**==================== Unique Elements ===================**

=> Here we want to find the no.of unique elements in the array and also the array of unique elements.

So we add the elements to the Hashset because Hashset won’t allow duplicate elements.

The size of the hashset will be the no.of unique elements and we manipulate the given array with elements in the hashset to get the array.

Time Complexity: O(n)

Space Complexity: O(n)

=> To achieve space complexity we use two pointer technique and manipulate the array according to the indices.

Time Complexity: O(n)

Space Complexity: O(1)

**Program**

package com.mycompany.app.DataStructures;

import java.util.HashSet;

import java.util.Scanner;

import java.util.Set;

public class UniqueElements{

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int[] arr = new int[n];

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

arr[i] = sc.nextInt();

}

int i=0;

for(int j=1;j<n;j++) {

if(arr[i]!=arr[j]) {

i++;

arr[i]=arr[j];

}

}

System.out.println("The number of unique elements are: "+(i+1));

System.out.println("The array after updation is: ");

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

System.out.print(arr[k]+" ");

}

}

}

**================= Second Largest ELement ================**

=> Here we try to find the second largest element in the array.

To achieve that first we check if the array size is less than 2 , because there will be no second element for one number.

Later we maintain two variables to hold up the first largest and the second largest.

We update them accordingly to get the result.

**Program**

package com.mycompany.app.DataStructures;

import java.util.Scanner;

public class SecondLargestElement{

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int arr[] = new int[n];

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

arr[i] = sc.nextInt();

}

if(n==1) {

System.out.println(arr[0]+" "+-1);

}

else {

int largest = arr[0], secondLargest = Integer.MIN\_VALUE;

for(int i=1;i<n;i++) {

if(arr[i]>largest) {

int temp = largest;

largest = arr[i];

secondLargest = temp;

}

if(arr[i]>secondLargest && arr[i]!=largest) {

secondLargest = arr[i];

}

}

System.out.println("Largest: "+largest+" SecondLargest: "+secondLargest);

}

}

}

**======================== SQL ==========================**

=> We have connected the database to the project using maven dependencies.

This way we can access the database from the project instead of the SQL editor.

We have added the dependencies and also connected the database with URL, username and password.

The localhost port number for database is 3306.

We have performed the CRUD operations on the database using python.

**Program**

package com.mycompany.app.Databases;

import java.sql.\*;

public class MySQLCRUDExample {

private static final String URL = "jdbc:mysql://localhost:3306/employeemanagement";

private static final String USERNAME = "root";

private static final String PASSWORD = "root";

//Generate comment for all the instance variables

private static Connection connection;

private static Statement statement;

private static ResultSet resultSet;

private static PreparedStatement preparedStatement;

public static void main(String[] args) {

try {

connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);

createRecord();

readRecord();

updateRecord();

deleteRecord();

} catch (SQLException e) {

e.printStackTrace();

}finally{

try {

if (resultSet != null) {

resultSet.close();

}

if (statement != null) {

statement.close();

}

if (connection != null) {

connection.close();

}

if(preparedStatement != null){

preparedStatement.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

//following is my create table statement

/\*create table employees(

id int auto\_increment primary key,

first\_name varchar(50),

last\_name varchar(50),

email varchar(100),

hire\_date DATE,

salary decimal(10,2)

); \*/

private static void createRecord(){

String query = "INSERT INTO employees (first\_name, last\_name, email, hire\_date, salary) VALUES (?, ?, ?, ?, ?)";

try {

preparedStatement = connection.prepareStatement(query);

preparedStatement.setString(1, "Jinesh");

preparedStatement.setString(2, "Doe");

preparedStatement.setString(3, "john.doe@example.com");

preparedStatement.setDate(4, new Date(System.currentTimeMillis()));

preparedStatement.setDouble(5, 50000.00);

preparedStatement.executeUpdate();

System.out.println("Record created successfully.");

} catch (SQLException e) {

e.printStackTrace();

}

}

private static void readRecord(){

String query = "SELECT \* FROM employees";

try {

statement = connection.createStatement();

resultSet = statement.executeQuery(query);

while (resultSet.next()) {

System.out.println(resultSet.getString("first\_name") + " " + resultSet.getString("last\_name"));

}

} catch (SQLException e) {

e.printStackTrace();

}

}

private static void updateRecord(){

String query = "UPDATE employees SET salary = ? WHERE id = ?";

try {

preparedStatement = connection.prepareStatement(query);

preparedStatement.setDouble(1, 60000.00);

preparedStatement.setInt(2, 4);

preparedStatement.executeUpdate();

System.out.println("Record updated successfully.");

} catch (SQLException e) {

e.printStackTrace();

}

}

private static void deleteRecord(){

String query = "DELETE FROM employees WHERE id = ?";

try {

preparedStatement = connection.prepareStatement(query);

preparedStatement.setInt(1, 1);

preparedStatement.executeUpdate();

System.out.println("Record deleted successfully.");

} catch (SQLException e) {

e.printStackTrace();

}

}

}