1.create stack using linked list

class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

class Stack {

Node top;

public Stack() {

this.top = null;

}

public void push(int data) {

Node newNode = new Node(data);

if (this.top == null) {

this.top = newNode;

} else {

newNode.next = this.top;

this.top = newNode;

}

}

public int pop() {

if (this.top == null) {

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

return -1;

}

int data = this.top.data;

this.top = this.top.next;

return data;

}

public int peek() {

if (this.top == null) {

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

return -1;

}

return this.top.data;

}

public boolean isEmpty() {

return this.top == null;

}

public void printStack() {

Node temp = this.top;

while (temp != null) {

System.out.print(temp.data + " ");

temp = temp.next;

}

System.out.println();

}

}

public class Main {

public static void main(String[] args) {

Stack stack = new Stack();

stack.push(1);

stack.push(2);

stack.push(3);

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

stack.printStack();

System.out.println("Popped element: " + stack.pop());

System.out.println("Popped element: " + stack.pop());

System.out.println("Stack elements after popping:");

stack.printStack();

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

System.out.println("Is stack empty " + stack.isEmpty());

}

}

2.create queue using linked list

class Node {

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

class Queue {

Node front;

Node rear;

public Queue() {

this.front = null;

this.rear = null;

}

public void enqueue(int data) {

Node newNode = new Node(data);

if (rear == null) {

front = newNode;

rear = newNode;

} else {

rear.next = newNode;

rear = newNode;

}

}

public int dequeue() {

if (front == null) {

throw new RuntimeException("Queue is empty");

}

int data = front.data;

front = front.next;

if (front == null) {

rear = null;

}

return data;

}

public int peek() {

if (front == null) {

throw new RuntimeException("Queue is empty");

}

return front.data;

}

public boolean isEmpty() {

return front == null;

}

}

public class Main {

public static void main(String[] args) {

Queue queue = new Queue();

queue.enqueue(5);

queue.enqueue(2);

queue.enqueue(3);

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

System.out.println(queue.peek());

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

}

}

3.create hash map

import java.util.HashMap;

import java.util.Map;

public class hashmap{

public static void main(String[] args) {

HashMap<Integer, String> map = new HashMap<>();

map.put(1, "Apple");

map.put(2, "Banana");

map.put(3, "watermelon");

System.out.println("Initial HashMap: " + map);

String value1 = map.get(1);

String value2 = map.get(2);

System.out.println("Value for key 1: " + value1);

System.out.println("Value for key 2: " + value2);

boolean hasKey3 = map.containsKey(3);

boolean hasKey5 = map.containsKey(5);

System.out.println("HashMap contains key 3: " + hasKey3);

System.out.println("HashMap contains key 5: " + hasKey5);

boolean hasValueCherry = map.containsValue("Cherry");

boolean hasValueGrape = map.containsValue("Grape");

System.out.println("HashMap contains value 'Cherry': " + hasValueCherry);

System.out.println("HashMap contains value 'Grape': " + hasValueGrape);

map.remove(2);

System.out.println("HashMap after removing key 2: " + map);

map.put(3, "Citrus");

System.out.println("HashMap after updating key 3: " + map);

System.out.println("Iterating over HashMap:");

for (Map.Entry<Integer, String> entry : map.entrySet()) {

System.out.println("Key: " + entry.getKey() + ", Value: " + entry.getValue());

}

int size = map.size();

System.out.println("Size of HashMap: " + size);

map.clear();

System.out.println("HashMap after clearing: " + map);

System.out.println("Size after clearing: " + map.size());

}

}

4.create comparing using student

import java.util.ArrayList;

import java.util.Collections;

class Student implements Comparable<Student> {

int rollNo;

String name;

Student(int rollNo, String name) {

this.rollNo = rollNo;

this.name = name;

}

public int compareTo(Student other) {

if (this.rollNo < other.rollNo) {

return -1;

} else if (this.rollNo > other.rollNo) {

return 1;

} else {

return 0;

}

}

public String toString() {

return "Student{rollNo=" + rollNo + ", name='" + name + "'}";

}

}

public class Main {

public static void main(String[] args) {

ArrayList<Student> students = new ArrayList<>();

students.add(new Student(3, "teju"));

students.add(new Student(1, "ash"));

students.add(new Student(2, "sumi"));

Collections.sort(students);

for (Student student : students) {

System.out.println(student);

}

}

}