**ICS1312 – JAVA PROGRAMMING LABORATORY**

**DATE : 2.8.2025**

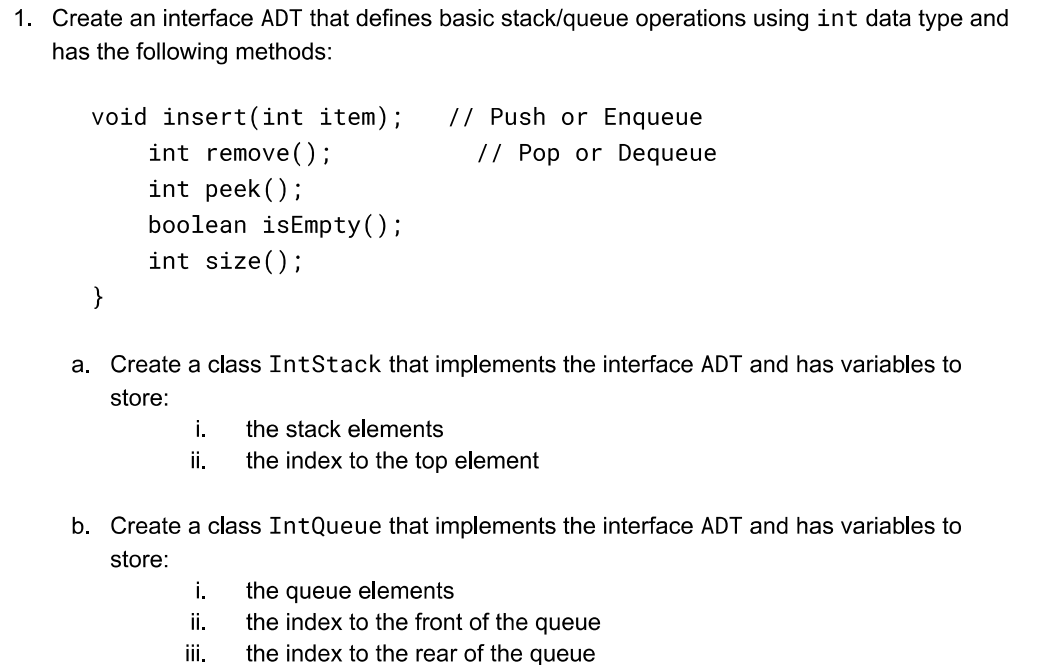
**ASSIGNMENT : 3**

**TITLE : JAVA INTERFACE FOR ADT**

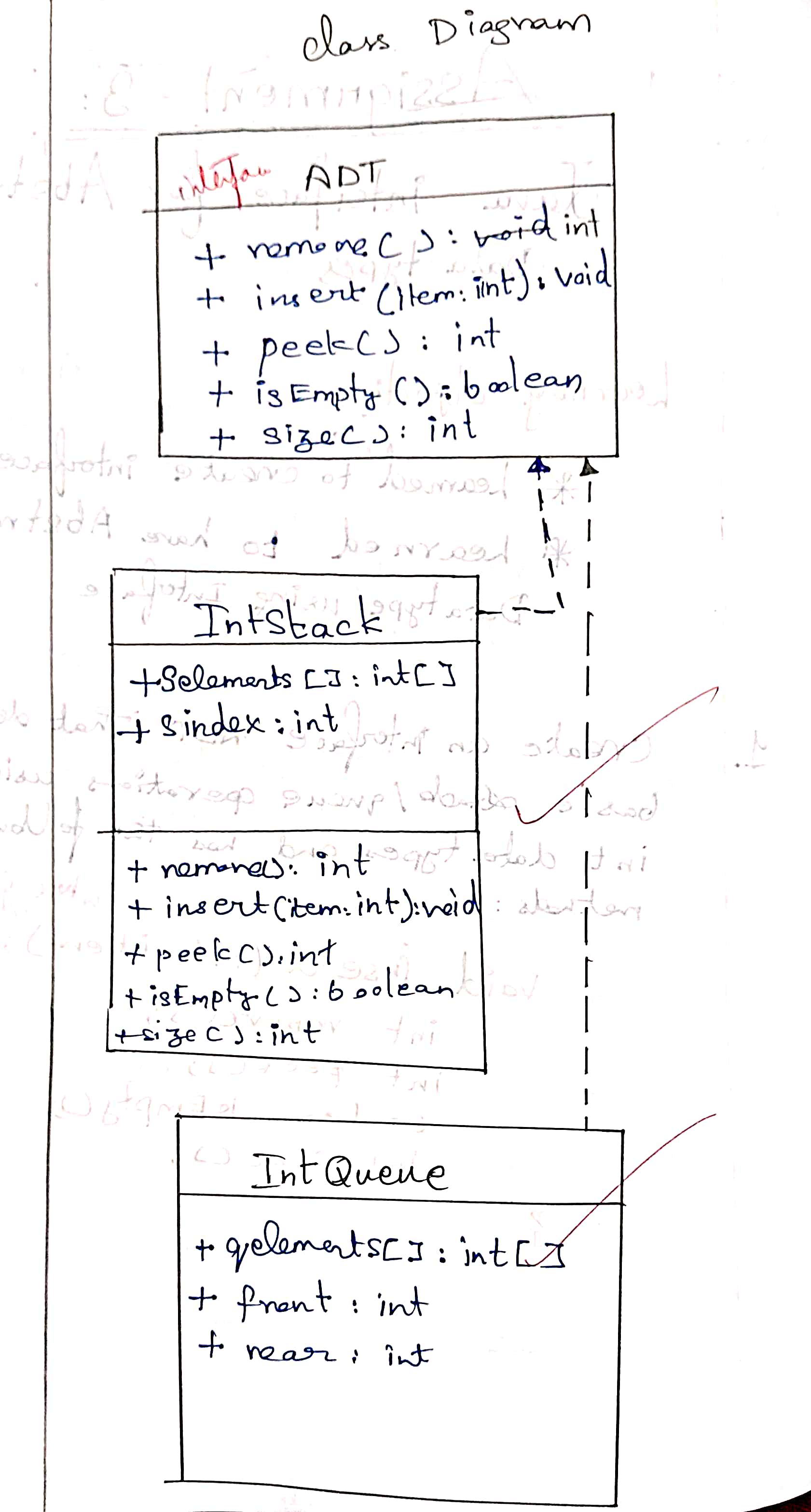
**ROLLL NO : 3122247001017**

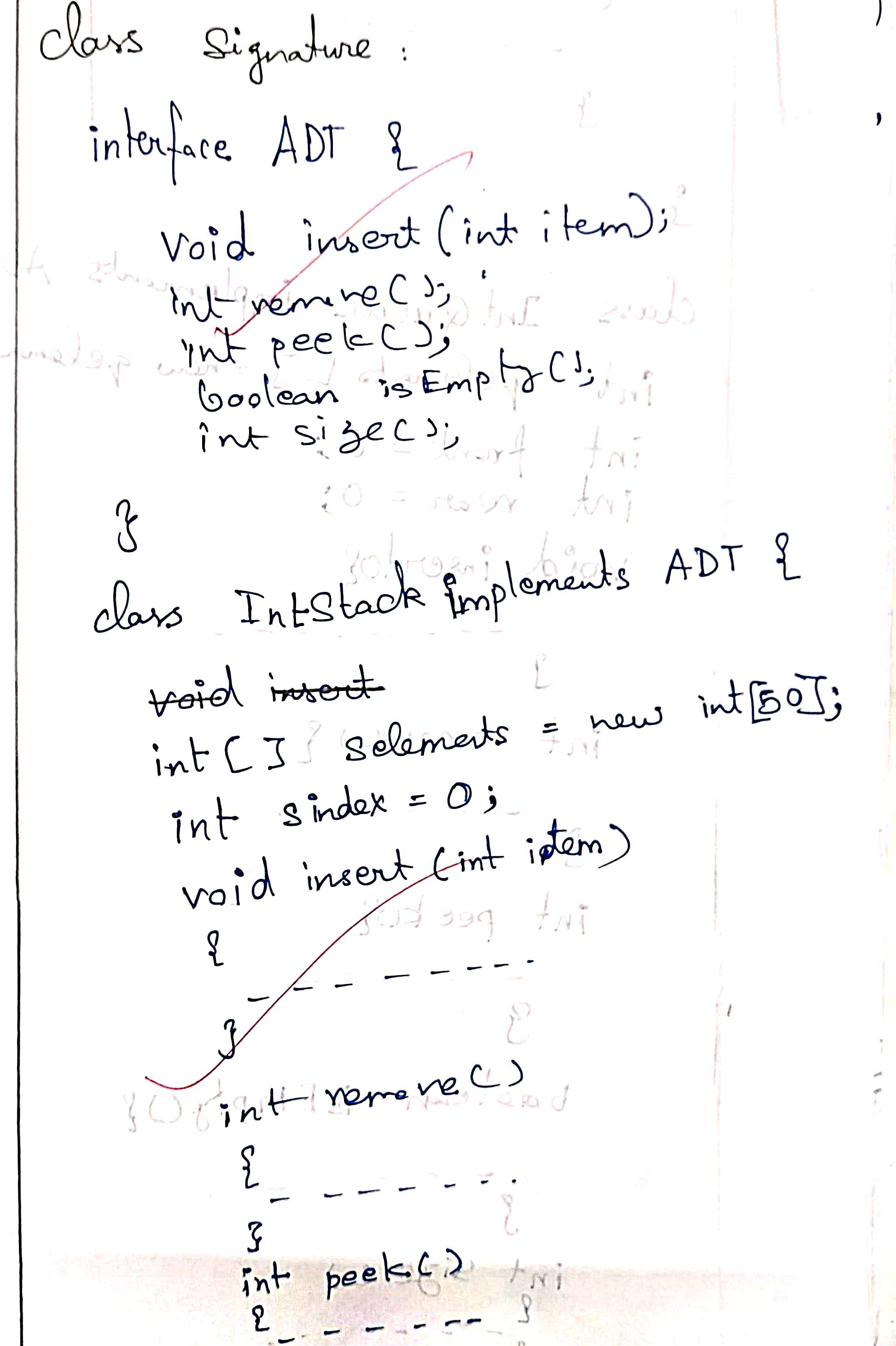
**LEARNING OBJECTIVE:**

* TO IMPLEMENT INTERFACE FOR THE GIVEN USE CASE
* TO CREATE STACK ADT USING INTERFACE IN JAVA
* TO CREATE SET ADT USING INTERFACE IN JAVA



CLASS DIAGRAM





A close-up of a paper

AI-generated content may be incorrect.

A paper with writing on it

AI-generated content may be incorrect.

CODE :

    interface ADT {

        void insert(int item);

        int remove();

        int peek();

        boolean isEmpty();

        int size();

    }

    class IntStack implements ADT {

        private int[] stack;

        private int top;

        public IntStack(int capacity) {

            stack = new int[capacity];

            top = -1;

        }

        public void insert(int item) {

            if (top < stack.length - 1) {

                stack[++top] = item;

            } else {

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

            }

        }

        public int remove() {

            if (!isEmpty()) {

                return stack[top--];

            }

            System.out.println("Stack Underflow");

            return -1;

        }

        public int peek() {

            if (!isEmpty()) {

                return stack[top];

            }

            return -1;

        }

        public boolean isEmpty() {

            return top == -1;

        }

        public int size() {

            return top + 1;

        }

    }

    class IntQueue implements ADT {

        private int[] queue;

        private int front, rear, count;

        public IntQueue(int capacity) {

            queue = new int[capacity];

            front = 0;

            rear = -1;

            count = 0;

        }

        public void insert(int item) {

            if (count < queue.length) {

                rear = (rear + 1) % queue.length;

                queue[rear] = item;

                count++;

            } else {

                System.out.println("Queue Overflow");

            }

        }

        public int remove() {

            if (!isEmpty()) {

                int item = queue[front];

                front = (front + 1) % queue.length;

                count--;

                return item;

            }

            System.out.println("Queue Underflow");

            return -1;

        }

        public int peek() {

            if (!isEmpty()) {

                return queue[front];

            }

            return -1;

        }

        public boolean isEmpty() {

            return count == 0;

        }

        public int size() {

            return count;

        }

    }

    public class Main {

    public static void main(String[] args) {

        System.out.println("Stack Example:");

        ADT stack = new IntStack(5);

        stack.insert(10);

        stack.insert(20);

        System.out.println("Top of stack: " + stack.peek());

        System.out.println("Removed from stack: " + stack.remove());

        System.out.println("Stack size: " + stack.size());

        System.out.println("\nQueue Example:");

        ADT queue = new IntQueue(5);

        queue.insert(100);

        queue.insert(200);

        System.out.println("Front of queue: " + queue.peek());

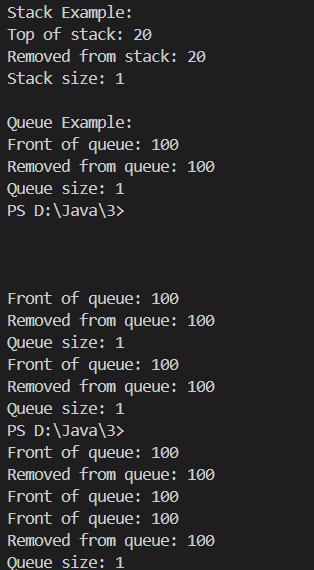
        System.out.println("Removed from queue: " + queue.remove());

        System.out.println("Queue size: " + queue.size());

    }

}

**OUTPUT :**



A white text with black text

AI-generated content may be incorrect.

A close-up of a paper

AI-generated content may be incorrect.

A close-up of a paper

AI-generated content may be incorrect.

A close-up of a paper

AI-generated content may be incorrect.

A close-up of a paper

AI-generated content may be incorrect.CODE :

    interface SetADT {

        boolean add(int item);

        boolean remove(int item);

        boolean contains(int item);

        SetADT union(SetADT otherSet);

    }

   class IntSet implements SetADT {

        private int[] elements;

        private int size;

        public IntSet() {

            elements = new int[100];

            size = 0;

        }

        public boolean add(int item) {

            if (contains(item)) return false;

            elements[size++] = item;

            return true;

        }

        public boolean remove(int item) {

            for (int i = 0; i < size; i++) {

                if (elements[i] == item) {

                    for (int j = i; j < size - 1; j++) {

                        elements[j] = elements[j + 1];

                    }

                    size--;

                    return true;

                }

            }

            return false;

        }

        public boolean contains(int item) {

            for (int i = 0; i < size; i++) {

                if (elements[i] == item) return true;

            }

            return false;

        }

        public SetADT union(SetADT otherSet) {

            IntSet result = new IntSet();

            for (int i = 0; i < this.size; i++) {

                result.add(this.elements[i]);

            }

            if (otherSet instanceof IntSet) {

                IntSet other = (IntSet) otherSet;

                for (int i = 0; i < other.size; i++) {

                    result.add(other.elements[i]);

                }

            }

            return result;

        }

        public void display() {

            System.out.print("Set: ");

            for (int i = 0; i < size; i++) {

                System.out.print(elements[i] + " ");

            }

            System.out.println();

        }

    }

    public class Main2 {

    public static void main(String[] args) {

        IntSet set1 = new IntSet();

        set1.add(5);

        set1.add(3);

        set1.add(9);

        set1.add(5);

        System.out.println("Set 1:");

        set1.display();

        IntSet set2 = new IntSet();

        set2.add(3);

        set2.add(7);

        set2.add(1);

        System.out.println("Set 2:");

        set2.display();

        SetADT unionSet = set1.union(set2);

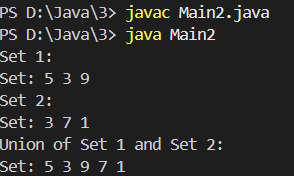
        System.out.println("Union of Set 1 and Set 2:");

        ((IntSet) unionSet).display();

    }

}

**OUTPUT:**

****

**LEARNING OUTCOMES :**

* **LEARNED TO IMPLEMENT INTERFACE**
* **LEARNED TO IMPLAMENT STACK IN JAVA**
* **LEARNED TO IMPLEMENT SET IN JAVA**