**Lab 4**

**Name: M.TAHA Cms: 023-23-0377**

**Task 1:**

**Code:**

public class Stack { private int arr[]; private int top; private int capacity;

Stack(int size) { arr = new int[size]; capacity = size; top = -1;

}

public void push(int x) { if (isFull()) {

System.out.println("Stack Overflow"); return;

}

arr[++top] = x;

System.out.println("Inserting " + x);

}

public int pop() { if (isEmpty()) {

System.out.println("Stack Underflow"); return -1;

}

return arr[top--];

}

public int top() { if (!isEmpty()) { return arr[top];

}

System.out.println("Stack is empty"); return -1;

}

public int size() { return top + 1;

}

public Boolean isEmpty() { return top == -1;

}

public Boolean isFull() {

return top == capacity - 1;

}

public int peek() { return top();

}

public static void main(String[] args) { Stack stack = new Stack(3);

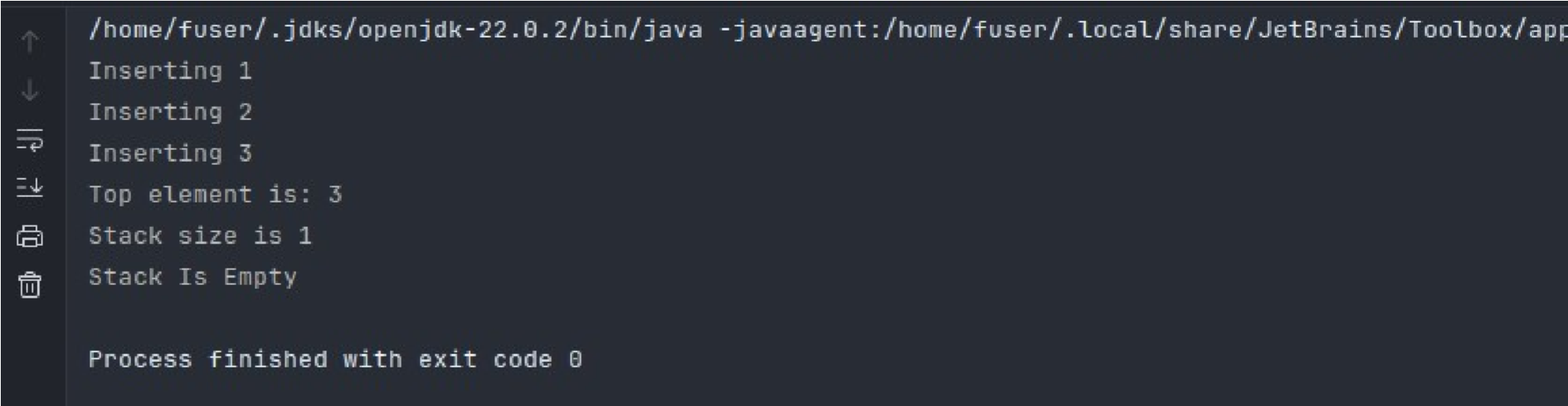
stack.push(1); stack.push(2); stack.pop(); stack.pop(); stack.push(3);

System.out.println("Top element is: " + stack.peek()); System.out.println("Stack size is " + stack.size()); stack.pop();

if (stack.isEmpty())

System.out.println("Stack Is Empty"); else

System.out.println("Stack Is Not Empty"); }}

**Output**

**Task 2: Stack using Linked list: Understand provided code and implement all required methods in Stack.**

**Code:**

class Stak { private node top;

public Stak() { this.top = null;

}

public void push(int x) { node newNode = new node(x);

newNode.next = top;

top = newNode;

}

public boolean isEmpty() { return top == null;

}

public int top() { if (isEmpty()) {

throw new RuntimeException("Stack is empty");

}

return top.data;

}

public void pop() {

if (isEmpty()) {

throw new RuntimeException("Stack underflow");

}

top = top.next;

}

public int peek() {

if (isEmpty()) {

throw new RuntimeException("Stack is empty");

}

return top.data;

}} public class StackImpl {

public static void main(String[] args) {

Stak stak = new Stak();

stak.push(1); stak.push(2); stak.push(3);

System.out.println("Top element is " + stak.peek()); stak.pop(); stak.pop(); stak.pop();

if (stak.isEmpty()) {

System.out.print("Stack is empty");

} else {

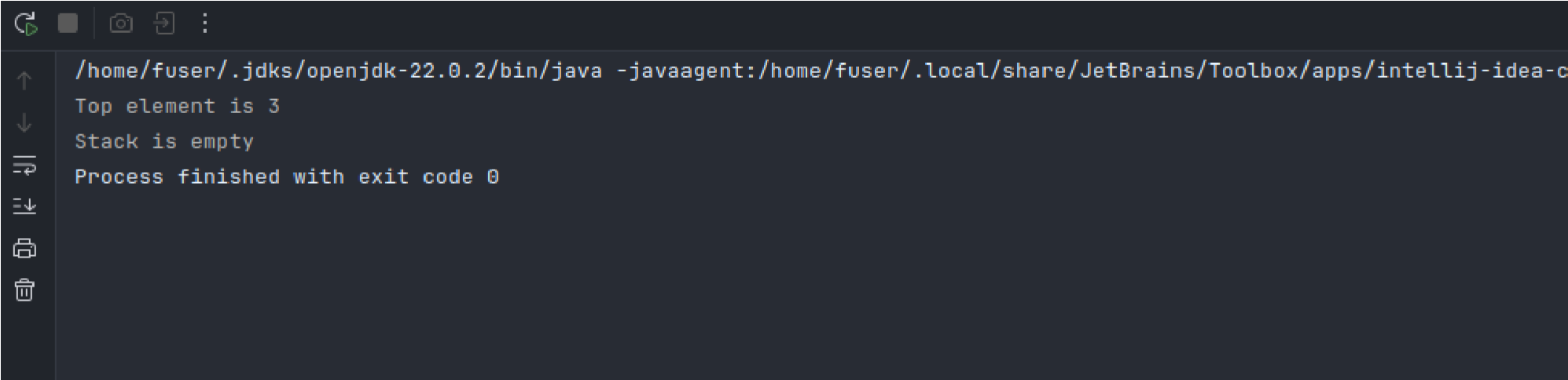
System.out.print("Stack is not empty");

}

}

}

**Output**



**Task 3:Queue using array: Understand provided code and implement all required methods in Queue.**

**Code:**

public class Queue { private int arr[]; private int front; private int rear; private int capacity; private int count;

Queue(int size) { arr = new int[size]; capacity = size; front = 0;

rear = 0; count = 0;

}

public void enqueue(int item) { if (isFull()) {

System.out.println("Queue is Full"); return;

} arr[rear] = item; rear = (rear + 1) % capacity; count++;

}

public void dequeue() { if (isEmpty()) {

System.out.println("Queue is Empty");

return;

}

front = (front + 1) % capacity;

count--;

} public int peek() { if (isEmpty()) {

System.out.println("Queue is Empty"); return -1; }

return arr[front];

} public int size() { return count;

}

public boolean isEmpty() {

return count == 0;

}

public boolean isFull() { return count == capacity;

}}

class Main { public static void main(String[] args) { Queue q = new Queue(5);

q.enqueue(1);

q.enqueue(2);

q.enqueue(3);

System.out.println("Front element is: " + q.peek()); q.dequeue();

System.out.println("Front element is: " + q.peek());

System.out.println("Queue size is " + q.size());

q.dequeue();

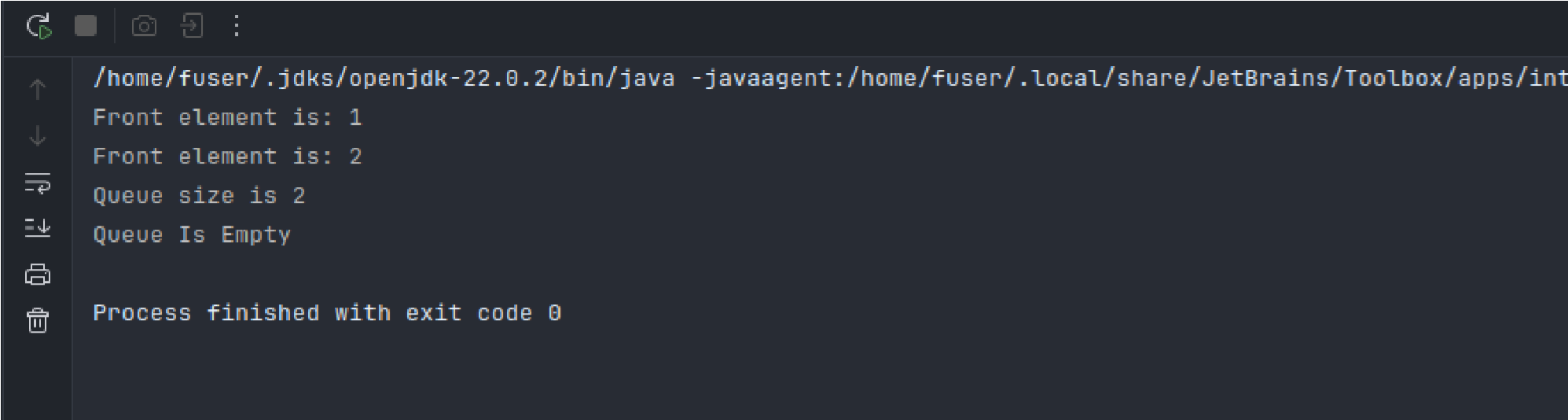
q.dequeue();

if (q.isEmpty()) {

System.out.println("Queue Is Empty");

} else {

System.out.println("Queue Is Not Empty"); }}}

**OutPut:**

**Task 4: Queue using Linked list: Understand provided code and implement all required methods in Queue.**

**Code:**

class node { int data; node next;

node(int data) { this.data = data; this.next = null;

}}

public class Queu { private node rear = null, front = null;

public void enqueue(int item) {

node newNode = new node(item); if (rear == null) {

front = rear = newNode;

} else {

rear.next = newNode;

rear = newNode;

}

}

public int dequeue() { if (isEmpty()) {

System.out.println("Queue is Empty"); return -1;

} int item = front.data; front = front.next; if (front == null) { rear = null;

} return item;

} public int peek() { if (isEmpty()) {

System.out.println("Queue is Empty"); return -1;

}

return front.data;

}

public boolean isEmpty() { return front == null;

}} class main {

public static void main(String[] args) { Queu q = new Queu();

q.enqueue(1);

q.enqueue(2);

q.enqueue(3);

q.enqueue(4);

System.out.printf("Front element is %d\n", q.peek()); q.dequeue();

q.dequeue();

q.dequeue();

q.dequeue();

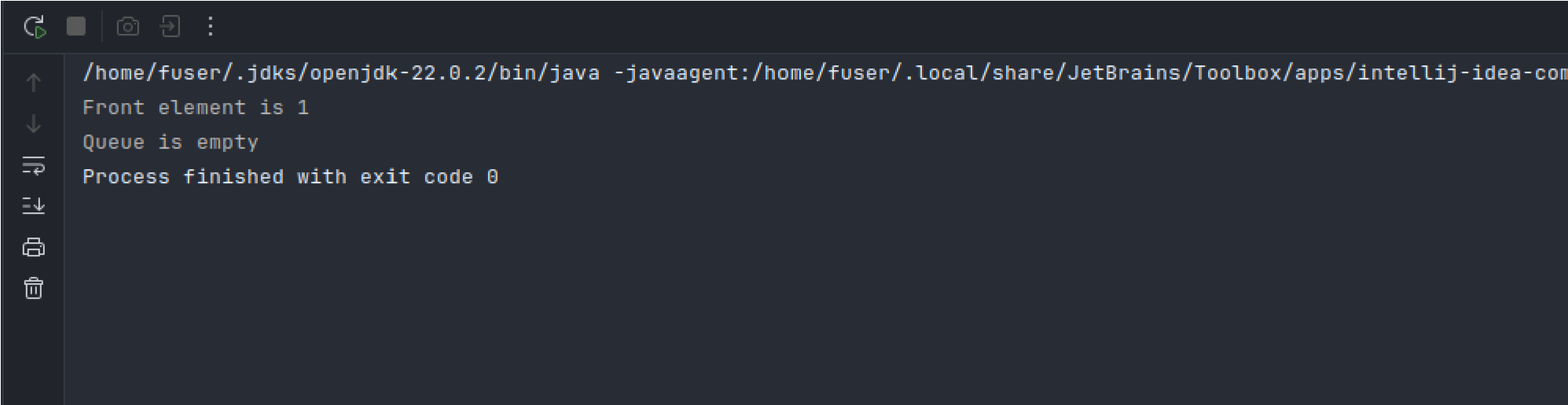
if (q.isEmpty()) {

System.out.print("Queue is empty");

} else {

System.out.print("Queue is not empty");

}}}

**Output:**

**Task 5: Queue using two Stacks: Understand provided code and implement all required methods in Queue Class.**

**Code:**

import java.util.Stack;

class Queueduostack { private Stack<Integer> s1; private Stack<Integer> s2;

Queueduostack() { s1 = new Stack<Integer>(); s2 = new Stack<Integer>();

}

public void enqueue(int data) {

s1.push(data);

}

public int dequeue() { if (s2.isEmpty()) { while (!s1.isEmpty()) {

s2.push(s1.pop());

}

}

if (s2.isEmpty()) { return -1;

}

return s2.pop();

}

public static void main(String[] args) {

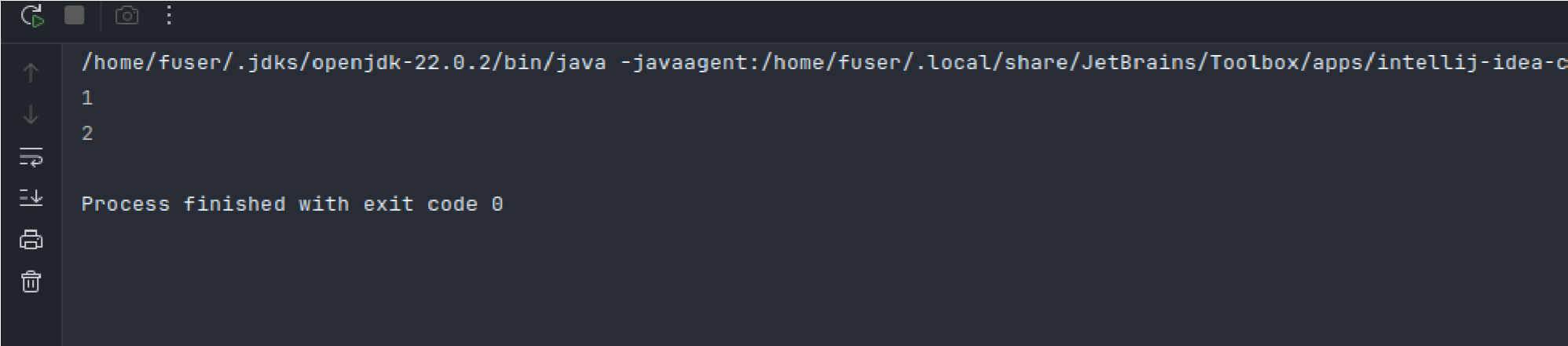
int[] keys = {1, 2, 3, 4, 5};

Queueduostack q = new Queueduostack(); for (int key : keys) { q.enqueue(key);

}

System.out.println(q.dequeue());

System.out.println(q.dequeue()); }}

**Output:**

**Task 6: Think about the inverse of task 05 (Stack using queue) and implement all the required methods.**

**Code:**

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

class StackUsingQueue { private Queue<Integer> q1; private Queue<Integer> q2;

StackUsingQueue() { q1 = new LinkedList<>(); q2 = new LinkedList<>();

}

public void push(int data) { q2.add(data);

while (!q1.isEmpty()) {

q2.add(q1.remove());

}

Queue<Integer> temp = q1;

q1 = q2; q2 = temp;

}

public int pop() { if (q1.isEmpty()) { return -1;

}

return q1.remove();

}

public int top() { if (q1.isEmpty()) { return -1;

}

return q1.peek();

}

public boolean isEmpty() {

return q1.isEmpty();

}

public static void main(String[] args) {

StackUsingQueue stack = new StackUsingQueue(); stack.push(1); stack.push(2); stack.push(3);

System.out.println(stack.pop());

System.out.println(stack.top());

System.out.println(stack.pop());

System.out.println(stack.isEmpty());

System.out.println(stack.pop());

System.out.println(stack.isEmpty());

}}

**Output:**

