# **Name: Abdurrahman Qureshi**

# **Roll No: 242466**

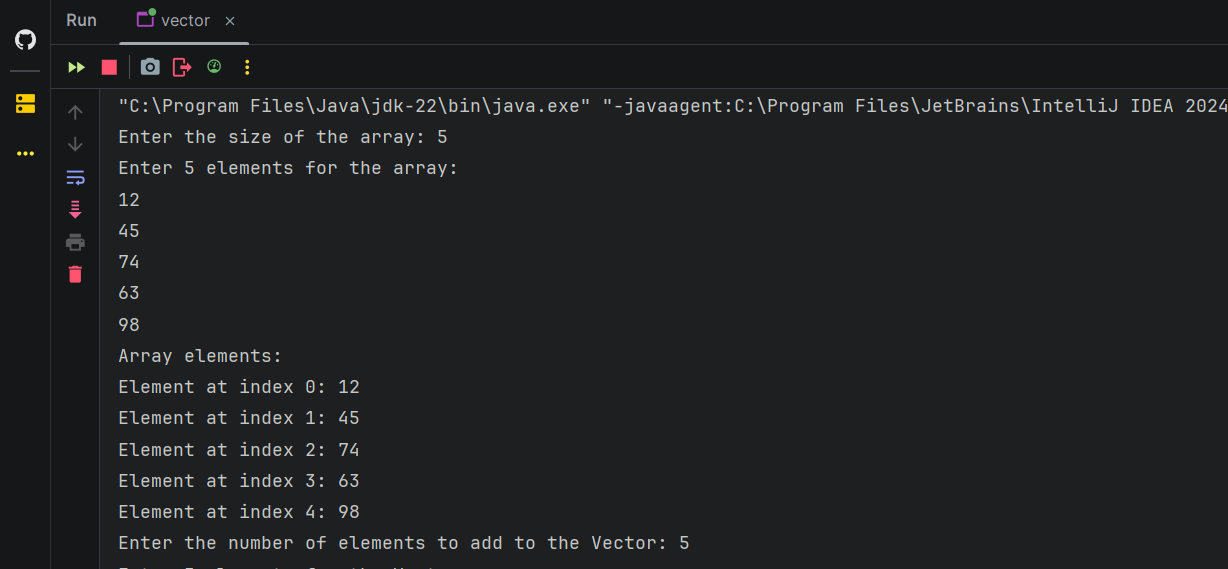
Practical No: 6

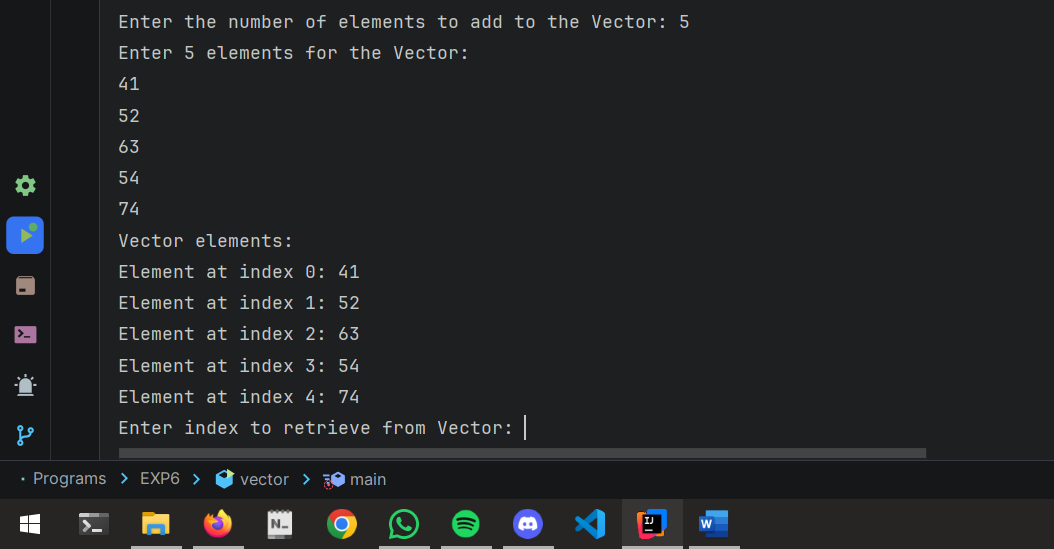
1) **Implementation of Simple Array and Vector java Application**

CODE:

package EXP6;  
  
import java.util.Scanner;  
import java.util.Vector;  
  
public class vector {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter the size of the array: ");  
 int size = scanner.nextInt();  
 int[] array = new int[size];  
  
 System.*out*.println("Enter " + size + " elements for the array:");  
 for (int i = 0; i < size; i++) {  
 array[i] = scanner.nextInt();  
 }  
  
 System.*out*.println("Array elements:");  
 for (int i = 0; i < array.length; i++) {  
 System.*out*.println("Element at index " + i + ": " + array[i]);  
 }  
  
 Vector<Integer> vector = new Vector<>();  
  
 System.*out*.print("Enter the number of elements to add to the Vector: ");  
 int vectorSize = scanner.nextInt();  
 System.*out*.println("Enter " + vectorSize + " elements for the Vector:");  
 for (int i = 0; i < vectorSize; i++) {  
 vector.add(scanner.nextInt());  
 }  
  
 System.*out*.println("Vector elements:");  
 for (int i = 0; i < vector.size(); i++) {  
 System.*out*.println("Element at index " + i + ": " + vector.get(i));  
 }  
  
 System.*out*.print("Enter index to retrieve from Vector: ");  
 int index = scanner.nextInt();  
 if (index >= 0 && index < vector.size()) {  
 System.*out*.println("Element at index " + index + ": " + vector.get(index));  
 } else {  
 System.*out*.println("Invalid index!");  
 }  
  
 scanner.close();  
 }  
}

OUTPUT:



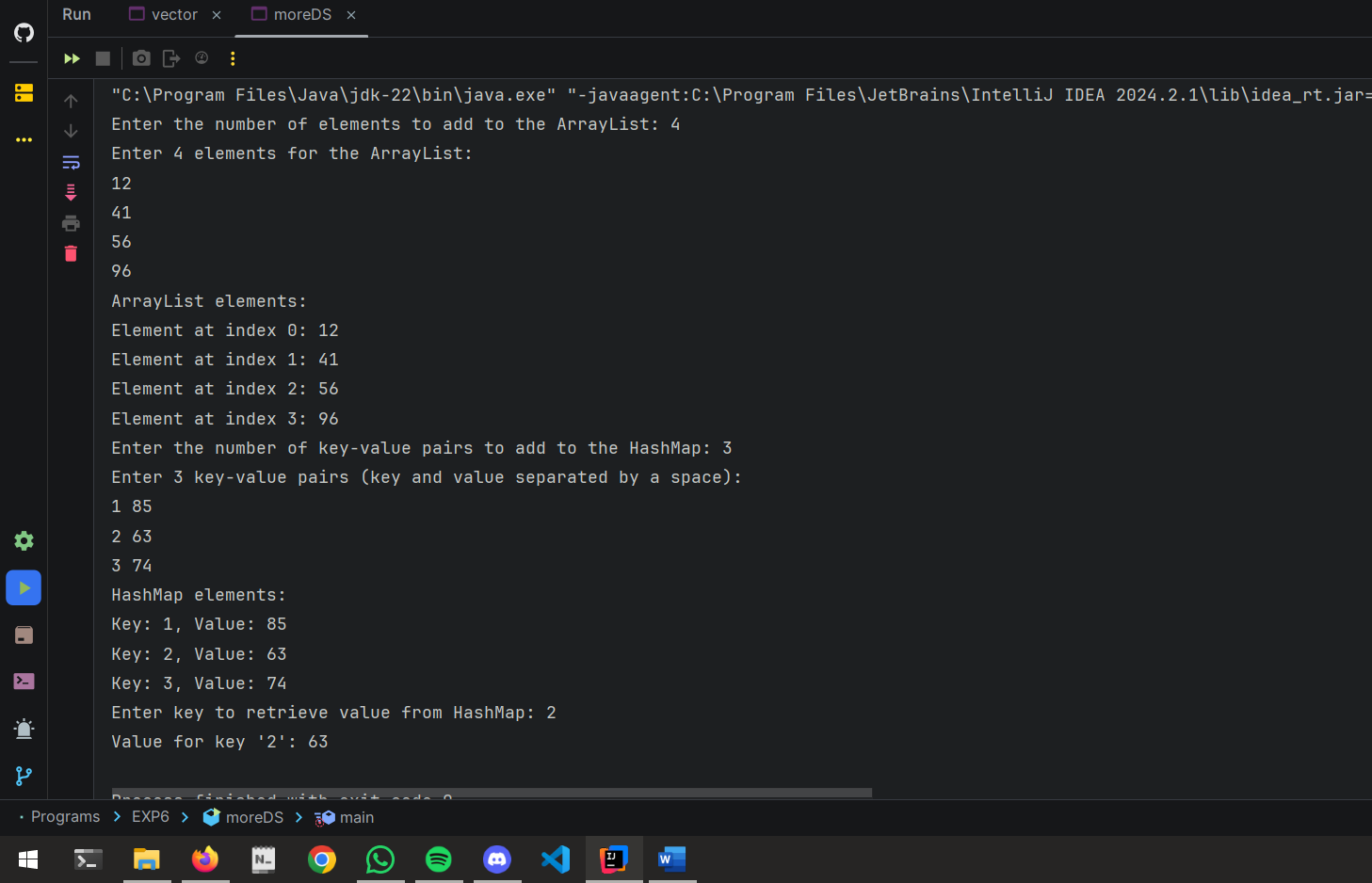


2) Implementation of Array List, Map etc.

CODE:

package EXP6;  
  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.Scanner;  
  
public class moreDS {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 ArrayList<String> arrayList = new ArrayList<>();  
  
 System.*out*.print("Enter the number of elements to add to the ArrayList: ");  
 int listSize = scanner.nextInt();  
 scanner.nextLine();  
 System.*out*.println("Enter " + listSize + " elements for the ArrayList:");  
 for (int i = 0; i < listSize; i++) {  
 arrayList.add(scanner.nextLine());  
 }  
  
 System.*out*.println("ArrayList elements:");  
 for (int i = 0; i < arrayList.size(); i++) {  
 System.*out*.println("Element at index " + i + ": " + arrayList.get(i));  
 }  
  
 HashMap<String, String> hashMap = new HashMap<>();  
  
 System.*out*.print("Enter the number of key-value pairs to add to the HashMap: ");  
 int mapSize = scanner.nextInt();  
 scanner.nextLine();  
 System.*out*.println("Enter " + mapSize + " key-value pairs (key and value separated by a space):");  
 for (int i = 0; i < mapSize; i++) {  
 String[] entry = scanner.nextLine().split(" ");  
 if (entry.length == 2) {  
 hashMap.put(entry[0], entry[1]);  
 } else {  
 System.*out*.println("Invalid input! Please enter a key and a value.");  
 i--;  
 }  
 }  
  
 System.*out*.println("HashMap elements:");  
 for (String key : hashMap.keySet()) {  
 System.*out*.println("Key: " + key + ", Value: " + hashMap.get(key));  
 }  
  
 System.*out*.print("Enter key to retrieve value from HashMap: ");  
 String keyToRetrieve = scanner.nextLine();  
 if (hashMap.containsKey(keyToRetrieve)) {  
 System.*out*.println("Value for key '" + keyToRetrieve + "': " + hashMap.get(keyToRetrieve));  
 } else {  
 System.*out*.println("Key not found!");  
 }  
  
 scanner.close();  
 }  
}

OUPTUT:

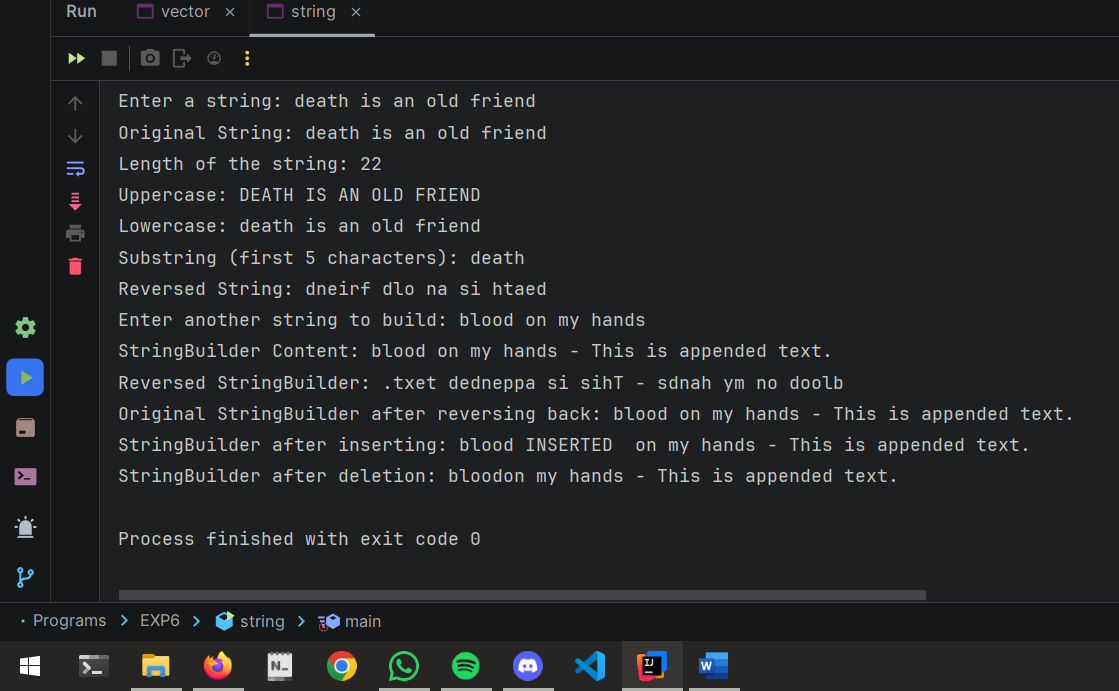


3) Implementation of String and String Builder Class etc.

CODE:

package EXP6;  
  
import java.util.Scanner;  
  
public class string {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter a string: ");  
 String inputString = scanner.nextLine();  
  
 System.*out*.println("Original String: " + inputString);  
 System.*out*.println("Length of the string: " + inputString.length());  
 System.*out*.println("Uppercase: " + inputString.toUpperCase());  
 System.*out*.println("Lowercase: " + inputString.toLowerCase());  
 System.*out*.println("Substring (first 5 characters): " + inputString.substring(0, 5));  
 System.*out*.println("Reversed String: " + *reverseString*(inputString));  
  
 System.*out*.print("Enter another string to build: ");  
 StringBuilder stringBuilder = new StringBuilder(scanner.nextLine());  
  
 stringBuilder.append(" - This is appended text.");  
 System.*out*.println("StringBuilder Content: " + stringBuilder);  
 System.*out*.println("Reversed StringBuilder: " + stringBuilder.reverse());  
 stringBuilder.reverse();  
 System.*out*.println("Original StringBuilder after reversing back: " + stringBuilder);  
 stringBuilder.insert(5, " INSERTED ");  
 System.*out*.println("StringBuilder after inserting: " + stringBuilder);  
 stringBuilder.delete(5, 16);  
 System.*out*.println("StringBuilder after deletion: " + stringBuilder);  
  
 scanner.close();  
 }  
  
 public static String reverseString(String str) {  
 return new StringBuilder(str).reverse().toString();  
 }  
}

OUTPUT:



4) Banking distributed application with Thread Synchronization

CODE:

package EXP5;  
  
class BankAccount {  
 private int balance;  
  
 public BankAccount(int initialBalance) {  
 this.balance = initialBalance;  
 }  
  
 public synchronized void deposit(int amount) {  
 balance += amount;  
 System.*out*.println("Deposited: " + amount + ", New Balance: " + balance);  
 }  
  
 public synchronized void withdraw(int amount) {  
 if (balance >= amount) {  
 balance -= amount;  
 System.*out*.println("Withdrew: " + amount + ", New Balance: " + balance);  
 } else {  
 System.*out*.println("Insufficient funds to withdraw: " + amount + ", Current Balance: " + balance);  
 }  
 }  
  
 public int getBalance() {  
 return balance;  
 }  
}  
  
class DepositThread extends Thread {  
 private BankAccount account;  
 private int amount;  
  
 public DepositThread(BankAccount account, int amount) {  
 this.account = account;  
 this.amount = amount;  
 }  
  
 @Override  
 public void run() {  
 account.deposit(amount);  
 }  
}  
  
class WithdrawThread extends Thread {  
 private BankAccount account;  
 private int amount;  
  
 public WithdrawThread(BankAccount account, int amount) {  
 this.account = account;  
 this.amount = amount;  
 }  
  
 @Override  
 public void run() {  
 account.withdraw(amount);  
 }  
}  
  
class threadBank {  
 public static void main(String[] args) {  
 BankAccount account = new BankAccount(1000);  
  
 DepositThread deposit1 = new DepositThread(account, 500);  
 WithdrawThread withdraw1 = new WithdrawThread(account, 200);  
 WithdrawThread withdraw2 = new WithdrawThread(account, 800);  
  
 deposit1.start();  
 withdraw1.start();  
 withdraw2.start();  
  
 try {  
 deposit1.join();  
 withdraw1.join();  
 withdraw2.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 System.*out*.println("Final Balance: " + account.getBalance());  
 }  
}

OUTPUT:

