**Questions:**

1. Problem Statement

Design and implement a data structure for a Least Recently Used (LRU) cache. It should support the following operations: get and put.

get(key): Get the value (will always be positive) of the key if the key exists in the cache, otherwise return -1.

put(key, value): Set or insert the value if the key is not already present. When the cache reaches its capacity, it should invalidate the least recently used item before inserting a new item.

Constraints

The number of get and put operations will be in the range [1, 10^5].

The capacity of the cache is between 1 and 10^5.

import java.util.Map;

import java.util.HashMap;

public class LRUCache {

private final int capacity;

private final Node head;

private final Node tail;

private final Map<Integer, Node> cache;

public LRUCache(int capacity) {

this.capacity = capacity;

this.cache = new HashMap<>();

this.head = new Node(0, 0);

this.tail = new Node(0, 0);

head.next = tail;

tail.prev = head;

}

public int get(int key) {

Node node = cache.get(key);

if (node == null) return -1;

moveToHead(node);

return node.value;

}

public void put(int key, int value) {

Node node = cache.get(key);

if (node == null) {

node = new Node(key, value);

cache.put(key, node);

addToHead(node);

if (cache.size() > capacity) {

removeLast();

}

} else {

node.value = value;

moveToHead(node);

}

}

private void addToHead(Node node) {

node.prev = head;

node.next = head.next;

head.next.prev = node;

head.next = node;

}

private void removeLast() {

Node last = tail.prev;

last.prev.next = tail;

tail.prev = last.prev;

cache.remove(last.key);

}

private void moveToHead(Node node) {

removeNode(node);

addToHead(node);

}

private void removeNode(Node node) {

node.prev.next = node.next;

node.next.prev = node.prev;

}

private class Node {

int key;

int value;

Node prev;

Node next;

public Node(int key, int value) {

this.key = key;

this.value = value;

}

}

public static void main(String[] args) {

LRUCache cache = new LRUCache(2);

cache.put(1, 1);

cache.put(2, 2);

System.out.println(cache.get(1)); // returns 1

cache.put(3, 3);

System.out.println(cache.get(2)); // returns -1 (evicted)

cache.put(4, 4);

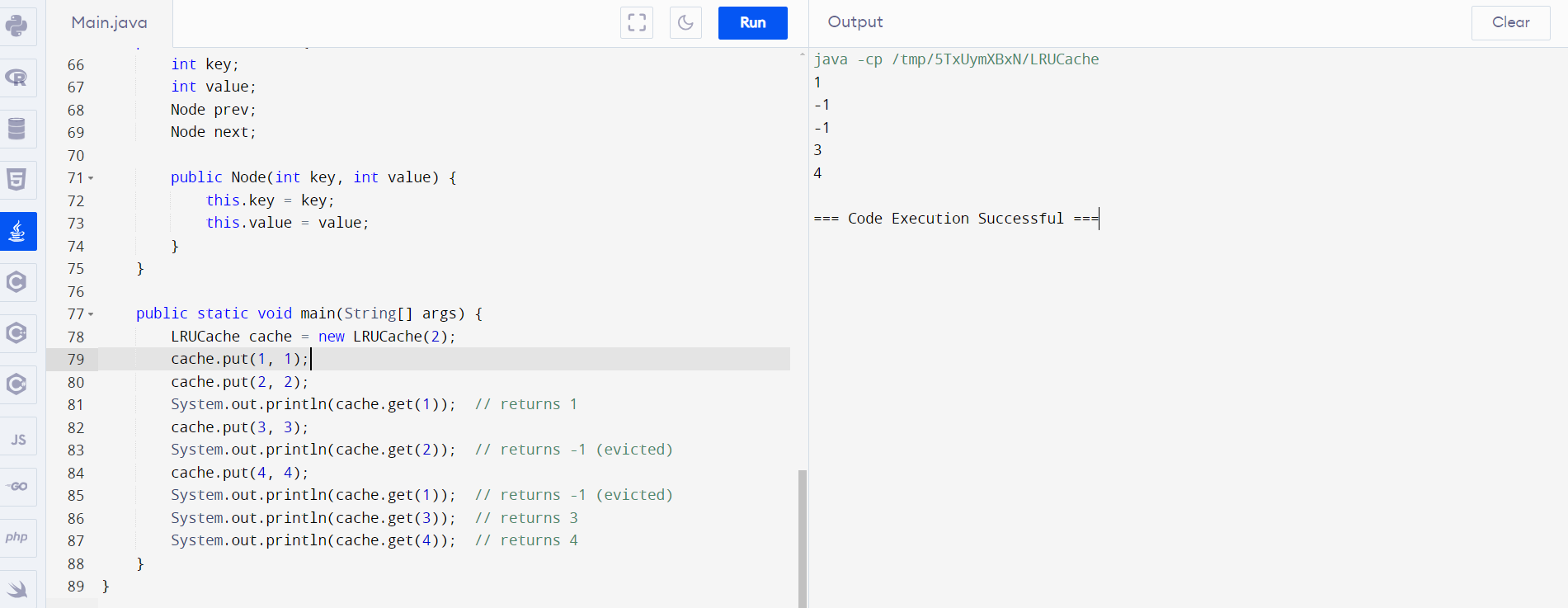
System.out.println(cache.get(1)); // returns -1 (evicted)

System.out.println(cache.get(3)); // returns 3

System.out.println(cache.get(4)); // returns 4

}

}



**2**.Write a Java program that demonstrates the ConcurrentModificationException. Explain why the exception is thrown and how to handle it properly.

import java.util.ArrayList;

import java.util.Iterator;

public class ConcurrentModificationExceptionExample {

public static void main(String[] args) {

ArrayList<String> list = new ArrayList<String>();

list.add("Apple");

list.add("Banana");

list.add("Cherry");

Iterator<String> iterator = list.iterator();

while (iterator.hasNext()) {

String fruit = iterator.next();

System.out.println(fruit);

list.remove(fruit); // This line throws ConcurrentModificationException

}

}

}

correct program:

import java.util.ArrayList;

import java.util.Iterator;

public class ConcurrentModificationExceptionExample {

public static void main(String[] args) {

ArrayList<String> list = new ArrayList<String>();

list.add("Apple");

list.add("Banana");

list.add("Cherry");

Iterator<String> iterator = list.iterator();

while (iterator.hasNext()) {

String fruit = iterator.next();

System.out.println(fruit);

iterator.remove(); // This line is safe and does not throw ConcurrentModificationException

}

}

}



3. Create a custom annotation @LogExecutionTime to log the execution time of annotated methods. Implement an annotation processor to handle this annotation.

class Example {

public void exampleMethod() {

long startTime = System.currentTimeMillis();

// method code here

long endTime = System.currentTimeMillis();

long executionTime = endTime - startTime;

System.out.println("Method took " + executionTime + " milliseconds to execute");

}

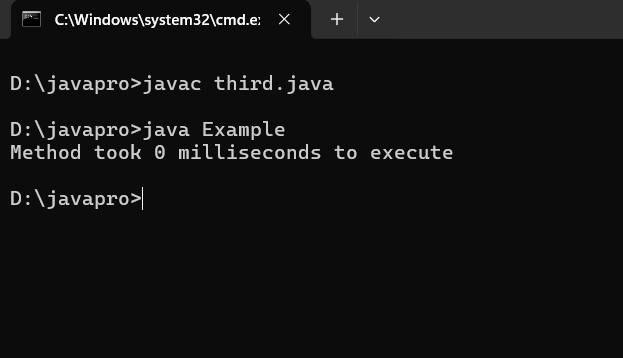
public static void main(String[] args) {

Example example = new Example();

example.exampleMethod();

}

}



5.Problem Statement

Implement a trie with insert, search, and startsWith methods.

insert(word): Inserts a word into the trie.

search(word): Returns if the word is in the trie.

startsWith(prefix): Returns if there is any word in the trie that starts with the given prefix.

Constraints

You may assume that all inputs are consist of lowercase letters a-z.

All inputs are guaranteed to be non-empty strings.

answer this program with logical manner with out put

import java.util.Map;

import java.util.HashMap;

class Trie {

private TrieNode root;

public Trie() {

root = new TrieNode();

}

public void insert(String word) {

TrieNode current = root;

for (int i = 0; i < word.length(); i++) {

char c = word.charAt(i);

if (!current.children.containsKey(c)) {

current.children.put(c, new TrieNode());

}

current = current.children.get(c);

}

current.isEndOfWord = true;

}

public boolean search(String word) {

TrieNode current = root;

for (int i = 0; i < word.length(); i++) {

char c = word.charAt(i);

if (!current.children.containsKey(c)) {

return false;

}

current = current.children.get(c);

}

return current.isEndOfWord;

}

public boolean startsWith(String prefix) {

TrieNode current = root;

for (int i = 0; i < prefix.length(); i++) {

char c = prefix.charAt(i);

if (!current.children.containsKey(c)) {

return false;

}

current = current.children.get(c);

}

return true;

}

private class TrieNode {

private boolean isEndOfWord;

private Map<Character, TrieNode> children;

public TrieNode() {

isEndOfWord = false;

children = new HashMap<>();

}

}

public static void main(String[] args) {

Trie trie = new Trie();

trie.insert("apple");

trie.insert("app");

trie.insert("banana");

System.out.println(trie.search("apple")); // true

System.out.println(trie.search("app")); // true

System.out.println(trie.search("banana")); // true

System.out.println(trie.search("ban")); // false

System.out.println(trie.startsWith("apple")); // true

System.out.println(trie.startsWith("app")); // true

System.out.println(trie.startsWith("ban")); // true

System.out.println(trie.startsWith("ba")); // true

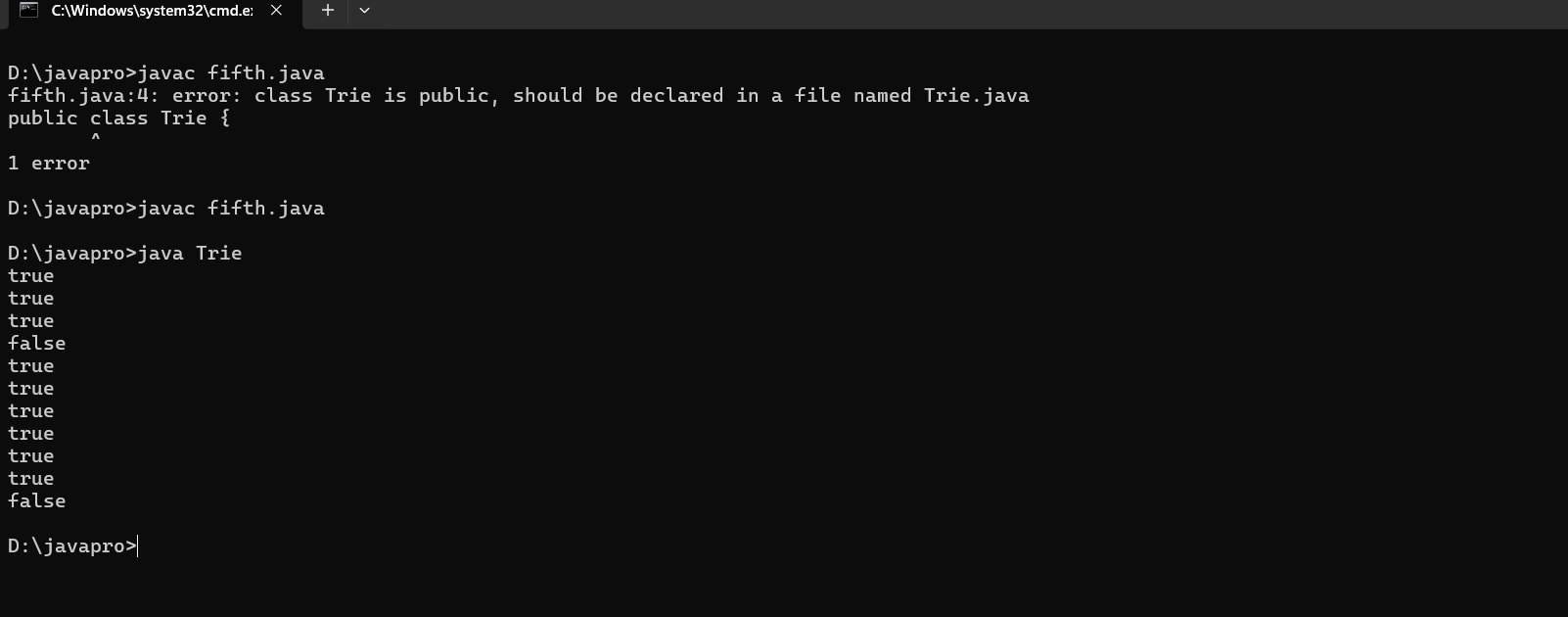
System.out.println(trie.startsWith("b")); // true

System.out.println(trie.startsWith("a")); // true

System.out.println(trie.startsWith("orange")); // false

}

}



**8.**Find the kth largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.

import java.util.Scanner;

class Sixth {

public static void main(String[] args) {

Scanner number = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int size = number.nextInt();

int[] unsorted = new int[size];

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

System.out.print("Enter element " + (i + 1) + ": ");

unsorted[i] = number.nextInt();

}

int temp;

for (int i = 0; i < unsorted.length; i++) {

for (int j = i + 1; j < unsorted.length; j++) {

if (unsorted[i] > unsorted[j]) {

temp = unsorted[i];

unsorted[i] = unsorted[j];

unsorted[j] = temp;

}

}

}

System.out.print("Sorted array: ");

for (int i = 0; i < unsorted.length; i++) {

System.out.println(unsorted[i] + " ");

}

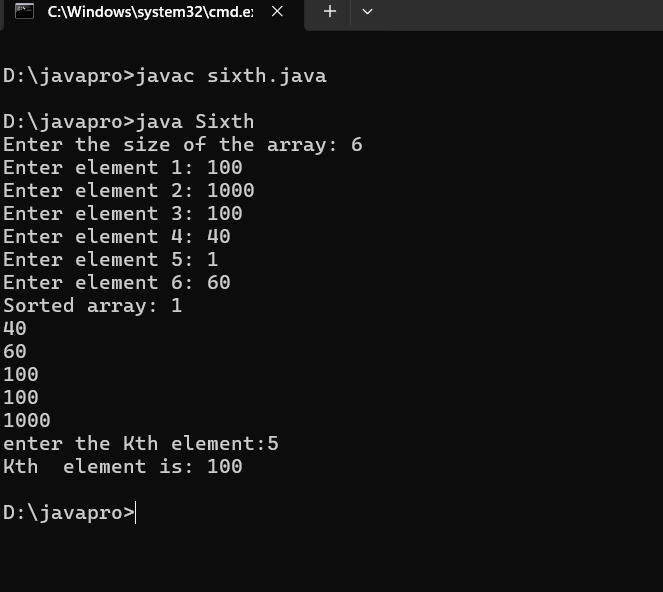
System.out.print("enter the Kth element:");

int k = number.nextInt();

System.out.println("Kth element is: " + unsorted[k - 1]);

}

}



10.Write a Java program that checks if a given string is a palindrome. A palindrome is a word, phrase, number, or other sequences of characters that reads the same forward and backward (ignoring spaces, punctuation, and capitalization).

import java.util.Scanner;

class PalindromeChecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a string: ");

String input = scanner.nextLine();

boolean isPalindrome = isPalindrome(input);

if (isPalindrome) {

System.out.println("The string is a palindrome.");

} else {

System.out.println("The string is not a palindrome.");

}

}

public static boolean isPalindrome(String str) {

str = str.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();

int left = 0;

int right = str.length() - 1;

while (left < right) {

if (str.charAt(left) != str.charAt(right)) {

return false;

}

left++;

right--;

}

return true;

}

}

