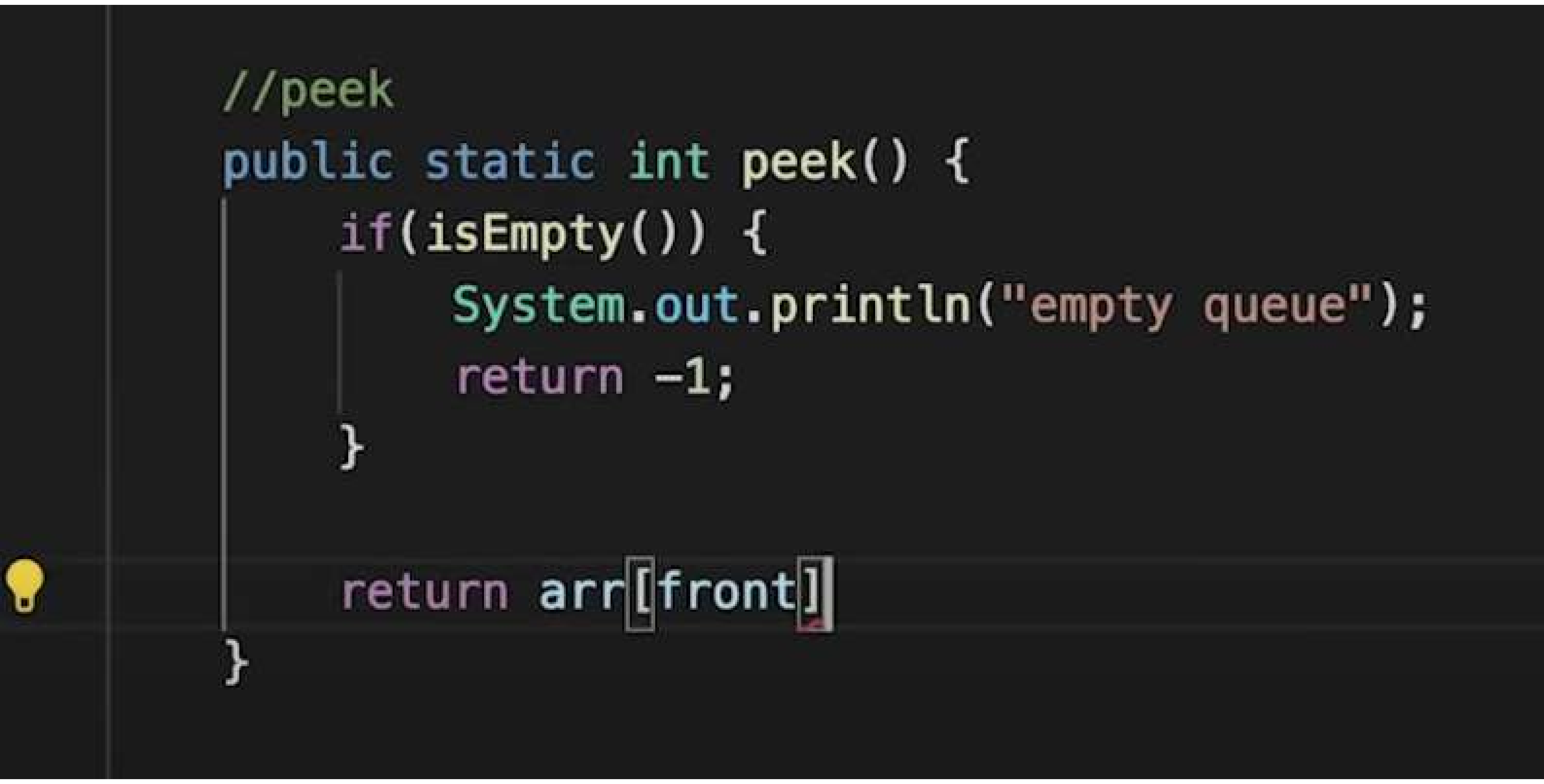
```java
Queue(int n) {
    arr = new int[n];
    this.size = n;
}

public static boolean isEmpty() {
    return rear == -1 && front == -1;
}

public static boolean isFull() {
    return (rear+1) % size == front;
}

//enqueue
public static void add(int data) {
    if(rear == size-1) {
        System.out.println("full queue");
        return;
    }
```



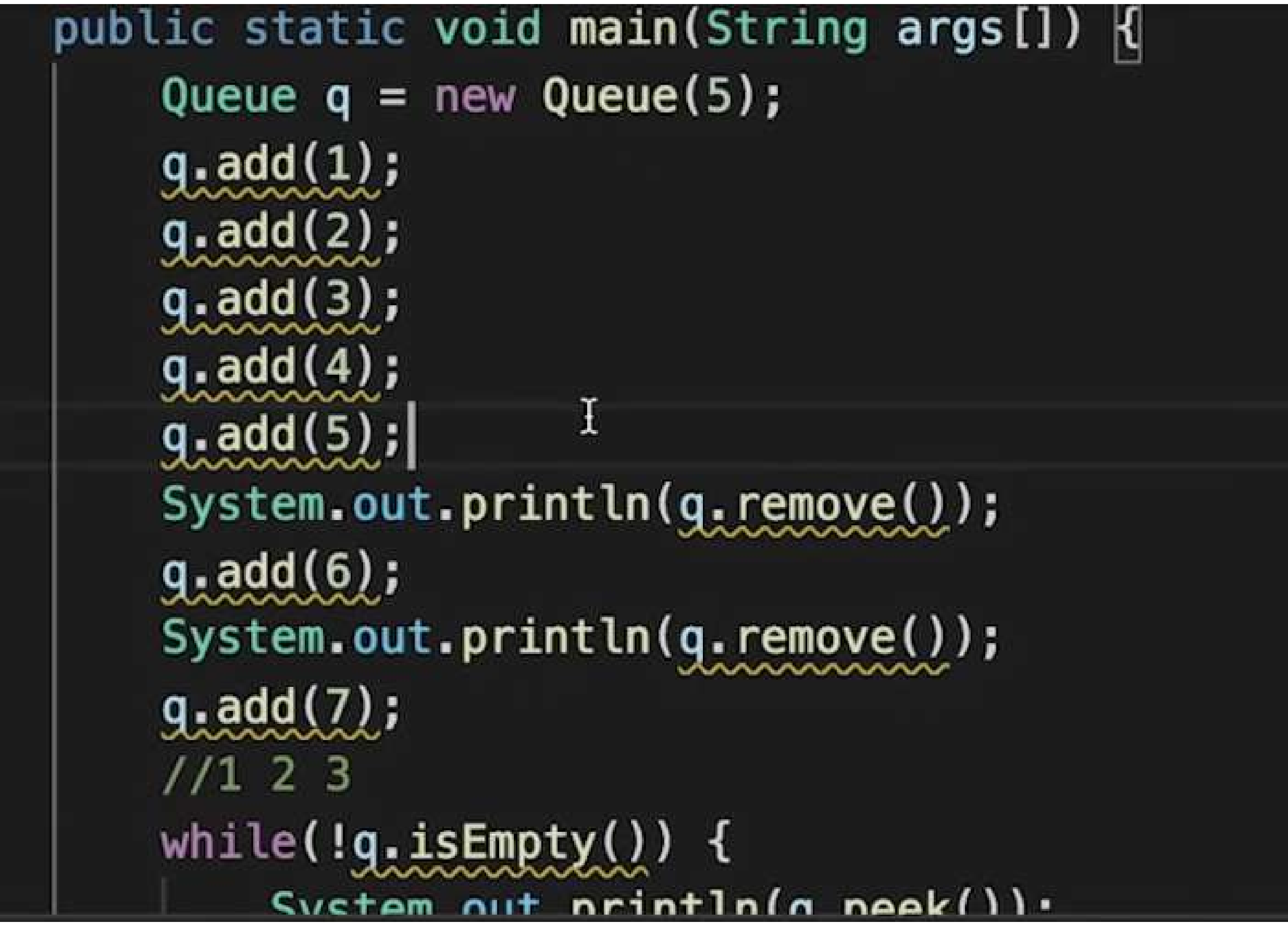
```
//enqueue
public static void add(int data) {
    if(isFull()) {
        System.out.println("full queue");
        return;
    }
    //1st element add
    if(front == -1) {
        front = 0;
    }
    rear = (rear + 1) % size;
    arr[rear] = data;
}
```



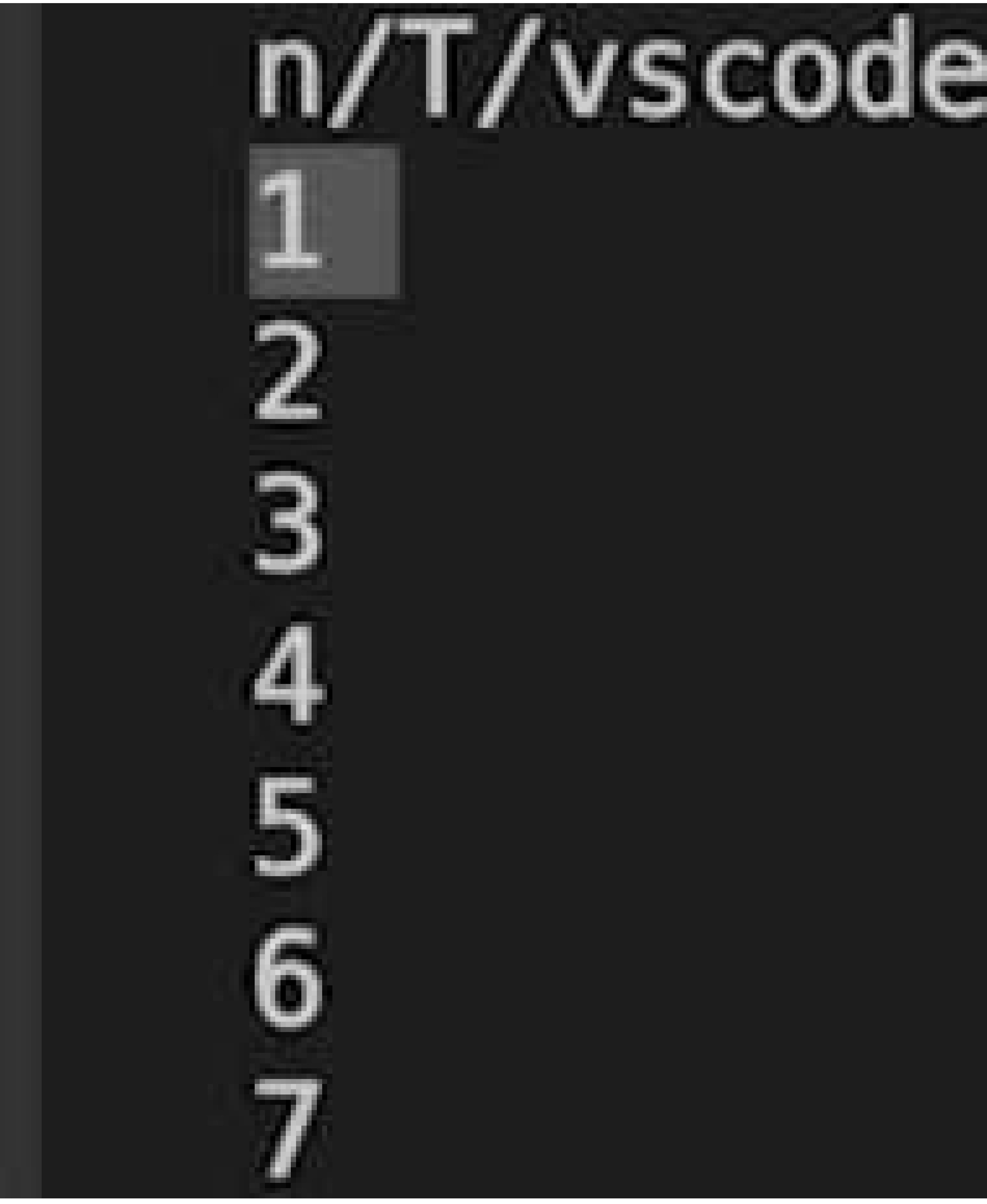
```java
//dequeue - O(1)
public static int remove() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }

    int result = arr[front];
    //single element condition
    if(rear == front) {
        rear = front = -1;
    } else {
        front = (front + 1) % size;
    }
    return result;
}
```



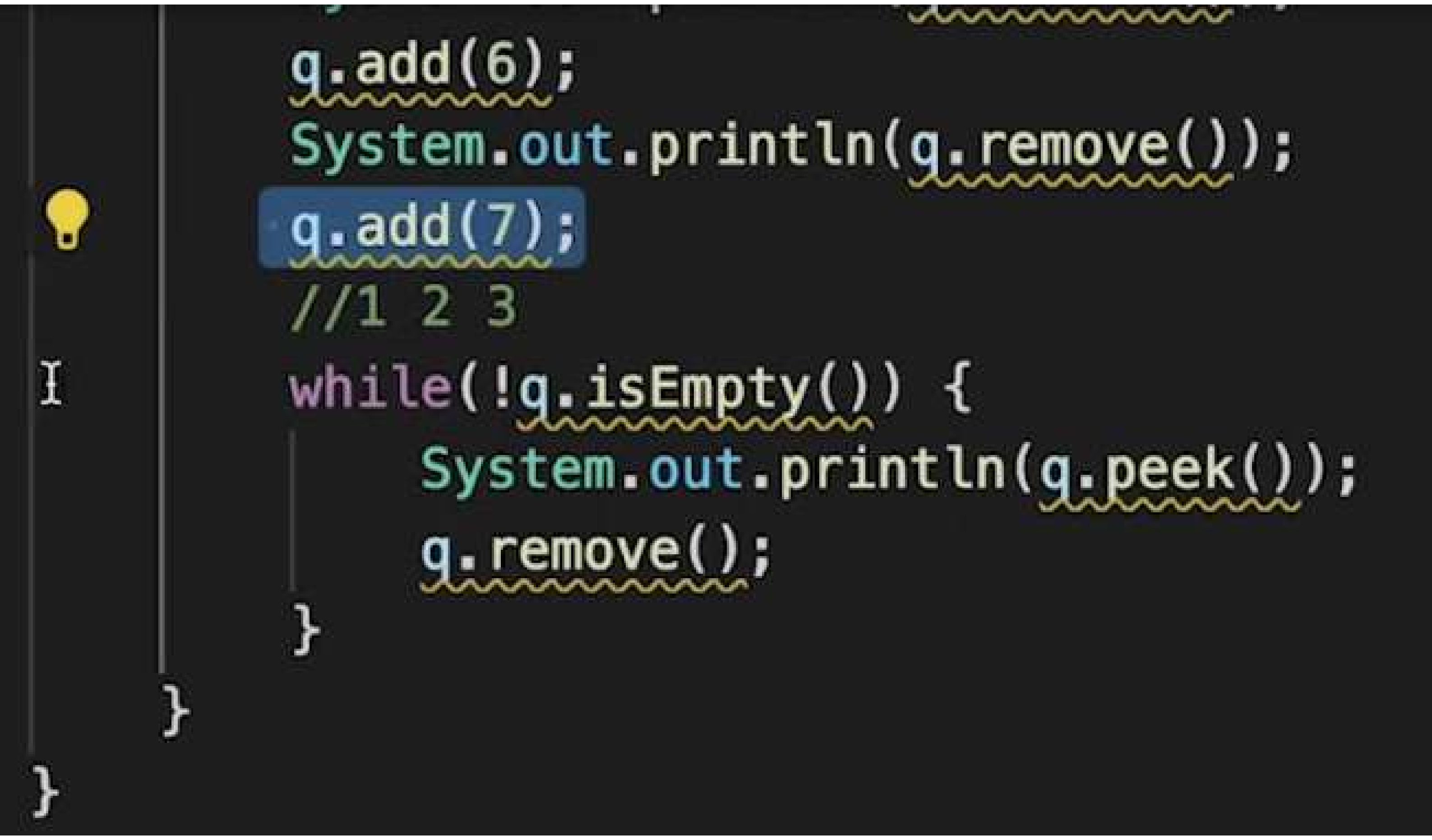
```
//peek
public static int peek() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }

    return arr[front]
}
```



```
public static void main(String args[]) {
    Queue q = new Queue(5);
    q.add(1);
    q.add(2);
    q.add(3);
    q.add(4);
    q.add(5);
    System.out.println(q.remove());
    q.add(6);
    System.out.println(q.remove());
    q.add(7);
    //1 2 3
    while(!q.isEmpty()) {
        System.out.println(q.peek());
```



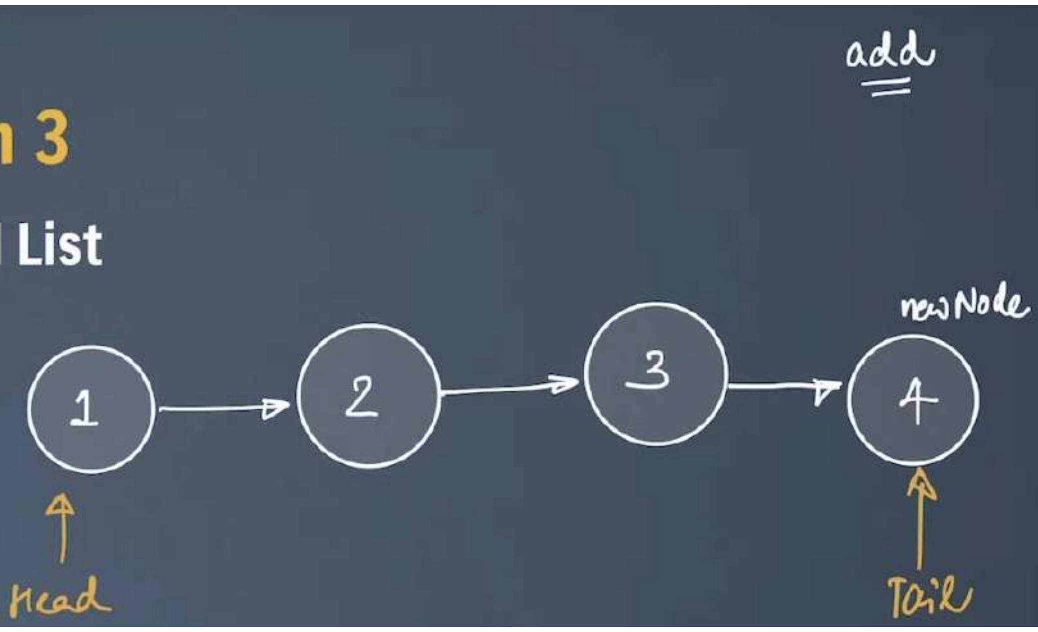
```
n/T/vscode
```

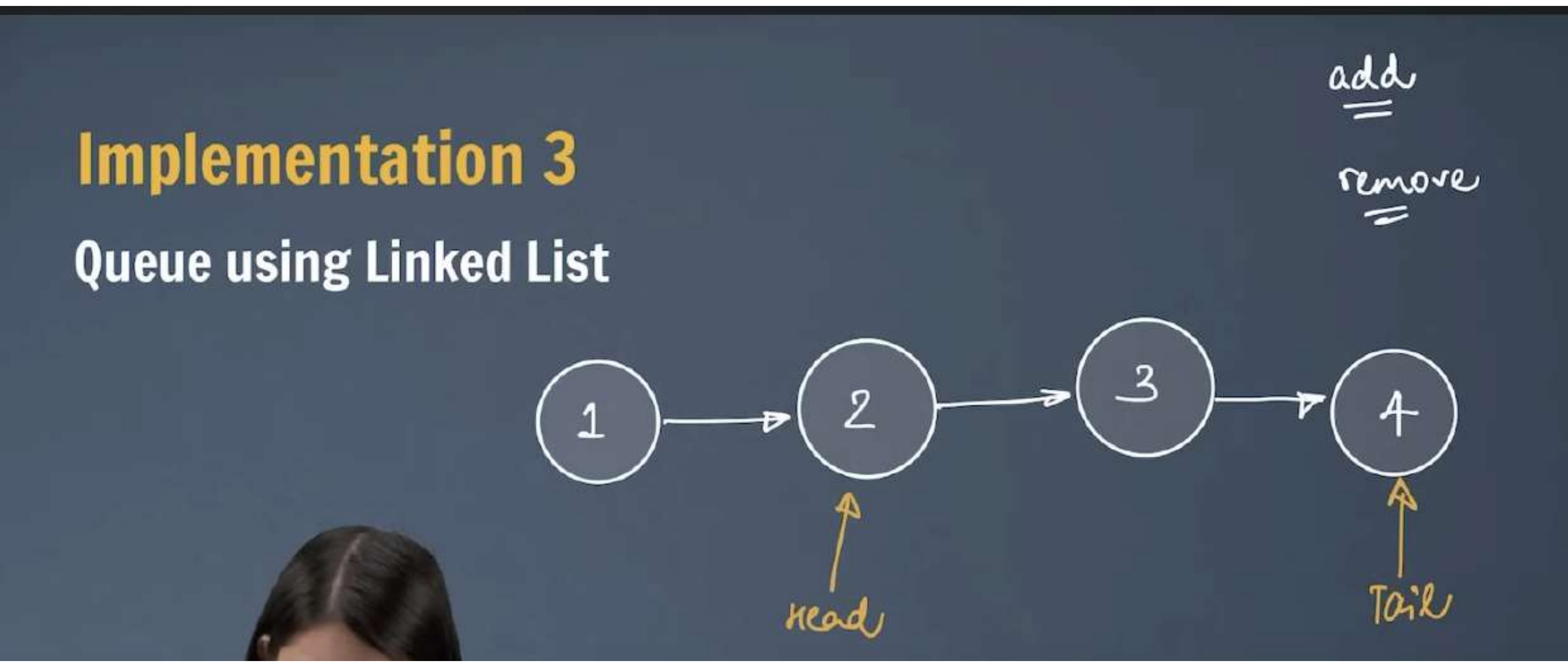


```
        q.add(6);
        System.out.println(q.remove());
        q.add(7);
        //1 2 3
        while(!q.isEmpty()) {
            System.out.println(q.peek());
            q.remove();
        }
    }
}
```

Implementation 3

Queue using Linked List





Implementation 3

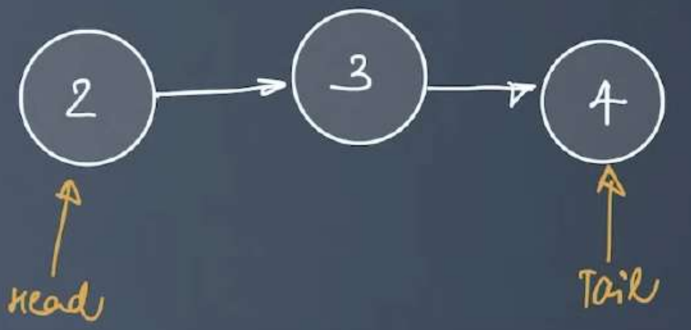
Queue using Linked List

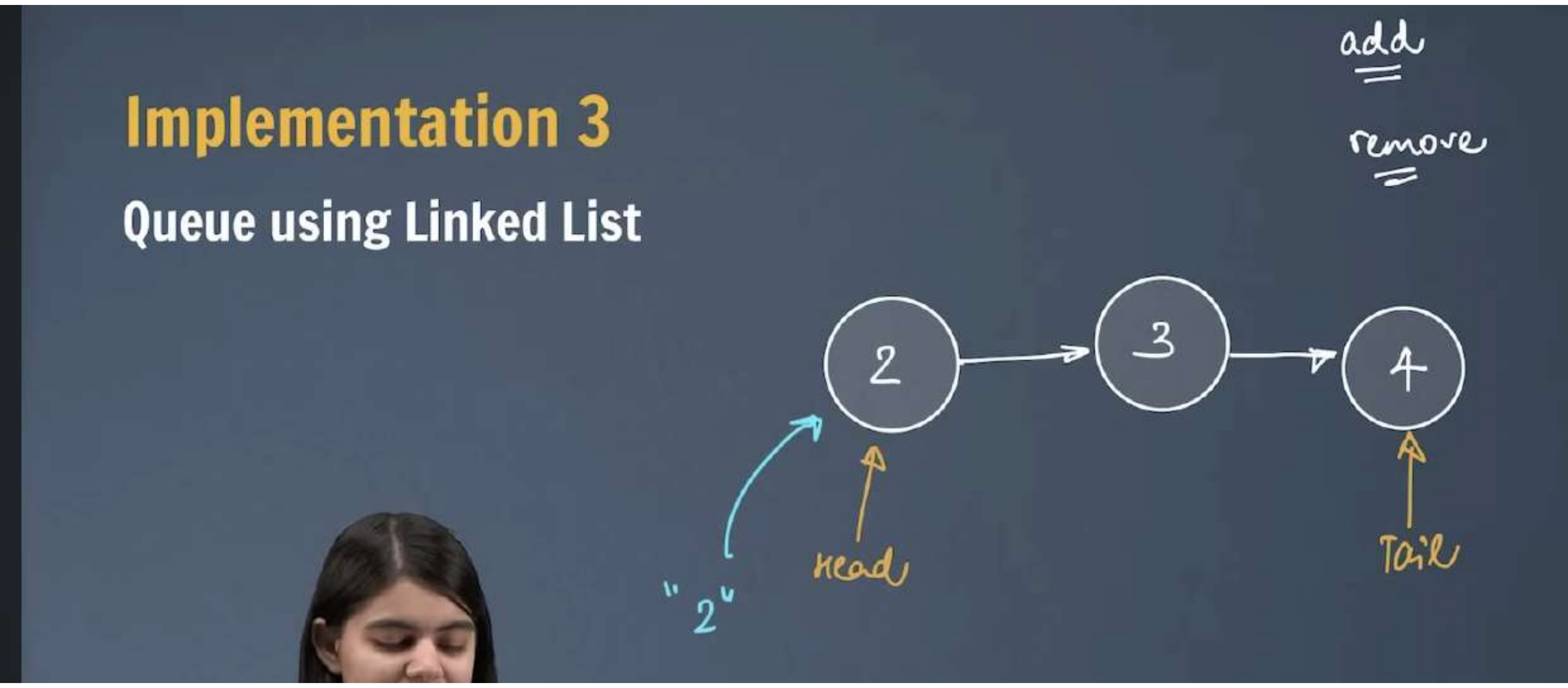
Implementation 3

Queue using Linked List

add

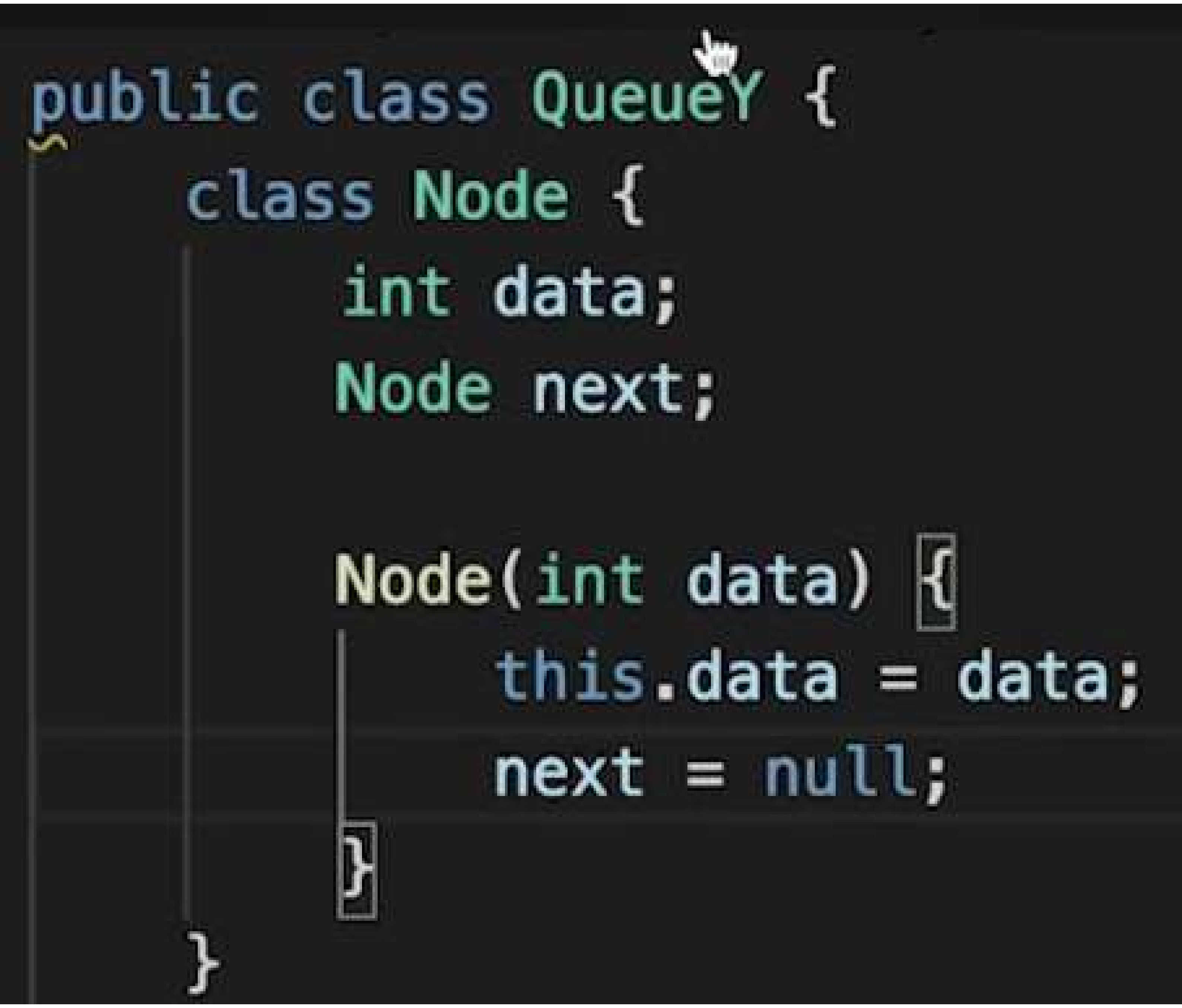
remove



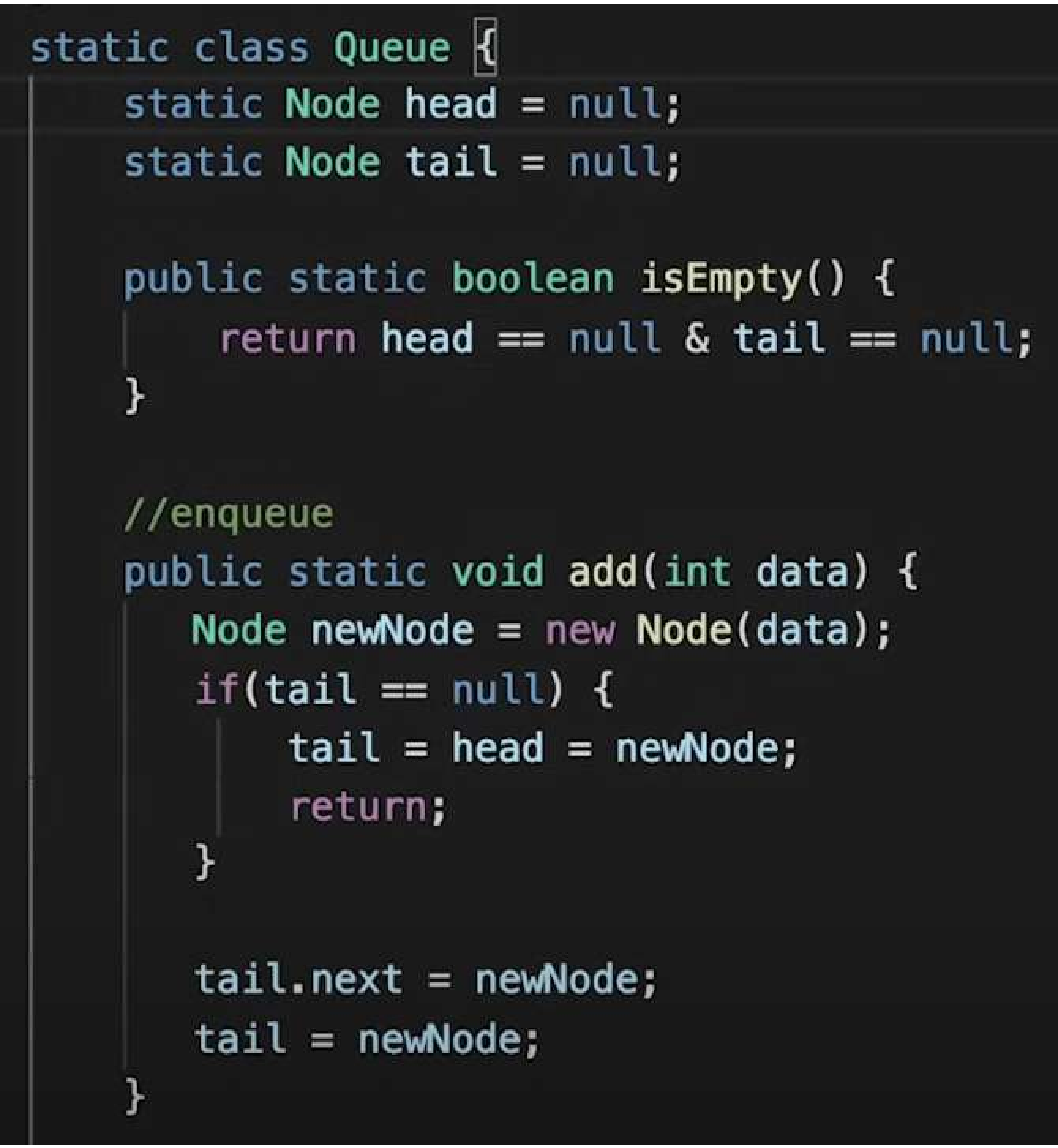


Implementation 3

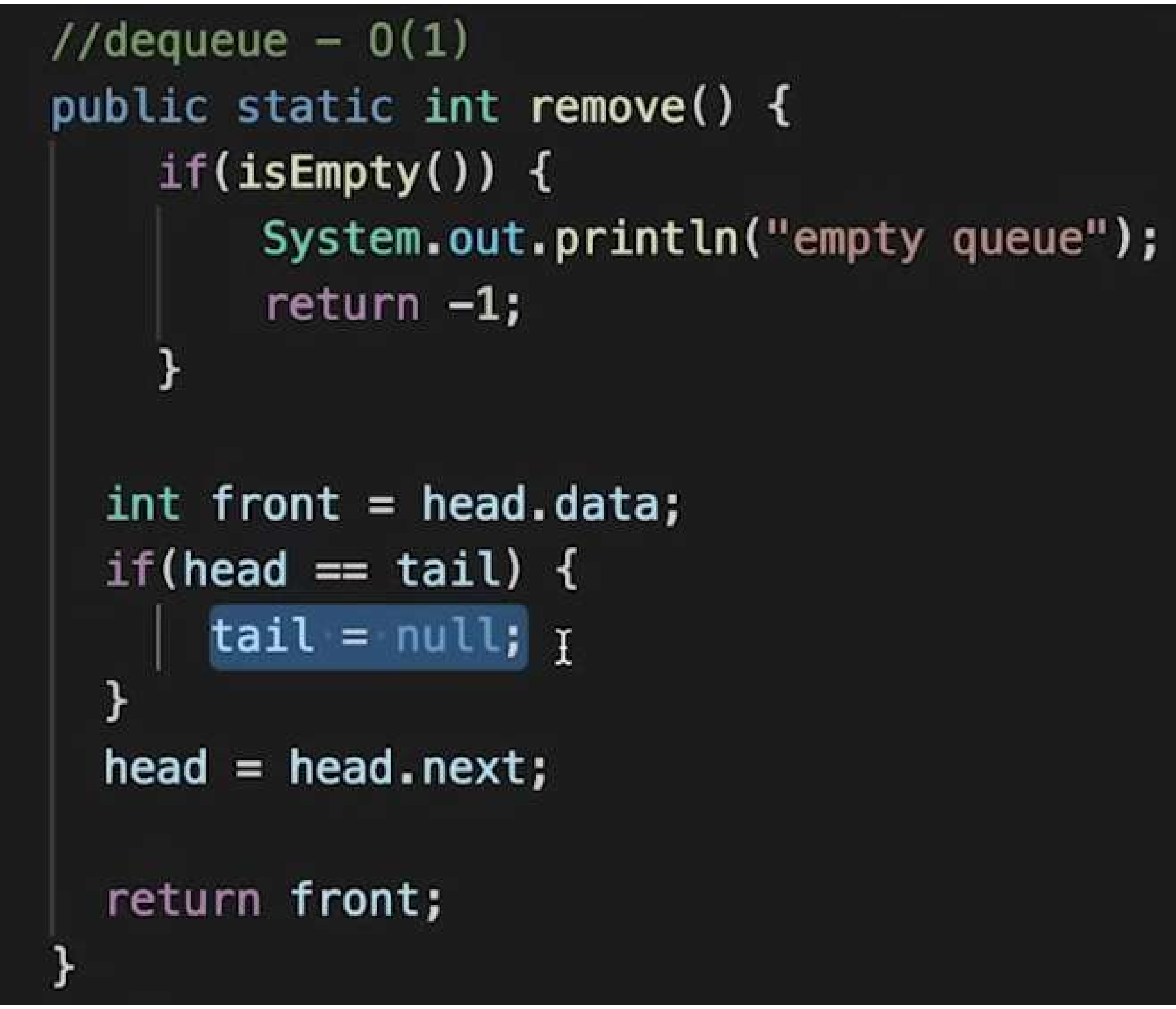
Queue using Linked List



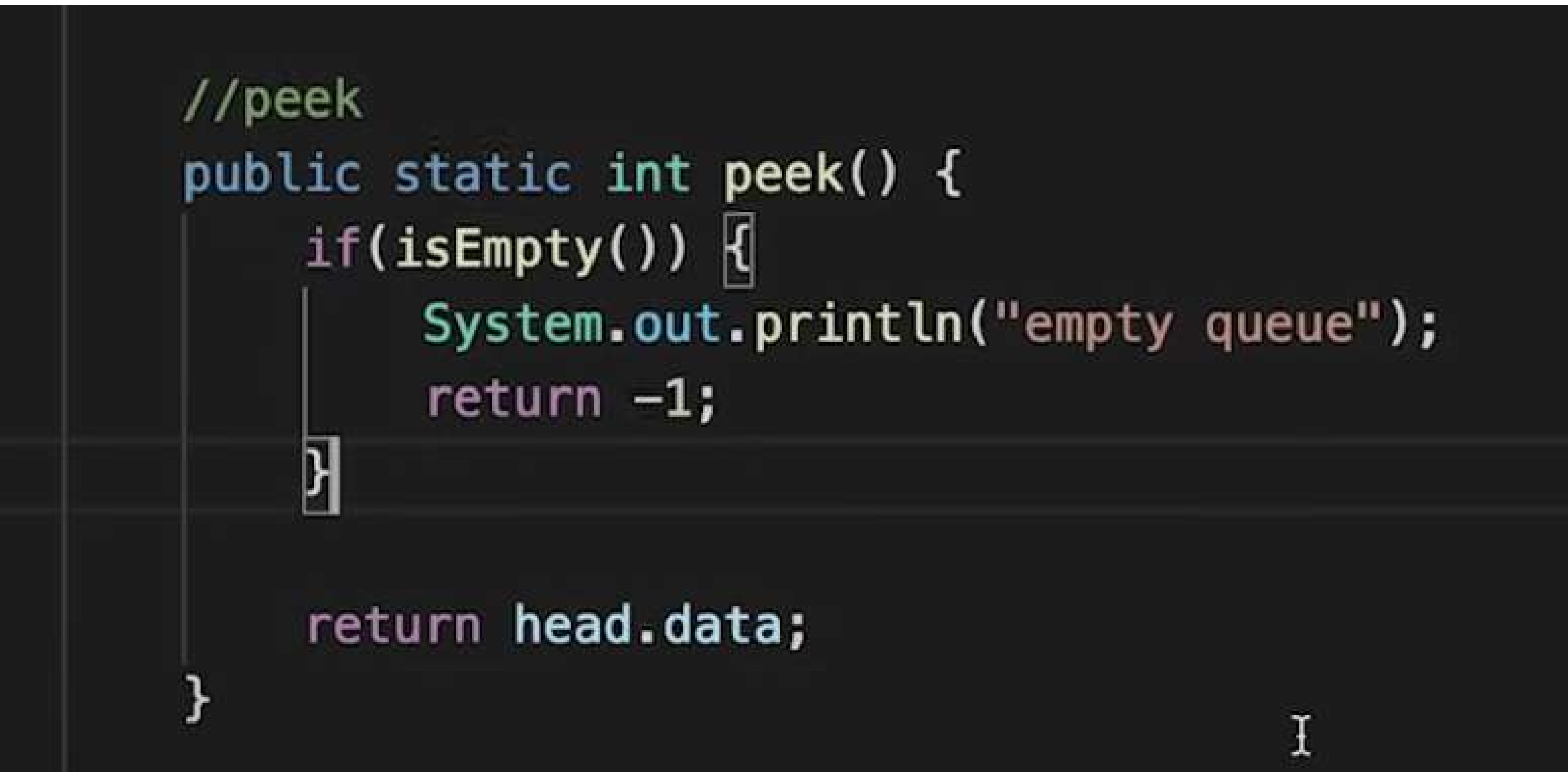
```
public class QueueY {
    class Node {
        int data;
        Node next;

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



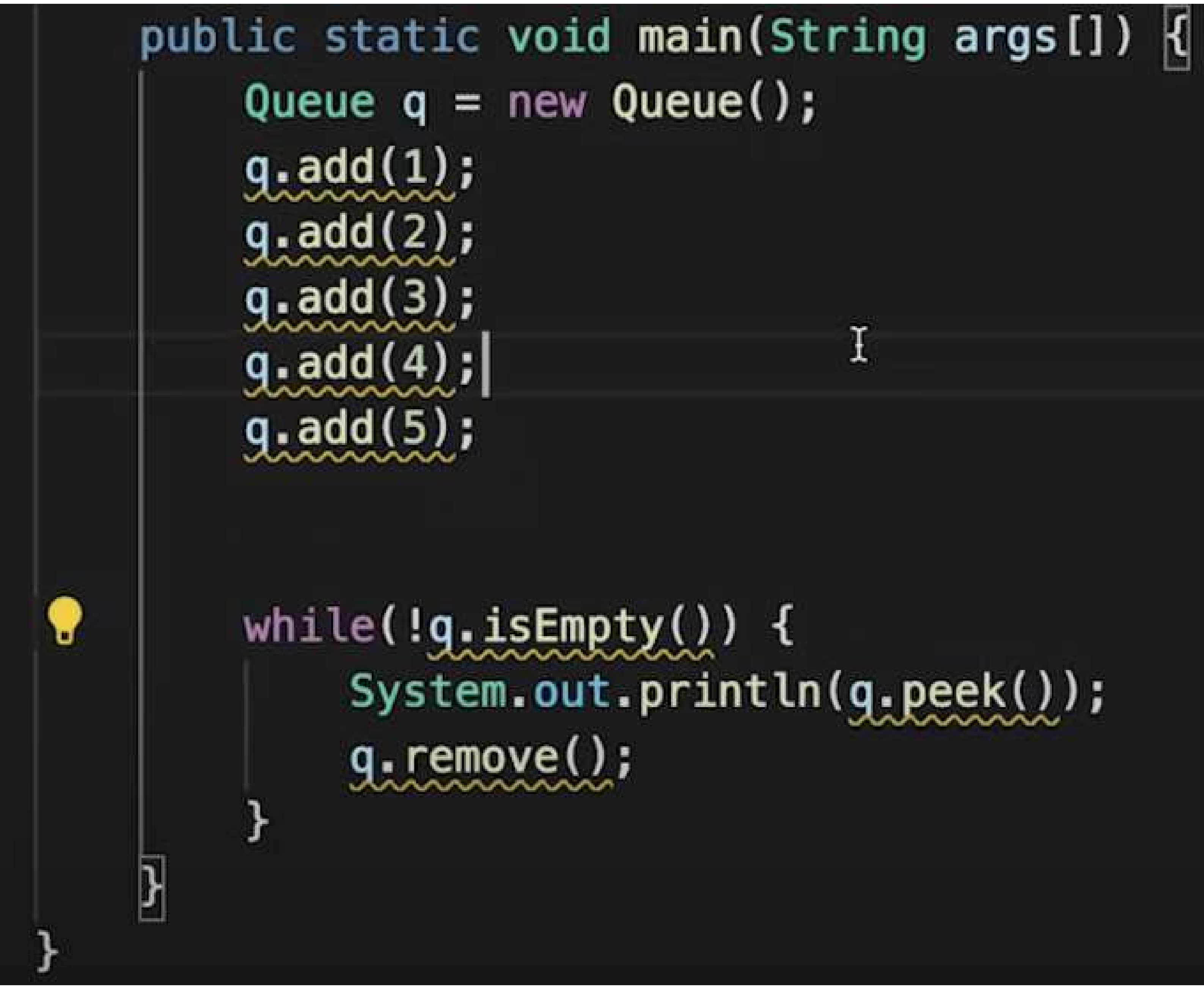
```java
static class Queue {
    static Node head = null;
    static Node tail = null;

    public static boolean isEmpty() {
        return head == null & tail == null;
    }

    //enqueue
    public static void add(int data) {
        Node newNode = new Node(data);
        if(tail == null) {
            tail = head = newNode;
            return;
        }

        tail.next = newNode;
        tail = newNode;
    }
```



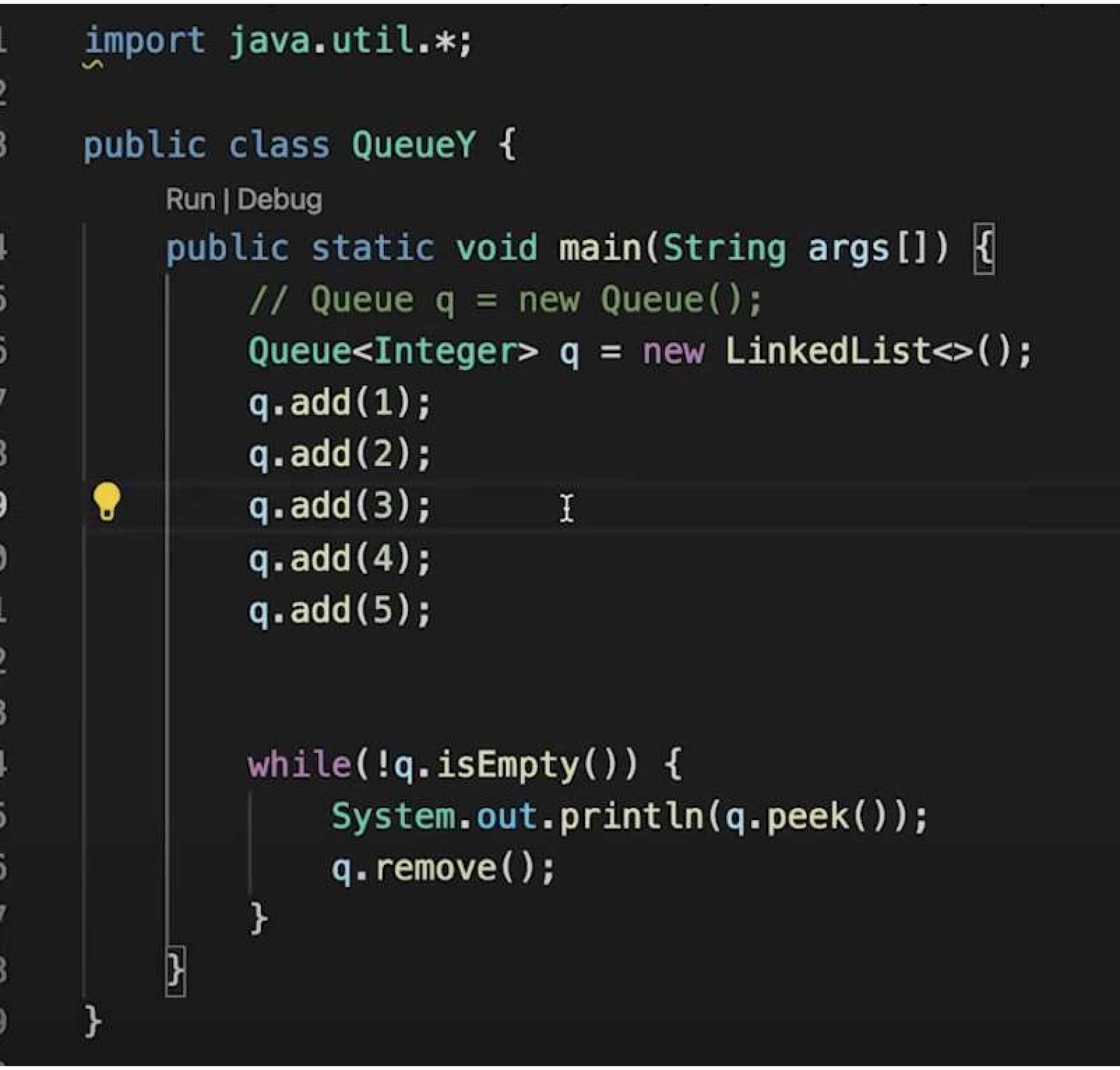
```java
//dequeue - O(1)
public static int remove() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }

  int front = head.data;
  if(head == tail) {
      tail = null;
  }
  head = head.next;

  return front;
}
```



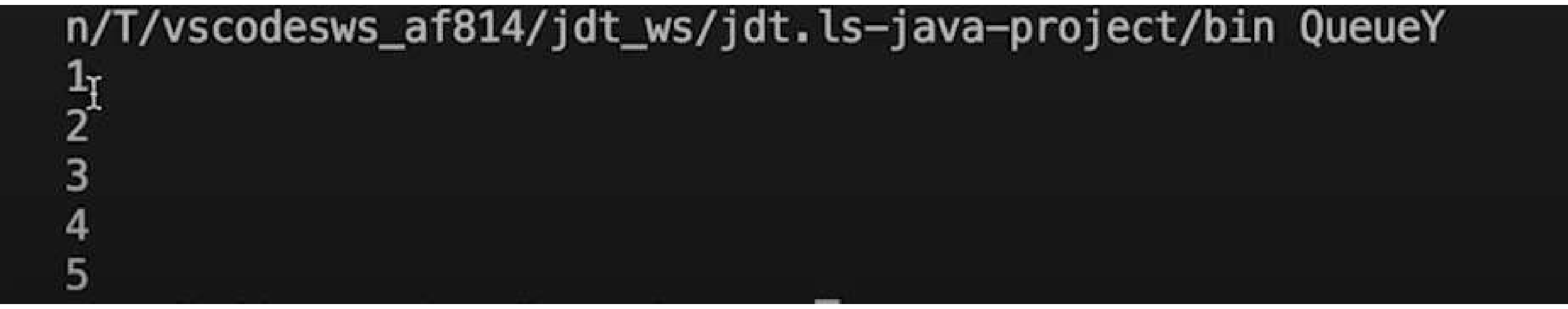
```
//peek
public static int peek() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }

    return head.data;
}
```



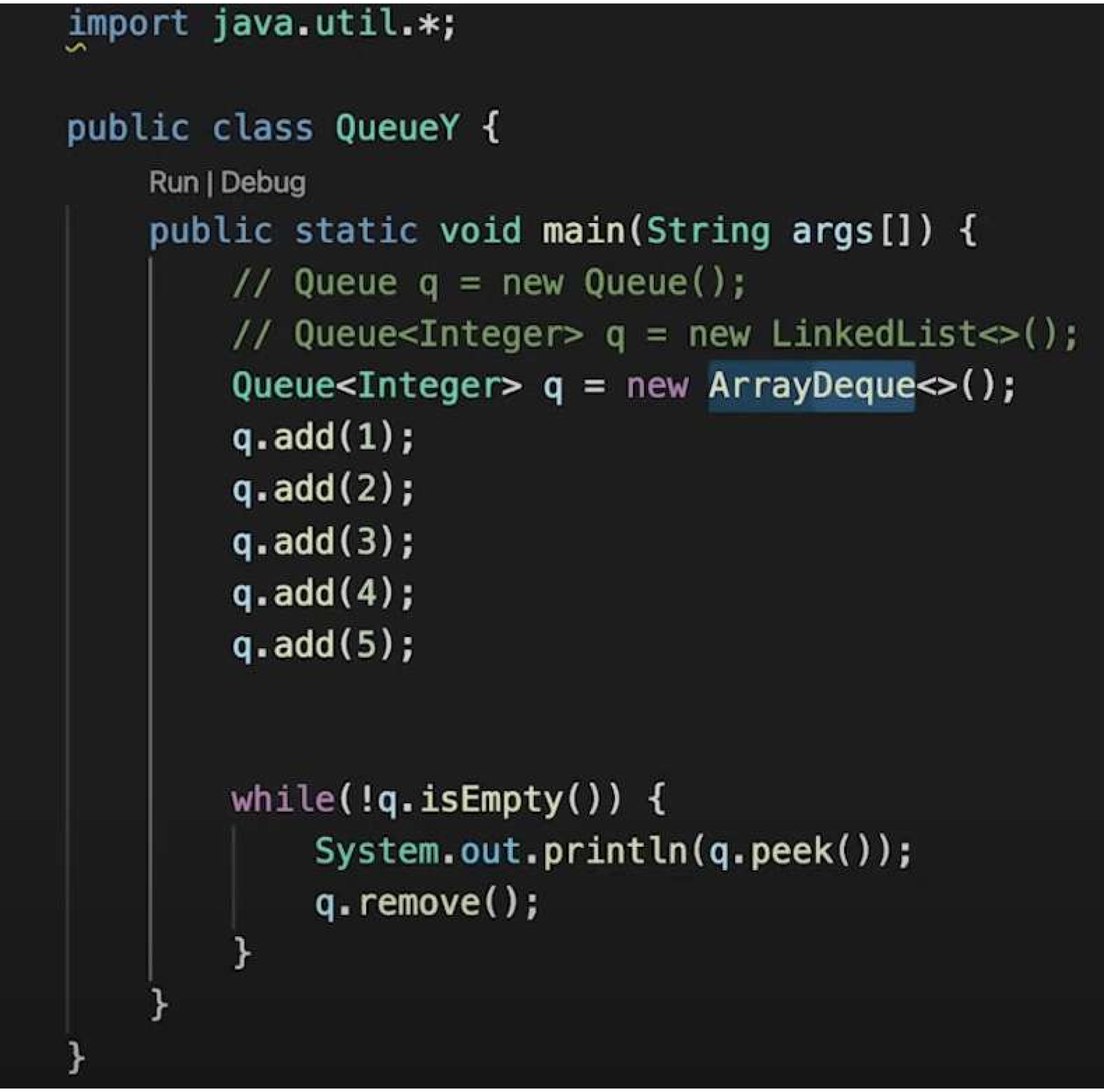
```
    public static void main(String args[]) {
        Queue q = new Queue();
        q.add(1);
        q.add(2);
        q.add(3);
        q.add(4);
        q.add(5);


        while(!q.isEmpty()) {
            System.out.println(q.peek());
            q.remove();
        }
    }
}
```


Run | Debug

```java
import java.util.*;

public class QueueY {
    public static void main(String args[]) {
        // Queue q = new Queue();
        Queue<Integer> q = new LinkedList<>();
        q.add(1);
        q.add(2);
        q.add(3);
        q.add(4);
        q.add(5);


        while(!q.isEmpty()) {
            System.out.println(q.peek());
            q.remove();
        }
    }
}
```



```
n/T/vscodesws_af814/jdt_ws/jdt.ls-java-project/bin QueueY
1
2
3
4
5
```



```java
import java.util.*;

public class QueueY {
    Run | Debug
    public static void main(String args[]) {
        // Queue q = new Queue();
        // Queue<Integer> q = new LinkedList<>();
        Queue<Integer> q = new ArrayDeque<>();
        q.add(1);
        q.add(2);
        q.add(3);
        q.add(4);
        q.add(5);


        while(!q.isEmpty()) {
            System.out.println(q.peek());
            q.remove();
        }
    }
}
```

1

2

3

4

5

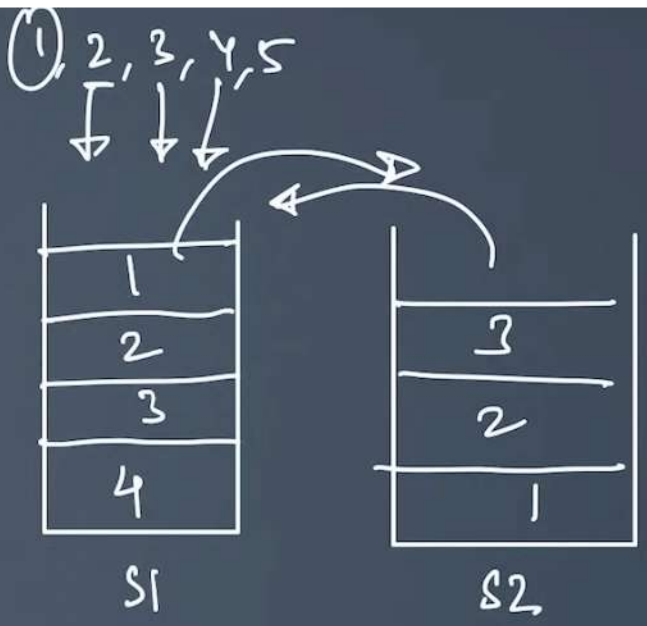
Question

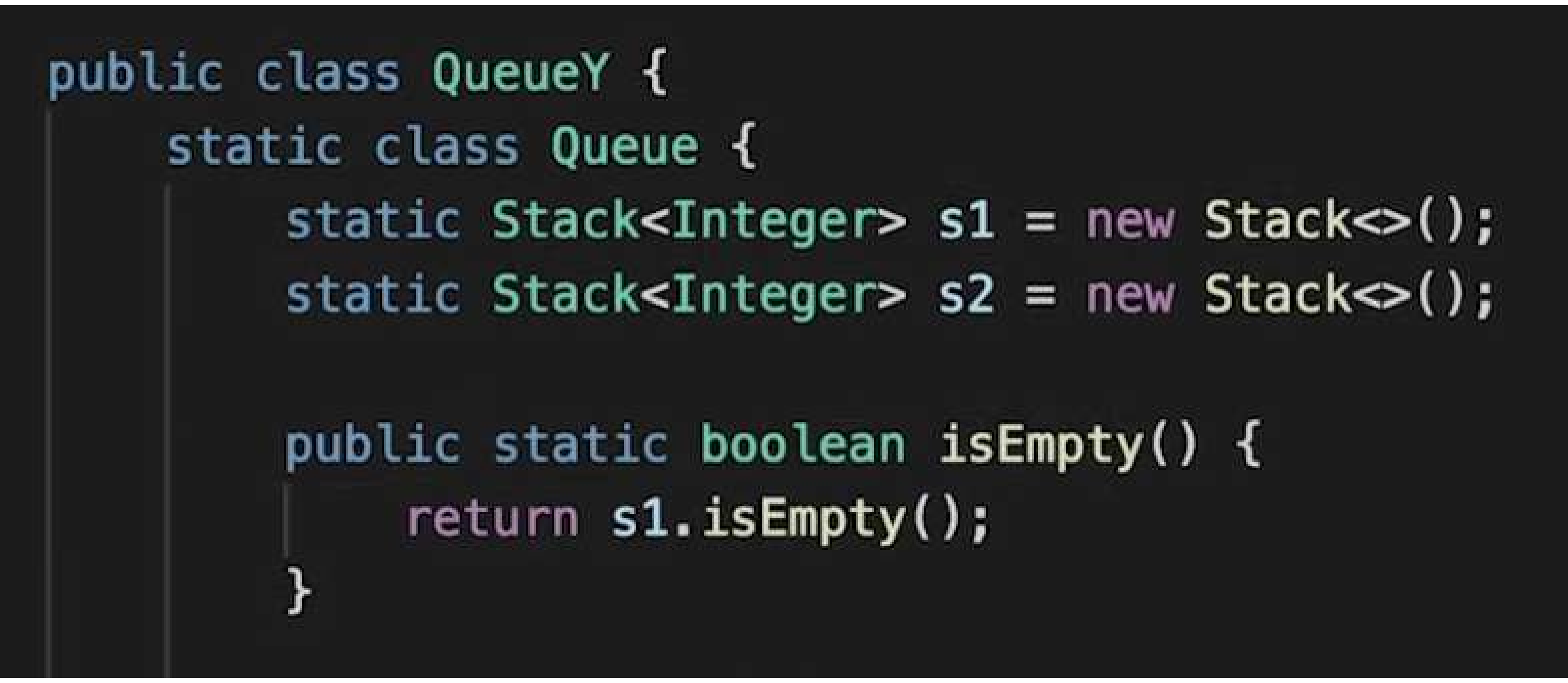
$$\begin{tikzcd} & \text{Queue using 2 Stacks} \arrow[dl] \arrow[dr] & \\ \text{Push} & & \text{Pop} \\ O(n) & & O(n) \end{tikzcd}$$

Question

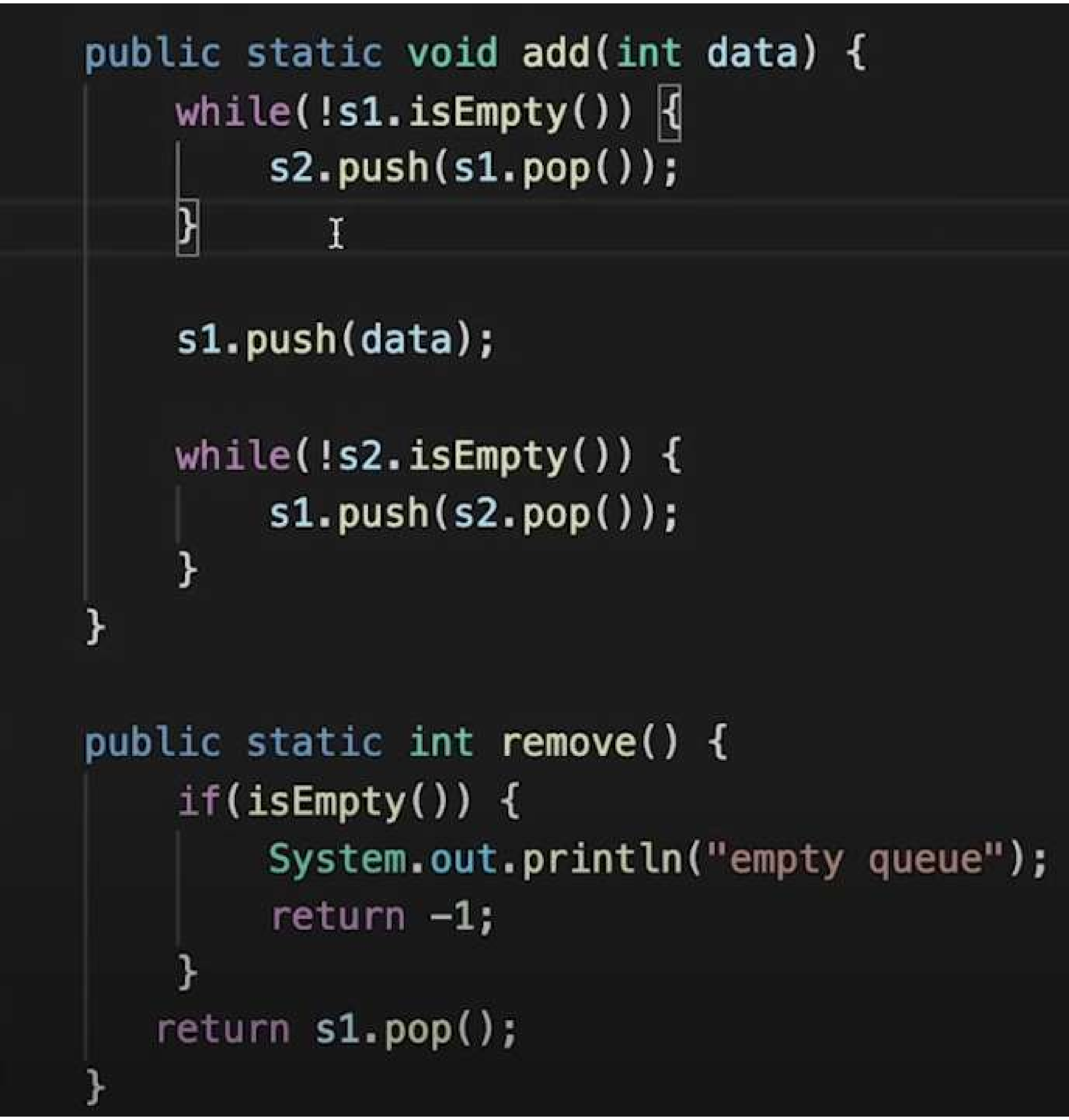
Queue using 2 Stacks

$$\begin{tikzcd}
 & \text{Queue using 2 Stacks} \arrow[dl] \arrow[dr] & \\
\text{Push } O(n) & & \text{Pop } O(n)
\end{tikzcd}$$

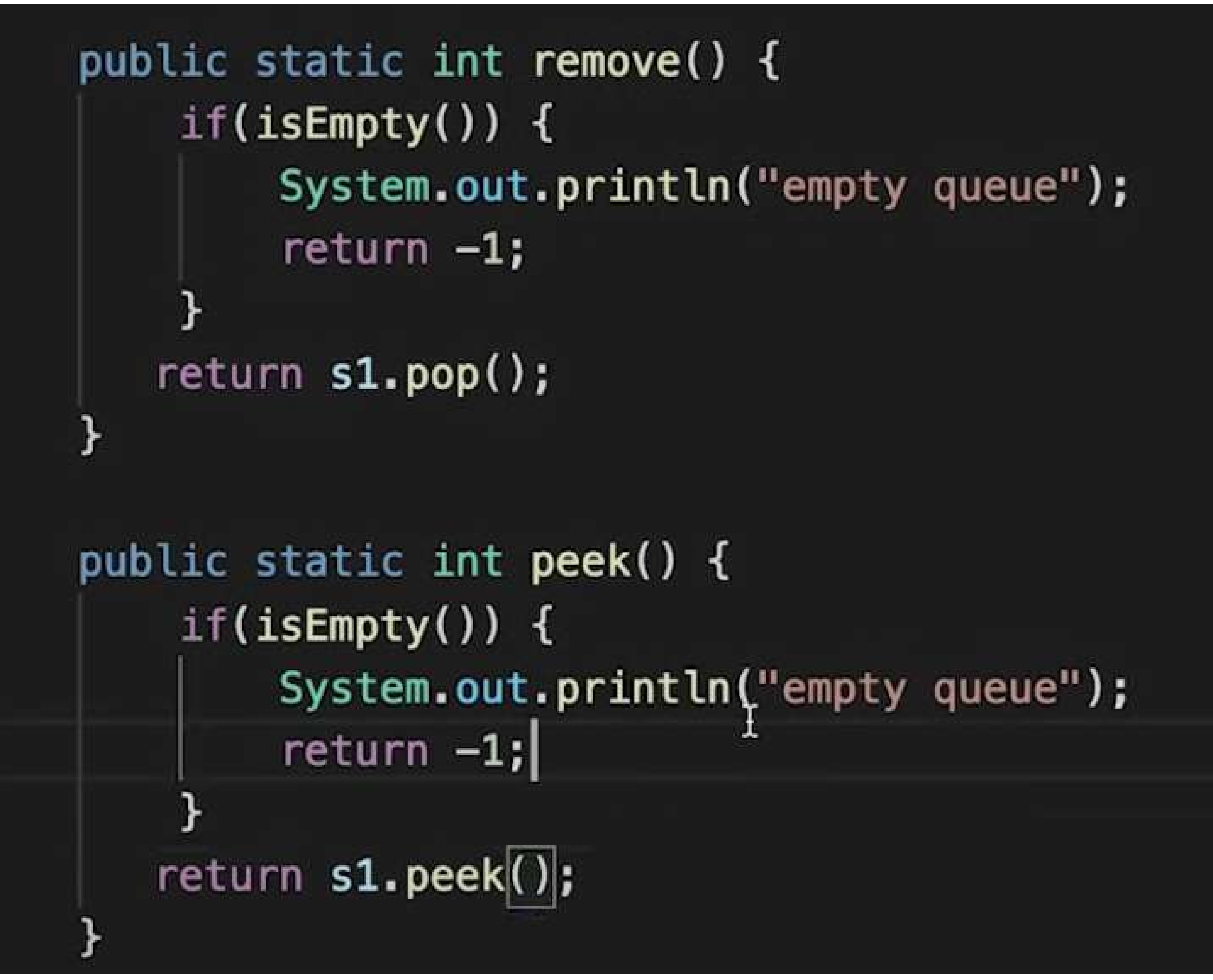




```java
public class QueueY {
    static class Queue {
        static Stack<Integer> s1 = new Stack<>();
        static Stack<Integer> s2 = new Stack<>();

        public static boolean isEmpty() {
            return s1.isEmpty();
        }
```



```java
public static void add(int data) {
    while(!s1.isEmpty()) {
        s2.push(s1.pop());
    }

    s1.push(data);

    while(!s2.isEmpty()) {
        s1.push(s2.pop());
    }
}

public static int remove() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }
   return s1.pop();
}
```



```java
public static int remove() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }
   return s1.pop();
}

public static int peek() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }
   return s1.peek();
}
```