**Experiment – 21**

**Aim: Write a program in java to sort ten elements of array in descending order.**

import java.util.Scanner;

import java.util.Arrays;

import java.util.Collections;

public class DescendingSort {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Integer[] arr = new Integer[10]; // Using Integer for Collections.reverseOrder()

// Input 10 elements

System.out.println("Enter 10 numbers:");

for (int i = 0; i < 10; i++) {

arr[i] = sc.nextInt();

}

// Sort in descending order

Arrays.sort(arr, Collections.reverseOrder());

// Display sorted array

System.out.println("Array in descending order:");

for (int num : arr) {

System.out.print(num + " ");

}

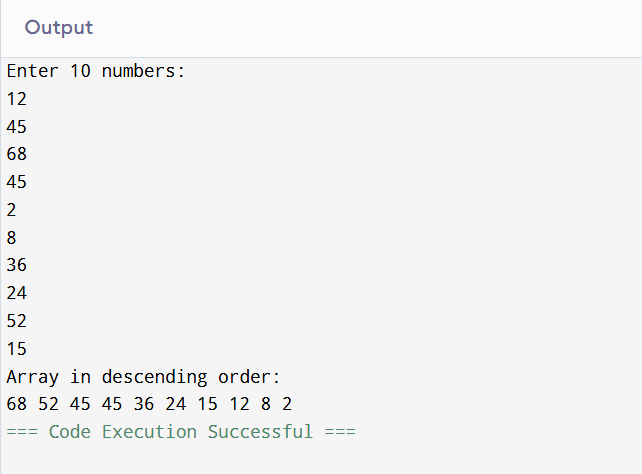
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**

****

**Experiment – 22**

**Aim: Write a Program in java Create a array of 17 elements in 5 rows. And calculate sum of all elements.**

import java.util.Scanner;

public class ArraySum2D {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int rows = 5;

int cols = 4; // We'll handle the last element separately

int[][] arr = new int[rows][cols];

int sum = 0;

System.out.println("Enter 17 elements:");

// Input first 16 elements (4 elements in each of the first 4 rows)

int count = 0;

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

for (int j = 0; j < cols; j++) {

// Stop after 17 elements

if (count == 17) break;

System.out.print("Element [" + i + "][" + j + "]: ");

arr[i][j] = sc.nextInt();

sum += arr[i][j];

count++;

}

}

// Display the array and sum

System.out.println("\nArray Elements:");

count = 0;

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

for (int j = 0; j < cols; j++) {

if (count == 17) break;

System.out.print(arr[i][j] + "\t");

count++;

}

System.out.println();

}

System.out.println("\nSum of all 17 elements: " + sum);

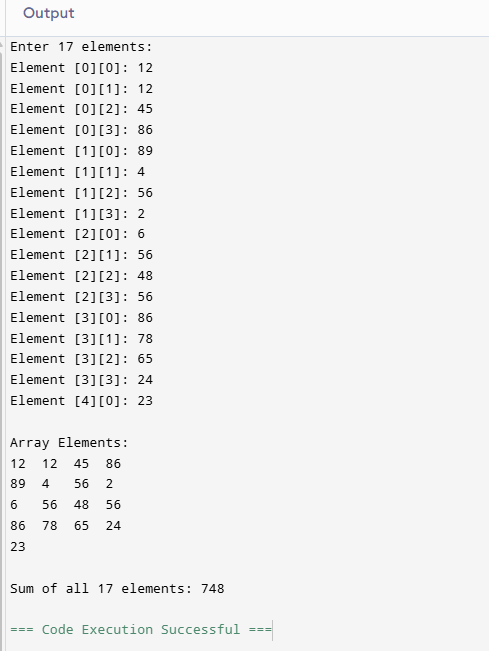
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**

****

**Experiment – 23**

**Aim: Write a program to find multiplication of two 3X3 matrices.**

import java.util.Scanner;

public class MatrixMultiplication {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int[][] A = new int[3][3];

int[][] B = new int[3][3];

int[][] C = new int[3][3]; // Resultant matrix

// Input matrix A

System.out.println("Enter elements of first 3x3 matrix (A):");

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

System.out.print("A[" + i + "][" + j + "]: ");

A[i][j] = sc.nextInt();

}

}

// Input matrix B

System.out.println("Enter elements of second 3x3 matrix (B):");

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

System.out.print("B[" + i + "][" + j + "]: ");

B[i][j] = sc.nextInt();

}

}

// Multiply matrices: C = A x B

for (int i = 0; i < 3; i++) { // row of A

for (int j = 0; j < 3; j++) { // column of B

C[i][j] = 0;

for (int k = 0; k < 3; k++) { // dot product

C[i][j] += A[i][k] \* B[k][j];

}

}

}

// Display result matrix

System.out.println("Resultant Matrix (A x B):");

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

System.out.print(C[i][j] + "\t");

}

System.out.println();

}

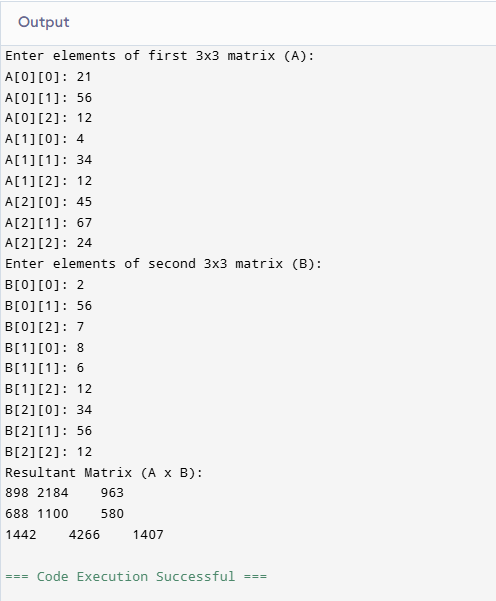
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 24**

**Aim: Write a program to print transpose of a matrix.**

import java.util.Scanner;

public class MatrixTranspose {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Input matrix size

System.out.print("Enter number of rows: ");

int rows = sc.nextInt();

System.out.print("Enter number of columns: ");

int cols = sc.nextInt();

int[][] matrix = new int[rows][cols];

int[][] transpose = new int[cols][rows];

// Input matrix elements

System.out.println("Enter the elements of the matrix:");

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

for (int j = 0; j < cols; j++) {

System.out.print("Element [" + i + "][" + j + "]: ");

matrix[i][j] = sc.nextInt();

}

}

// Compute transpose

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

for (int j = 0; j < cols; j++) {

transpose[j][i] = matrix[i][j];

}

}

// Display original matrix

System.out.println("\nOriginal Matrix:");

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

for (int j = 0; j < cols; j++) {

System.out.print(matrix[i][j] + "\t");

}

System.out.println();

}

// Display transpose

System.out.println("\nTranspose of the Matrix:");

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

for (int j = 0; j < rows; j++) {

System.out.print(transpose[i][j] + "\t");

}

System.out.println();

}

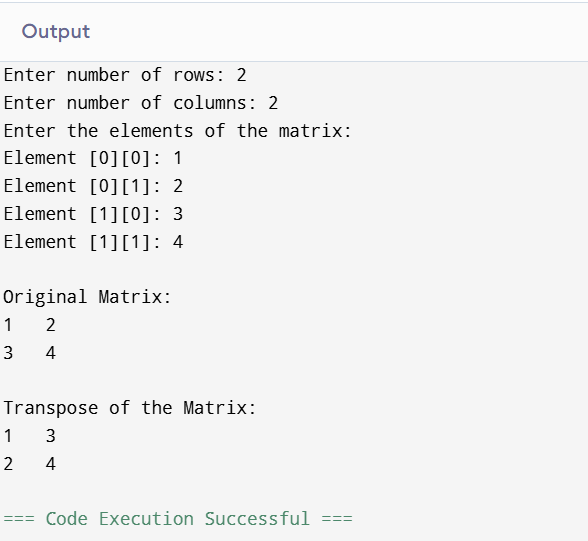
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment- 25**

**What are the two types of Exceptions in Java? What are the differences between them?**

**Types of Exceptions in Java**

In Java, there are two main types of exceptions:

**1. Checked Exceptions**

* These are checked by the compiler at compile-time.
* The programmer is required to handle them using try-catch or throws.
* They usually occur due to external factors like file handling, database access, etc.

**2. Unchecked Exceptions**

* These are not checked by the compiler at compile-time.
* It is not mandatory to handle them.
* They occur due to programming errors like dividing by zero or accessing a null object.

**Experiment – 26**

**What are the Memory Allocations available in Java?**

**Memory Allocations in Java**

**1. Heap Memory**

* **Used to store objects and instance variables.**
* **This is the area where all class instances and arrays are allocated.**
* **Managed by Garbage Collector.**

**2. Stack Memory**

* **Used for method execution and to store local variables and function call information.**
* **Each thread has its own stack.**
* **Memory is allocated and deallocated in LIFO (Last In First Out) order.**

**3. Method Area (or Metaspace in Java 8+)**

* **Stores class-level data, such as:**
  + **Class metadata**
  + **Static variables**
  + **Method definitions**
* **Shared among all threads.**

**4. Program Counter (PC) Register**

* **Each thread has its own PC register.**
* **It holds the address of the current instruction being executed.**
* **It helps in thread execution tracking.**

**5. Native Method Stack**

* **Used when Java interacts with native (non-Java) code, such as C/C++.**
* **Stores native method calls.**

**Experiment – 27**

**Explain final, finally, finalize.**

Here’s a clear and simple explanation of the differences between final, finally, and finalize in Java — three similar-sounding terms with completely different meanings:

**✅ final**

* **Keyword** in Java.
* Used to **restrict changes**.

**🔸Usage:**

1. **Final Variable** – cannot be changed after initialization.
2. **Final Method** – cannot be overridden by subclasses.
3. **Final Class** – cannot be extended (inherited).

**✅ finally**

* **Block** used with try-catch.
* Always **executes** after the try block, **whether an exception occurs or not**.
* Used for **clean-up activities** like closing files, releasing resources, etc.

**✅ finalize()**

* **Method** in the Object class.
* Called by **Garbage Collector** before destroying an object.
* Used to **perform clean-up** before an object is removed from memory.

**Experiment -28**

**27. What is a singleton class in Java? And How to break the singleton class object? A singleton class allows only one object to be created for the entire program.**

**✅ How to Create It:**

**public class Singleton {**

**private static Singleton instance;**

**private Singleton() {} // Private constructor**

**public static Singleton getInstance() {**

**if (instance == null)**

**instance = new Singleton();**

**return instance;**

**}**

**}**

**❌ How Singleton Can Be Broken:**

1. **Reflection**
2. **Serialization**
3. **Cloning**

**✅ How to Prevent Breaking:**

| **Break Method** | **Fix** |
| --- | --- |
| **Reflection** | **Throw error in constructor** |
| **Serialization** | **Use readResolve() method** |
| **Cloning** | **Override clone() and block it** |
|  |  |

**Experiment – 29**

**Differentiate between instance and local variables.**

**✅ Instance Variables**

* **Declared inside a class, but outside any method.**
* **Belong to the object of the class.**
* **Exist as long as the object exists.**
* **Can have access modifiers (like private, public).**
* **Automatically initialized to default values.**

**✅ Local Variables**

* **Declared inside a method, constructor, or block.**
* **Exist only during method execution.**
* **Belong to the method, not the object.**
* **Cannot have access modifiers.**
* **Must be explicitly initialized before use.**

**Experiment -30**

**Explain the types of Exceptions in java?**

**Types of Exceptions in Java**

**Java exceptions are broadly classified into two main types:**

**🔹 1. Checked Exceptions**

* **Checked at compile-time.**
* **The compiler forces the programmer to handle them.**
* **Often occur due to external factors like file handling, database access, etc.**

**✅ Examples:**

* **IOException**
* **SQLException**
* **FileNotFoundException**

**🔹 2. Unchecked Exceptions**

* **Checked at runtime (not at compile-time).**
* **The compiler does not force handling them.**
* **Usually caused by logic errors in the program.**

**✅ Examples:**

* **NullPointerException**
* **ArithmeticException**
* **ArrayIndexOutOfBoundsException**

**🔹 3. Errors (Not Exceptions, but Throwable)**

* **Serious problems that cannot be recovered by the program.**
* **Occur due to system failures, like memory issues.**

**✅ Examples:**

* **OutOfMemoryError**
* **StackOverflowError**

**Experiment – 31**

**Can you call a constructor of a class inside another constructor of same class?**

**✅ Yes, you can call a constructor of a class inside another constructor of the same class in Java.**

**This is called constructor chaining, and it is done using the this() keyword.**

**🔸 Syntax:**

**public ClassName() {**

**this(parameter); // Calls another constructor**

**}**

**🔸 Example:**

**public class Student {**

**String name;**

**int age;**

**// Constructor 1**

**public Student() {**

**this("Unknown", 0); // Calls Constructor 2**

**System.out.println("Default constructor called");**

**}**

**// Constructor 2**

**public Student(String name, int age) {**

**this.name = name;**

**this.age = age;**

**System.out.println("Parameterized constructor called");**

**}**

**}**

**Experiment – 32**

**Explain the Java thread lifecycle?**

**✅ Java Thread Lifecycle**

**In Java, a thread goes through five main states during its lifecycle:**

**🔹 1. New**

* **The thread is created using Thread class.**
* **It is not yet started.**

**java**

**CopyEdit**

**Thread t = new Thread(); // New state**

**🔹 2. Runnable**

* **The thread is ready to run after calling start().**
* **It is waiting for CPU to be assigned.**

**🔹 3. Running**

* **The thread is currently executing its task.**

**🔹 4. Blocked / Waiting / Timed Waiting**

* **The thread is temporarily inactive.**
* **Waiting for a resource or time to pass (e.g., sleep(), join()).**

**🔹 5. Terminated (Dead)**

* **The thread has finished execution or has been stopped due to an error.**

**Experiment – 33**

**Aim: Create a class to calculate Area of circle with one data member to store the radius and another to store area value.**

**Create method members**

**1. init - to input radius from user**

**2. calc - to calculate area**

**3. display- to display area**

import java.util.Scanner;

public class Circle {

double radius;

double area;

void init() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the radius of the circle: ");

radius = sc.nextDouble();

}

void calc() {

area = Math.PI \* radius \* radius;

}

void display() {

System.out.printf("The area of the circle is: %.2f\n", area);

}

public static void main(String[] args) {

Circle c = new Circle();

c.init();

c.calc();

c.display();

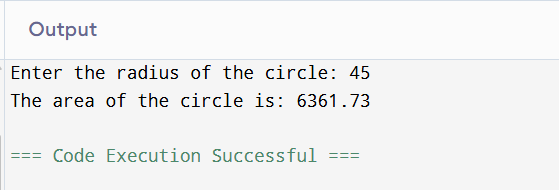
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 34**

**Aim: Write a program Create a class MathOperation with two data member X and Y to store the operand and third data member R to store result of operation.**

**Create method members**

**● init - to input X and Y from user**

**● add - to add X and Y and store in R**

**● multiply - to multiply X and Y and store in R**

**● power - to calculate X Y and store in R**

**● display- to display Result R**

import java.util.Scanner;

public class MathOperation {

double X, Y, R;

// Method to input X and Y from the user

void init() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter value for X: ");

X = sc.nextDouble();

System.out.print("Enter value for Y: ");

Y = sc.nextDouble();

}

// Method to add X and Y

void add() {

R = X + Y;

}

// Method to multiply X and Y

void multiply() {

R = X \* Y;

}

// Method to calculate X to the power Y

void power() {

R = Math.pow(X, Y);

}

// Method to display result R

void display() {

System.out.println("Result (R) = " + R);

}

// Main method to run the program

public static void main(String[] args) {

MathOperation mo = new MathOperation();

mo.init(); // Input values

mo.add(); // Perform addition

mo.display(); // Display result

mo.multiply(); // Perform multiplication

mo.display(); // Display result

mo.power(); // Perform power operation

mo.display(); // Display result

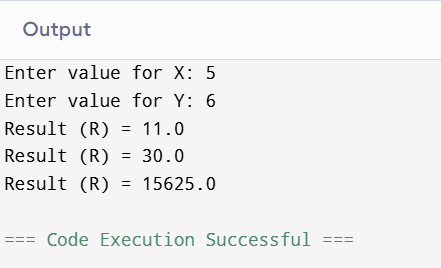
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**

****

**Experiment – 35**

**Aim: Create a class MathOperation containing method ‘multiply’ to calculate multiplication of following arguments.**

a. two integers

b. three float

c. all elements of array

d. one double and one integer 

import java.util.Scanner;

import java.util.Arrays;

public class MathOperation {

// a. Multiply two integers

void multiply(int a, int b) {

int result = Math.multiplyExact(a, b); // Using Math library

System.out.println("Multiplication of two integers: " + result);

}

// b. Multiply three floats

void multiply(float a, float b, float c) {

float result = a \* b \* c; // No library method for float multiply

System.out.println("Multiplication of three floats: " + result);

}

// c. Multiply all elements of an integer array

void multiply(int[] arr) {

int result = Arrays.stream(arr).reduce(1, (x, y) -> x \* y); // Using Java Stream API

System.out.println("Multiplication of array elements: " + result);

}

// d. Multiply one double and one integer

void multiply(double a, int b) {

double result = a \* b; // Simple arithmetic

System.out.println("Multiplication of double and integer: " + result);

}

// Main method to test all overloaded methods

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

MathOperation mo = new MathOperation();

// a. Multiply two integers

System.out.print("Enter two integers: ");

int int1 = sc.nextInt();

int int2 = sc.nextInt();

mo.multiply(int1, int2);

// b. Multiply three floats

System.out.print("Enter three floats: ");

float f1 = sc.nextFloat();

float f2 = sc.nextFloat();

float f3 = sc.nextFloat();

mo.multiply(f1, f2, f3);

// c. Multiply all elements of an array

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

int size = sc.nextInt();

int[] array = new int[size];

System.out.println("Enter " + size + " array elements:");

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

array[i] = sc.nextInt();

}

mo.multiply(array);

// d. Multiply one double and one integer

System.out.print("Enter a double and an integer: ");

double d = sc.nextDouble();

int i = sc.nextInt();

mo.multiply(d, i);

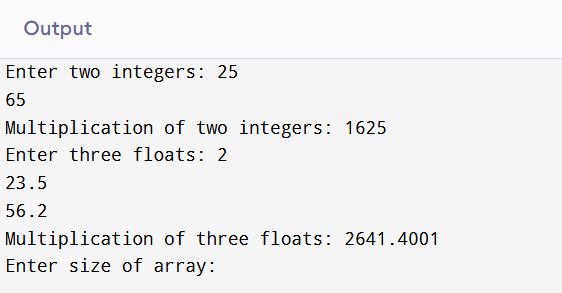
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 36**

**Aim: Write a program in java to find prime numbers in an array.**

import java.util.Scanner

public class PrimeInArray {

// Function to check if a number is prime

static boolean isPrime(int num) {

if (num <= 1)

return false;

for (int i = 2; i <= Math.sqrt(num); i++) {

if (num % i == 0)

return false;

}

return true;

}

// Main method

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Input array size

System.out.print("Enter number of elements in the array: ");

int n = sc.nextInt();

int[] arr = new int[n];

// Input array elements

System.out.println("Enter " + n + " elements:");

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

arr[i] = sc.nextInt();

}

// Check and print prime numbers

System.out.println("Prime numbers in the array are:");

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

if (isPrime(arr[i])) {

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

}

}

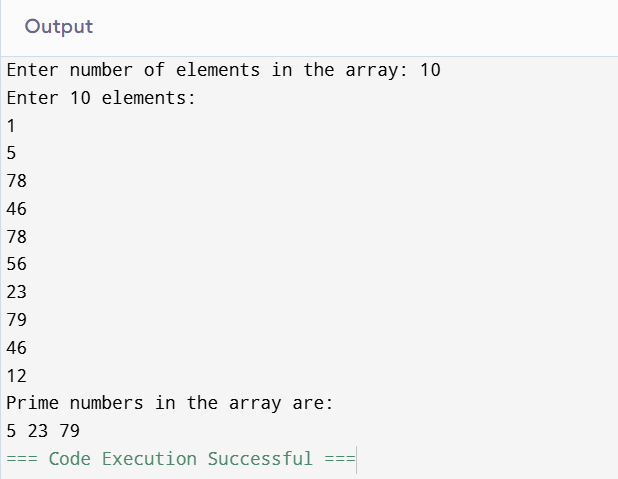
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**

****

**Experiment- 37**

**Aim:** **Write a Java program to create an ArrayList, add some colors (as strings), and print the collection.**

import java.util.ArrayList;

public class ColorList {

public static void main(String[] args) {

// Create an ArrayList of Strings

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

// Add color names to the ArrayList

colors.add("Red");

colors.add("Green");

colors.add("Blue");

colors.add("Yellow");

colors.add("Orange");

// Print the ArrayList

System.out.println("Colors in the list:");

for (String color : colors) {

System.out.println(color);

}

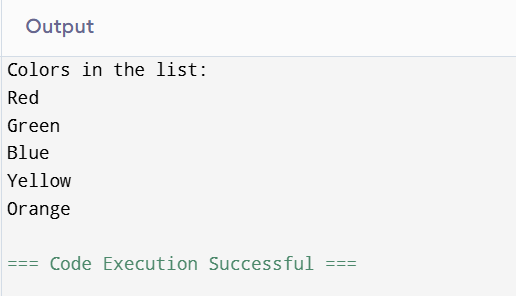
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment -38**

**Aim: Write a Java program to iterate through all elements in an ArrayList.**

import java.util.ArrayList;

import java.util.Iterator;

public class IterateArrayList {

public static void main(String[] args) {

// Create an ArrayList and add elements

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

items.add("Apple");

items.add("Banana");

items.add("Mango");

items.add("Grapes");

System.out.println("Using for loop:");

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

System.out.println(items.get(i));

}

System.out.println("\nUsing for-each loop:");

for (String item : items) {

System.out.println(item);

}

System.out.println("\nUsing Iterator:");

Iterator<String> it = items.iterator();

while (it.hasNext()) {

System.out.println(it.next());

}

System.out.println("\nUsing forEach() method (Java 8+):");

items.forEach(System.out::println);

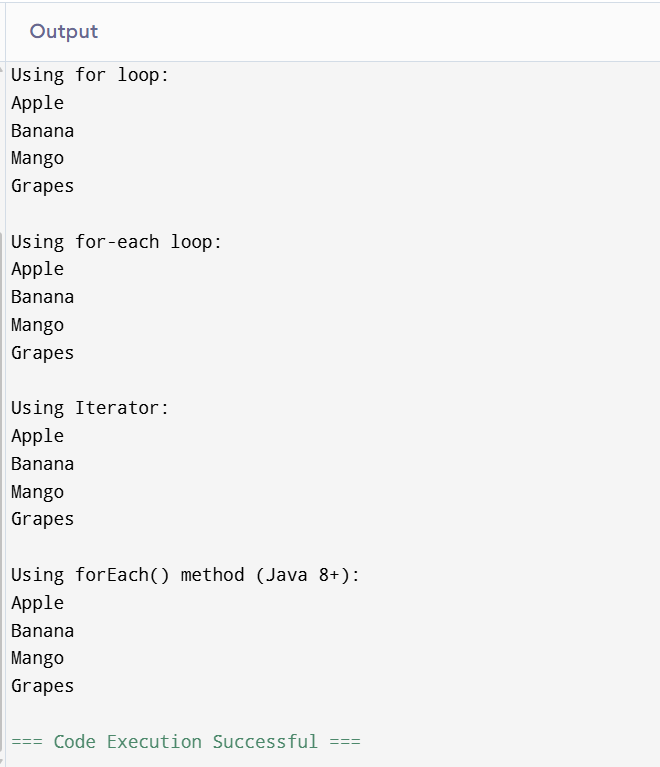
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 39**

**Aim: Write a Java program to insert an element into the ArrayList at the first position.**

import java.util.ArrayList;

public class InsertFirst {

public static void main(String[] args) {

// Create an ArrayList and add some elements

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

colors.add("Red");

colors.add("Green");

colors.add("Blue");

System.out.println("Original ArrayList: " + colors);

// Insert a new element at the first position (index 0)

colors.add(0, "Yellow");

System.out.println("After inserting at first position: " + colors);

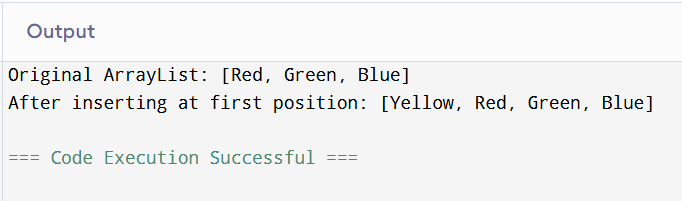
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment- 40**

**Aim: Write a Java program to retrieve an element at a specified index from a given ArrayList.**

import java.util.ArrayList;

import java.util.Scanner;

public class RetrieveElement {

public static void main(String[] args) {

// Create and populate the ArrayList

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

fruits.add("Apple");

fruits.add("Banana");

fruits.add("Mango");

fruits.add("Grapes");

fruits.add("Orange");

// Display the list

System.out.println("Fruits List: " + fruits);

// Ask the user for index input

Scanner sc = new Scanner(System.in);

System.out.print("Enter index (0 to " + (fruits.size() - 1) + ") to retrieve element: ");

int index = sc.nextInt();

// Check for valid index and retrieve the element

if (index >= 0 && index < fruits.size()) {

String item = fruits.get(index);

System.out.println("Element at index " + index + ": " + item);

} else {

System.out.println("Invalid index!");

}

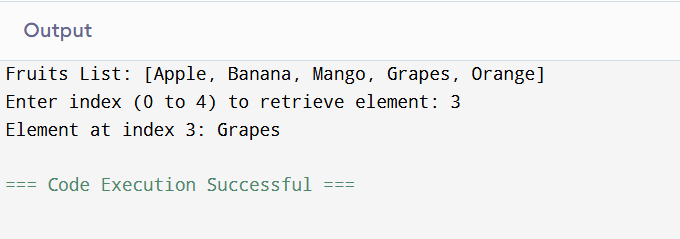
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**

****

**Experiment -41**

**Aim:** **Write a Java program to update an ArrayList element by a given element.**

import java.util.ArrayList;

import java.util.Scanner;

public class UpdateElement {

public static void main(String[] args) {

// Create and populate the ArrayList

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

colors.add("Red");

colors.add("Green");

colors.add("Blue");

colors.add("Yellow");

System.out.println("Original ArrayList: " + colors);

Scanner sc = new Scanner(System.in);

// Get the element to be replaced

System.out.print("Enter the element to replace: ");

String oldElement = sc.nextLine();

// Get the new element

System.out.print("Enter the new element: ");

String newElement = sc.nextLine();

// Check if the element exists and update

if (colors.contains(oldElement)) {

int index = colors.indexOf(oldElement);

colors.set(index, newElement);

System.out.println("Updated ArrayList: " + colors);

} else {

System.out.println("Element '" + oldElement + "' not found in the list.");

}

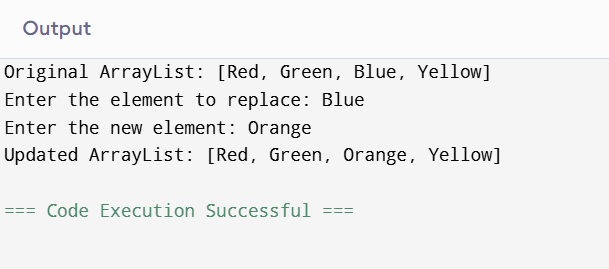
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment-42**

**Aim:** **Write a Java program to remove the third element from an ArrayList.**

import java.util.ArrayList;

public class RemoveThirdElement {

public static void main(String[] args) {

// Create and populate the ArrayList

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

animals.add("Cat");

animals.add("Dog");

animals.add("Elephant");

animals.add("Tiger");

animals.add("Lion");

System.out.println("Original ArrayList: " + animals);

// Remove the third element (index 2)

if (animals.size() >= 3) {

animals.remove(2); // Index starts from 0

System.out.println("ArrayList after removing third element: " + animals);

} else {

System.out.println("ArrayList does not have a third element to remove.");

}

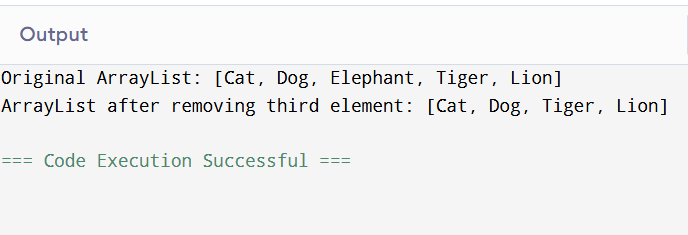
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment-43**

**Aim: Write a Java program to search for an element in an ArrayList.**

import java.util.ArrayList;

import java.util.Scanner;

public class SearchElement {

public static void main(String[] args) {

// Create and populate the ArrayList

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

cities.add("Delhi");

cities.add("Mumbai");

cities.add("Kolkata");

cities.add("Chennai");

cities.add("Bangalore");

// Display the ArrayList

System.out.println("Cities List: " + cities);

// Take input from user

Scanner sc = new Scanner(System.in);

System.out.print("Enter city to search: ");

String search = sc.nextLine();

// Search for the element

if (cities.contains(search)) {

System.out.println(search + " found at index " + cities.indexOf(search));

} else {

System.out.println(search + " not found in the list.");

}

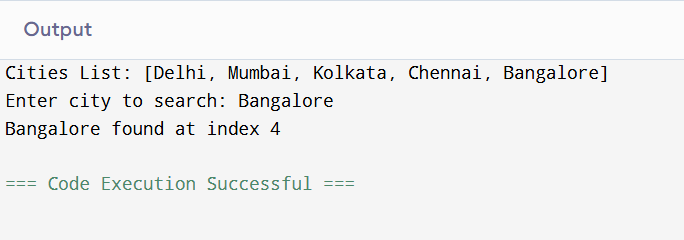
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment -44**

**Aim: Write a Java program to sort a given ArrayList.**

import java.util.ArrayList;

import java.util.Collections;

public class SortArrayList {

public static void main(String[] args) {

// Create and populate the ArrayList

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

names.add("Ravi");

names.add("Amit");

names.add("Zara");

names.add("Kiran");

names.add("Neha");

System.out.println("Original ArrayList: " + names);

// Sort the ArrayList in ascending order

Collections.sort(names);

System.out.println("Sorted ArrayList: " + names);

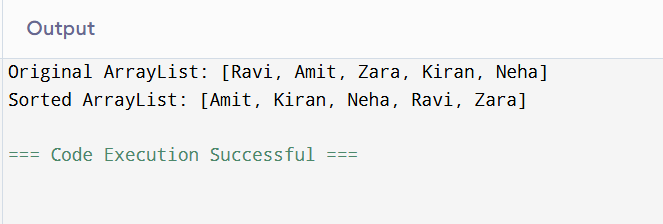
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment-45**

**Aim:** **Write a Java program to copy one array list into another.**

import java.util.ArrayList;

import java.util.Collections;

public class CopyArrayList {

public static void main(String[] args) {

// Source ArrayList

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

sourceList.add("Red");

sourceList.add("Green");

sourceList.add("Blue");

// Destination ArrayList with same size as source

ArrayList<String> destinationList = new ArrayList<>(sourceList.size());

// Fill destinationList with nulls to match source size

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

destinationList.add(null);

}

// Copy elements from sourceList to destinationList

Collections.copy(destinationList, sourceList);

// Display the lists

System.out.println("Source List: " + sourceList);

System.out.println("Destination List after copy: " + destinationList);

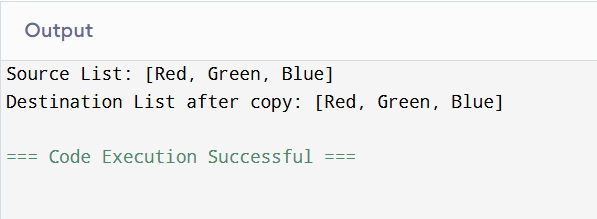
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment- 46**

**Aim: Write a Java program to shuffle elements in an array list.**

import java.util.ArrayList;

import java.util.Collections;

public class ShuffleArrayList {

public static void main(String[] args) {

// Create and populate the ArrayList

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

colors.add("Red");

colors.add("Blue");

colors.add("Green");

colors.add("Yellow");

colors.add("Purple");

System.out.println("Original ArrayList: " + colors);

// Shuffle the list

Collections.shuffle(colors);

System.out.println("Shuffled ArrayList: " + colors);

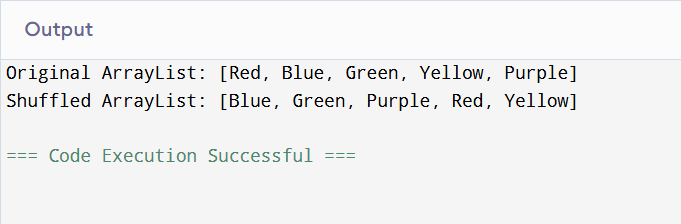
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 47**

**Aim: Write a Java program to reverse elements in an array list.**

import java.util.ArrayList;

import java.util.Collections;

public class ReverseArrayList {

public static void main(String[] args) {

// Create and populate the ArrayList

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

colors.add("Red");

colors.add("Blue");

colors.add("Green");

colors.add("Yellow");

colors.add("Purple");

System.out.println("Original ArrayList: " + colors);

// Reverse the list

Collections.reverse(colors);

System.out.println("Reversed ArrayList: " + colors);

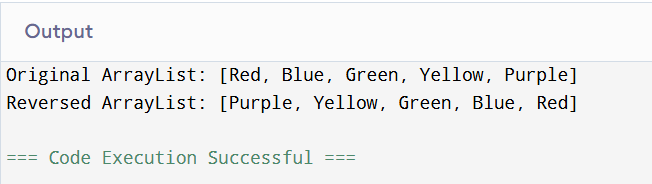
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment -48**

**Aim: Write a Java program to extract a portion of an array list.**

import java.util.ArrayList;

import java.util.List;

public class ExtractSubList {

public static void main(String[] args) {

// Create and populate the ArrayList

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

names.add("Ravi");

names.add("Amit");

names.add("Neha");

names.add("Kiran");

names.add("Zara");

names.add("Nikhil");

System.out.println("Original ArrayList: " + names);

// Extract sublist from index 1 (inclusive) to 4 (exclusive)

List<String> subList = names.subList(1, 4);

System.out.println("Extracted Portion (index 1 to 3): " + subList);

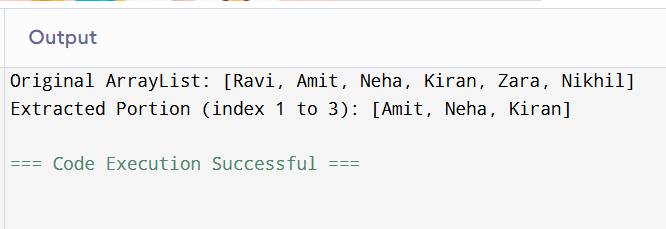
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment-49**

**Aim:** **Write a Java program to compare two array lists.**

import java.util.ArrayList;

public class CompareArrayLists {

public static void main(String[] args) {

// First ArrayList

ArrayList<String> list1 = new ArrayList<>();

list1.add("Red");

list1.add("Green");

list1.add("Blue");

// Second ArrayList

ArrayList<String> list2 = new ArrayList<>();

list2.add("Red");

list2.add("Green");

list2.add("Blue");

// Third ArrayList (different content)

ArrayList<String> list3 = new ArrayList<>();

list3.add("Red");

list3.add("Yellow");

list3.add("Blue");

// Compare list1 and list2

boolean isEqual1 = list1.equals(list2);

System.out.println("list1 and list2 are equal: " + isEqual1);

// Compare list1 and list3

boolean isEqual2 = list1.equals(list3);

System.out.println("list1 and list3 are equal: " + isEqual2);

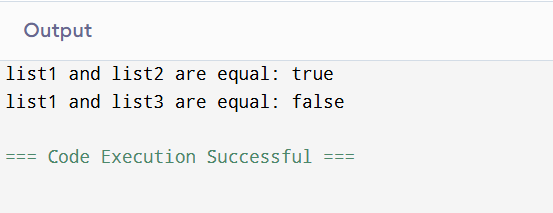
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment-50**

**Aim: Write a Java program that swaps two elements in an array list.**

import java.util.ArrayList;

import java.util.Collections;

public class SwapArrayListElements {

public static void main(String[] args) {

// Create and populate the ArrayList

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

colors.add("Red");

colors.add("Green");

colors.add("Blue");

colors.add("Yellow");

System.out.println("Original ArrayList: " + colors);

// Swap elements at index 1 and 3

Collections.swap(colors, 1, 3);

System.out.println("ArrayList after swapping index 1 and 3: " + colors);

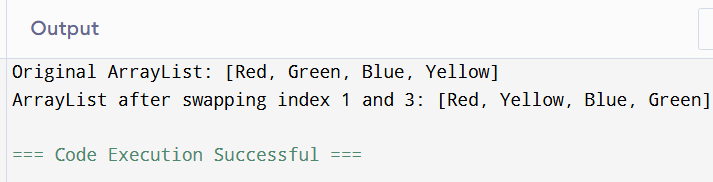
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**

****

**Experiment-51**

**Aim:** **Write a Java program to join two array lists.**

import java.util.ArrayList;

public class JoinArrayLists {

public static void main(String[] args) {

// First ArrayList

ArrayList<String> list1 = new ArrayList<>();

list1.add("Red");

list1.add("Green");

list1.add("Blue");

// Second ArrayList

ArrayList<String> list2 = new ArrayList<>();

list2.add("Yellow");

list2.add("Purple");

list2.add("Orange");

// Join list2 into list1

list1.addAll(list2);

// Display the joined ArrayList

System.out.println("Joined ArrayList: " + list1);

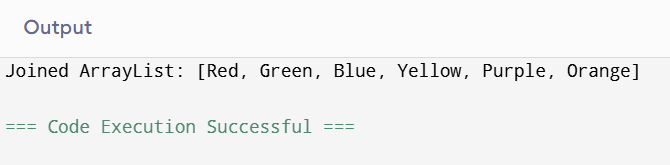
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 52**

**Aim:** **Write a Java program to clone an array list to another array list.**

import java.util.ArrayList;

public class CloneArrayList {

public static void main(String[] args) {

// Original ArrayList

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

originalList.add("Red");

originalList.add("Green");

originalList.add("Blue");

// Clone the ArrayList

@SuppressWarnings("unchecked")

ArrayList<String> clonedList = (ArrayList<String>) originalList.clone();

// Display both lists

System.out.println("Original ArrayList: " + originalList);

System.out.println("Cloned ArrayList: " + clonedList);

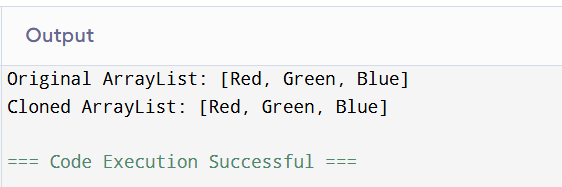
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 53**

**Aim:** **Write a Java program to empty an array list.**

import java.util.ArrayList;

public class EmptyArrayList {

public static void main(String[] args) {

// Create and populate the ArrayList

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

colors.add("Red");

colors.add("Green");

colors.add("Blue");

System.out.println("Original ArrayList: " + colors);

// Empty the ArrayList

colors.clear();

// Display the empty list

System.out.println("ArrayList after clear(): " + colors);

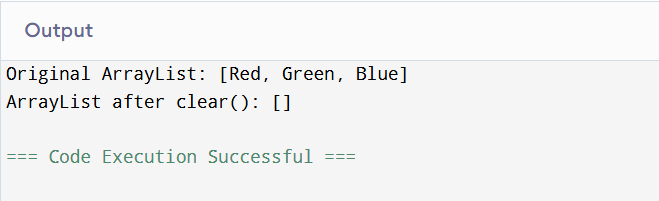
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 54**

**Aim:** **Write a Java program to test whether an array list is empty or not.**

import java.util.ArrayList;

public class CheckArrayListEmpty {

public static void main(String[] args) {

// Create an empty ArrayList

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

// Check if the ArrayList is empty

if (colors.isEmpty()) {

System.out.println("The ArrayList is empty.");

} else {

System.out.println("The ArrayList is not empty.");

}

// Add an element

colors.add("Red");

// Check again after adding an element

if (colors.isEmpty()) {

System.out.println("The ArrayList is empty.");

} else {

System.out.println("The ArrayList is not empty.");

}

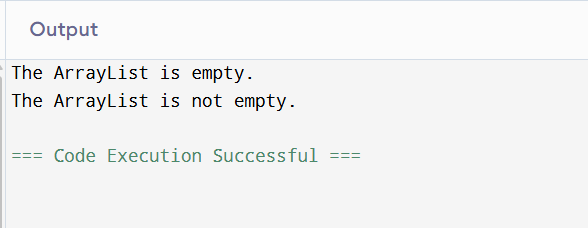
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment-55**

**Aim: Write a Java program for trimming the capacity of an array list.**

import java.util.ArrayList;

public class TrimArrayList {

public static void main(String[] args) {

// Create an ArrayList with initial capacity

ArrayList<String> colors = new ArrayList<>(20);

// Add some elements

colors.add("Red");

colors.add("Green");

colors.add("Blue");

System.out.println("Size before trimToSize(): " + colors.size());

// Trim the capacity to the current size

colors.trimToSize();

System.out.println("ArrayList trimmed to size. New size: " + colors.size());

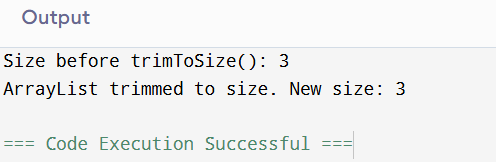
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment -56**

**Aim:** **Write a Java program to increase an array list size.**

import java.util.ArrayList;

public class IncreaseArrayListSize {

public static void main(String[] args) {

// Create an ArrayList with default capacity

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

// Add some elements

colors.add("Red");

colors.add("Green");

colors.add("Blue");

// Print size before increasing capacity

System.out.println("Size before ensureCapacity: " + colors.size());

// Increase the capacity to hold at least 20 elements

colors.ensureCapacity(20);

// Add more elements (optional demonstration)

colors.add("Yellow");

colors.add("Purple");

// Print size after increasing capacity

System.out.println("Size after adding more elements: " + colors.size());

System.out.println("ArrayList: " + colors);

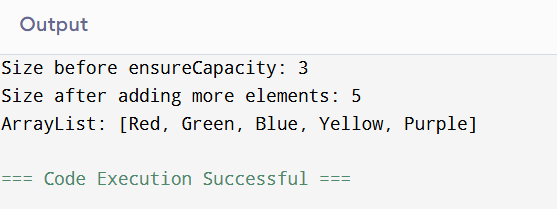
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment-57**

**Aim:** **Write a Java program to replace the second element of an ArrayList with the specified element.**

import java.util.ArrayList;

public class ReplaceSecondElement {

public static void main(String[] args) {

// Create and populate the ArrayList

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

colors.add("Red");

colors.add("Green");

colors.add("Blue");

System.out.println("Original ArrayList: " + colors);

// Replace the second element (index 1) with a new value

colors.set(1, "Yellow");

System.out.println("Updated ArrayList: " + colors);

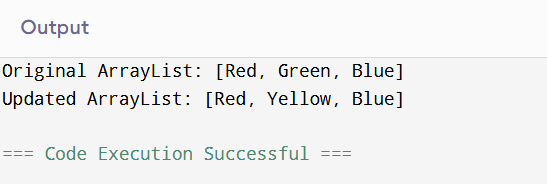
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 58**

**Aim:** **Write a Java program to print all the elements of an ArrayList using the elements' position.**

import java.util.ArrayList;

public class PrintArrayListByIndex {

public static void main(String[] args) {

// Create and populate the ArrayList

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

colors.add("Red");

colors.add("Green");

colors.add("Blue");

colors.add("Yellow");

System.out.println("ArrayList elements using index:");

// Loop through the ArrayList using positions (index)

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

System.out.println("Element at index " + i + ": " + colors.get(i));

}

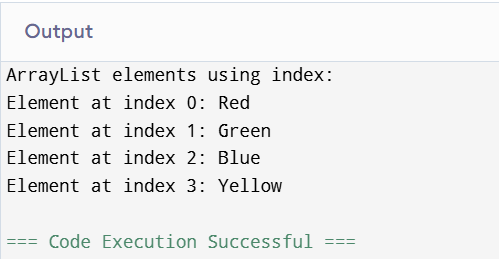
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 59**

**Aim:** **Write a Java program to append a specified element to the end of a linked list.**

import java.util.LinkedList;

public class AppendToLinkedList {

public static void main(String[] args) {

// Create a LinkedList of Strings

LinkedList<String> fruits = new LinkedList<>();

// Add initial elements

fruits.add("Apple");

fruits.add("Banana");

fruits.add("Mango");

System.out.println("Original LinkedList: " + fruits);

// Append an element to the end

fruits.add("Orange");

System.out.println("LinkedList after appending: " + fruits);

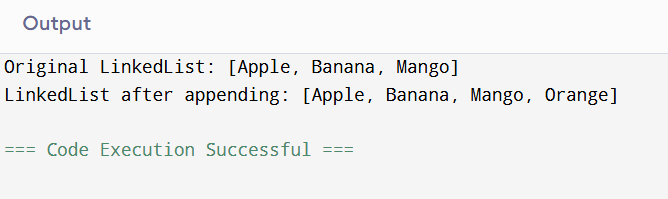
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment- 60**

**Aim:** **Write a Java program to iterate through all elements in a linked list.**

import java.util.LinkedList;

public class IterateLinkedList {

public static void main(String[] args) {

// Create and populate the LinkedList

LinkedList<String> animals = new LinkedList<>();

animals.add("Dog");

animals.add("Cat");

animals.add("Horse");

animals.add("Cow");

System.out.println("Iterating through LinkedList:");

// Using a for loop with index

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

System.out.println("Element at index " + i + ": " + animals.get(i));

}

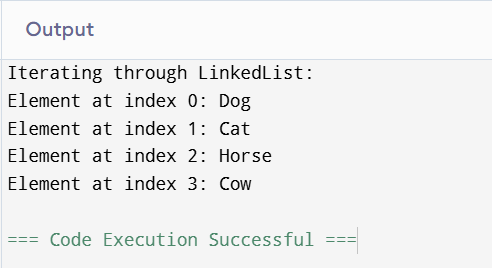
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 61**

**Aim:** **Write a Java program to iterate through all elements starting from a specified position in a linked list.**

import java.util.LinkedList;

public class IterateFromPosition {

public static void main(String[] args) {

// Create and populate the LinkedList

LinkedList<String> cities = new LinkedList<>();

cities.add("Delhi");

cities.add("Mumbai");

cities.add("Chennai");

cities.add("Kolkata");

cities.add("Bangalore");

// Specify the starting index

int startIndex = 2;

System.out.println("Iterating from index " + startIndex + ":");

// Check for valid index and iterate

if (startIndex >= 0 && startIndex < cities.size()) {

for (int i = startIndex; i < cities.size(); i++) {

System.out.println("Element at index " + i + ": " + cities.get(i));

}

} else {

System.out.println("Invalid starting index!");

}

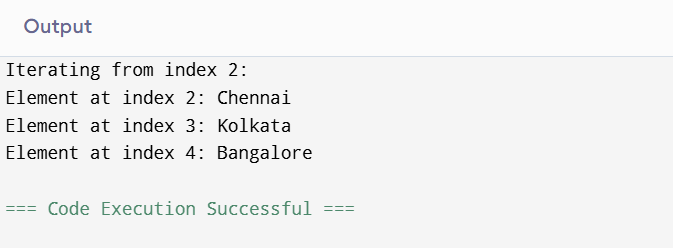
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment- 62**

**Aim: Write a Java program to iterate a linked list in reverse order**

import java.util.LinkedList;

import java.util.ListIterator;

public class ReverseLinkedListIteration {

public static void main(String[] args) {

// Create and populate the LinkedList

LinkedList<String> colors = new LinkedList<>();

colors.add("Red");

colors.add("Green");

colors.add("Blue");

colors.add("Yellow");

System.out.println("LinkedList in reverse order:");

// Create a ListIterator starting from the end

ListIterator<String> iterator = colors.listIterator(colors.size());

// Traverse in reverse

while (iterator.hasPrevious()) {

System.out.println(iterator.previous());

}

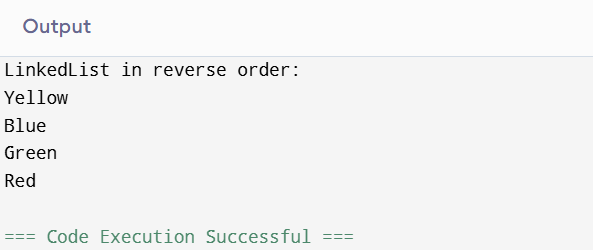
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment -63**

**Aim: Write a Java program to insert a specified element at a given position in a linked list.**

import java.util.LinkedList;

public class InsertAtPosition {

public static void main(String[] args) {

// Create and populate the LinkedList

LinkedList<String> languages = new LinkedList<>();

languages.add("Java");

languages.add("Python");

languages.add("C++");

System.out.println("Original LinkedList: " + languages);

// Insert element at a specific position (index 1)

languages.add(1, "JavaScript");

System.out.println("LinkedList after insertion: " + languages);

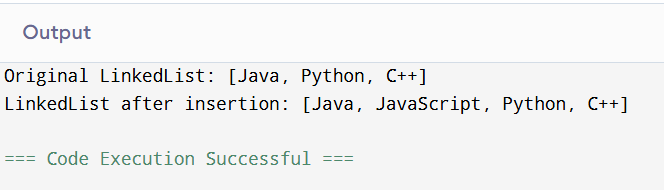
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment- 64**

**Aim:** **Write a Java program to insert elements at the first and last positions of a linked list.**

import java.util.LinkedList;

public class InsertFirstLast {

public static void main(String[] args) {

// Create and populate the LinkedList

LinkedList<String> fruits = new LinkedList<>();

fruits.add("Banana");

fruits.add("Mango");

fruits.add("Grapes");

System.out.println("Original LinkedList: " + fruits);

// Insert element at the first position

fruits.addFirst("Apple");

// Insert element at the last position

fruits.addLast("Orange");

System.out.println("LinkedList after inserting at first and last: " + fruits);

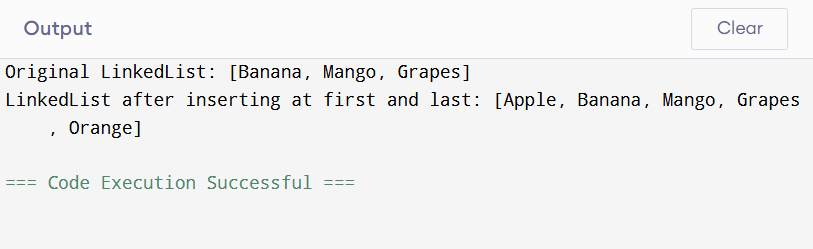
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 65**

**Aim: Write a Java program to add all elements from one TreeSet to another TreeSet.**

import java.util.TreeSet;

public class MergeTreeSets {

public static void main(String[] args) {

// First TreeSet with some elements

TreeSet<String> set1 = new TreeSet<>();

set1.add("Apple");

set1.add("Banana");

set1.add("Mango");

// Second TreeSet with additional elements

TreeSet<String> set2 = new TreeSet<>();

set2.add("Orange");

set2.add("Grapes");

System.out.println("TreeSet 1: " + set1);

System.out.println("TreeSet 2: " + set2);

// Add all elements of set1 to set2

set2.addAll(set1);

System.out.println("TreeSet 2 after adding all elements from TreeSet 1: " + set2);

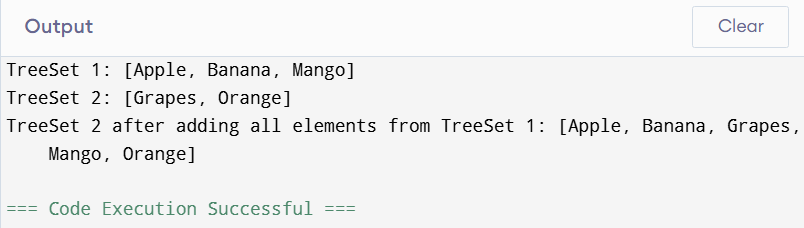
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment -66**

**Aim:** **Write a Java program to display the elements of a TreeSet in reverse order.**

import java.util.TreeSet;

public class ReverseTreeSet {

public static void main(String[] args) {

// Create and populate TreeSet

TreeSet<String> fruits = new TreeSet<>();

fruits.add("Apple");

fruits.add("Mango");

fruits.add("Banana");

fruits.add("Orange");

// Display original TreeSet

System.out.println("Original TreeSet (ascending order): " + fruits);

// Display in reverse order

System.out.println("TreeSet in reverse order: " + fruits.descendingSet());

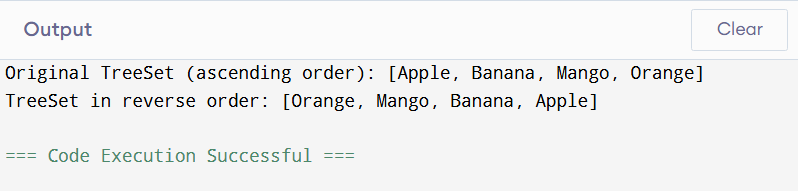
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 67**

**Aim:** **Write a Java program to retrieve the first and last elements from a TreeSet.**

import java.util.TreeSet;

public class TreeSetFirstLast {

public static void main(String[] args) {

// Create and populate TreeSet

TreeSet<String> colors = new TreeSet<>();

colors.add("Red");

colors.add("Blue");

colors.add("Green");

colors.add("Yellow");

// Display TreeSet

System.out.println("TreeSet: " + colors);

// Retrieve and display the first and last elements

String first = colors.first();

String last = colors.last();

System.out.println("First element: " + first);

System.out.println("Last element: " + last);

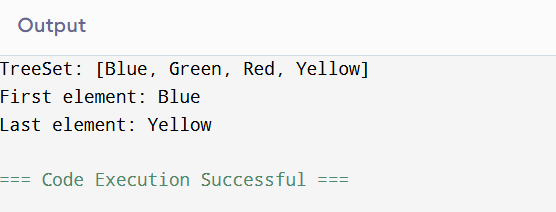
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment- 68**

**Aim: Write a Java program to clone a TreeSet into another TreeSet**

import java.util.TreeSet;

public class CloneTreeSetNumbers {

public static void main(String[] args) {

// Create and populate original TreeSet with numbers

TreeSet<Integer> originalSet = new TreeSet<>();

originalSet.add(10);

originalSet.add(50);

originalSet.add(20);

originalSet.add(70);

originalSet.add(30);

// Display original TreeSet

System.out.println("Original TreeSet: " + originalSet);

// Clone the TreeSet

@SuppressWarnings("unchecked")

TreeSet<Integer> clonedSet = (TreeSet<Integer>) originalSet.clone();

// Display cloned TreeSet

System.out.println("Cloned TreeSet: " + clonedSet);

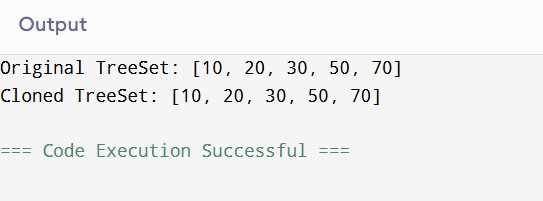
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 69**

**Aim:** **Write a Java program to count the number of elements in a TreeSet.**

import java.util.TreeSet;

public class TreeSetCount {

public static void main(String[] args) {

// Create and populate TreeSet with numbers

TreeSet<Integer> numbers = new TreeSet<>();

numbers.add(10);

numbers.add(20);

numbers.add(30);

numbers.add(40);

numbers.add(50);

// Display TreeSet

System.out.println("TreeSet Elements: " + numbers);

// Count elements using size()

int count = numbers.size();

System.out.println("Total number of elements in TreeSet: " + count);

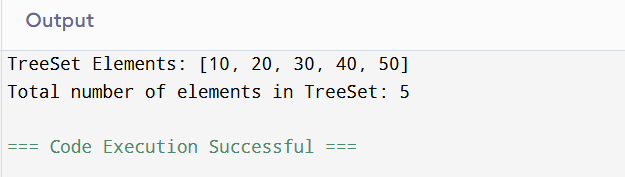
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 70**

**Aim: Write a Java program to compare two TreeSets.**

import java.util.TreeSet;

public class CompareTreeSets {

public static void main(String[] args) {

// First TreeSet

TreeSet<Integer> set1 = new TreeSet<>();

set1.add(10);

set1.add(20);

set1.add(30);

set1.add(40);

// Second TreeSet

TreeSet<Integer> set2 = new TreeSet<>();

set2.add(30);

set2.add(40);

set2.add(10);

set2.add(20);

// Compare sets for equality

boolean isEqual = set1.equals(set2);

System.out.println("Are both TreeSets equal? " + isEqual);

// Find common elements

TreeSet<Integer> common = new TreeSet<>(set1);

common.retainAll(set2);

System.out.println("Common elements: " + common);

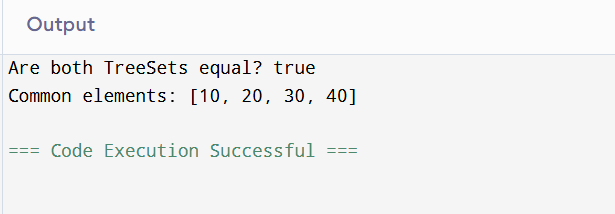
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 71**

**Aim:** **Write a Java program to clone one HashSet into another.**

import java.util.HashSet;

public class CloneNumberHashSet {

public static void main(String[] args) {

// 1. Create and populate the original HashSet with numbers

HashSet<Integer> originalSet = new HashSet<>();

originalSet.add(10);

originalSet.add(20);

originalSet.add(30);

originalSet.add(40);

originalSet.add(50);

// 2. Clone the original HashSet into another

@SuppressWarnings("unchecked")

HashSet<Integer> clonedSet = (HashSet<Integer>) originalSet.clone();

// 3. Display both sets

System.out.println("Original HashSet: " + originalSet);

System.out.println("Cloned HashSet: " + clonedSet);

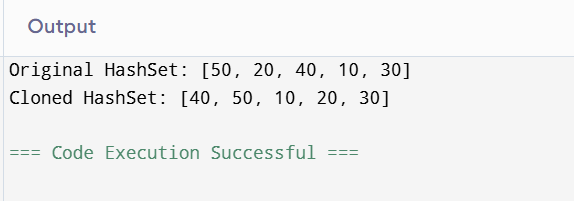
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 72**

**Aim: Write a Java program to convert a HashSet into an array.**

import java.util.HashSet;

public class HashSetToArray {

public static void main(String[] args) {

// 1. Create and populate the HashSet

HashSet<Integer> numberSet = new HashSet<>();

numberSet.add(100);

numberSet.add(200);

numberSet.add(300);

numberSet.add(400);

// 2. Convert HashSet to an array

Integer[] numberArray = new Integer[numberSet.size()];

numberSet.toArray(numberArray);

// 3. Print the array elements

System.out.println("Elements in the array:");

for (int num : numberArray) {

System.out.println(num);

}

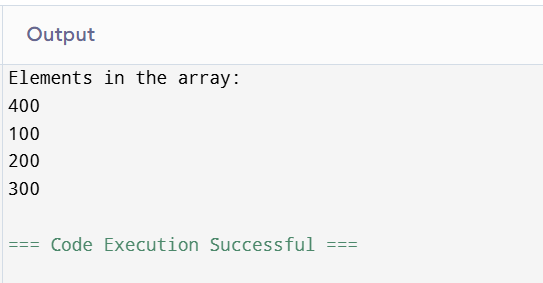
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment- 73**

**Aim:** **Write a Java program to convert a HashSet into a TreeSet.**

import java.util.HashSet;

import java.util.TreeSet;

public class HashSetToTreeSet {

public static void main(String[] args) {

// 1. Create and populate the HashSet

HashSet<Integer> numberSet = new HashSet<>();

numberSet.add(50);

numberSet.add(20);

numberSet.add(70);

numberSet.add(10);

numberSet.add(40);

// 2. Convert HashSet to TreeSet

TreeSet<Integer> sortedSet = new TreeSet<>(numberSet);

// 3. Display both sets

System.out.println("Original HashSet: " + numberSet);

System.out.println("Converted TreeSet (Sorted): " + sortedSet);

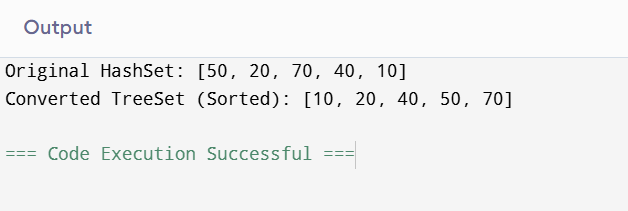
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment -74**

**Aim:** **Write a Java program to find numbers less than 7 in a TreeSet.**

import java.util.TreeSet;

public class TreeSetLessThanSeven {

public static void main(String[] args) {

// 1. Create and populate the TreeSet

TreeSet<Integer> numberSet = new TreeSet<>();

numberSet.add(1);

numberSet.add(3);

numberSet.add(5);

numberSet.add(7);

numberSet.add(9);

numberSet.add(11);

// 2. Find numbers less than 7

TreeSet<Integer> lessThanSeven = new TreeSet<>(numberSet.headSet(7));

// 3. Display the result

System.out.println("Original TreeSet: " + numberSet);

System.out.println("Numbers less than 7: " + lessThanSeven);

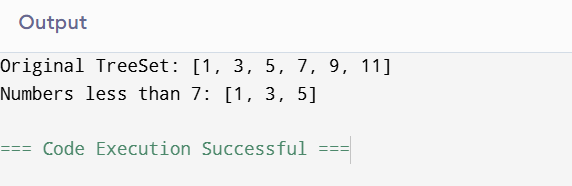
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 75**

**Aim: Write a Java program to compare two HashSets.**

import java.util.HashSet;

public class CompareHashSets {

public static void main(String[] args) {

// 1. Create and populate the first HashSet

HashSet<Integer> set1 = new HashSet<>();

set1.add(1);

set1.add(2);

set1.add(3);

set1.add(4);

// 2. Create and populate the second HashSet

HashSet<Integer> set2 = new HashSet<>();

set2.add(3);

set2.add(4);

set2.add(5);

set2.add(6);

// 3. Compare sets to find common elements

HashSet<Integer> commonElements = new HashSet<>(set1);

commonElements.retainAll(set2); // intersection

// 4. Display sets and comparison result

System.out.println("HashSet 1: " + set1);

System.out.println("HashSet 2: " + set2);

System.out.println("Common elements: " + commonElements);

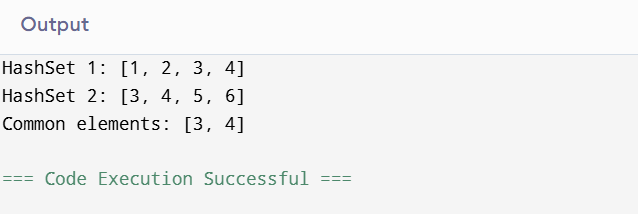
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 76**

**Aim: Write a Java program to retain common elements from two sets.**

import java.util.HashSet;

public class RetainCommonElements {

public static void main(String[] args) {

// 1. Create and populate the first set

HashSet<String> set1 = new HashSet<>();

set1.add("Apple");

set1.add("Banana");

set1.add("Cherry");

set1.add("Date");

// 2. Create and populate the second set

HashSet<String> set2 = new HashSet<>();

set2.add("Banana");

set2.add("Cherry");

set2.add("Elderberry");

set2.add("Fig");

// 3. Retain only the common elements in set1

set1.retainAll(set2);

// 4. Display the result

System.out.println("Common elements: " + set1);

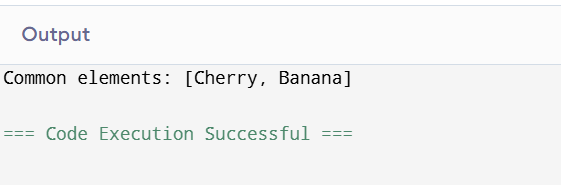
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 77**

**Aim: Write a Java program to remove all elements from a HashSet.**

import java.util.HashSet;

public class RemoveAllFromHashSet {

public static void main(String[] args) {

// 1. Create and populate the HashSet with numbers

HashSet<Integer> numberSet = new HashSet<>();

numberSet.add(10);

numberSet.add(20);

numberSet.add(30);

numberSet.add(40);

// 2. Display the original HashSet

System.out.println("Original HashSet: " + numberSet);

// 3. Remove all elements

numberSet.clear();

// 4. Display the HashSet after removal

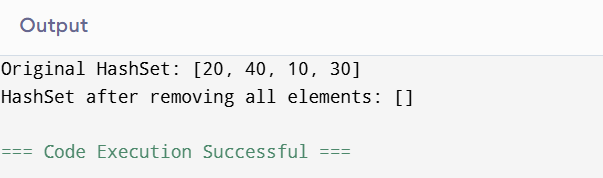
System.out.println("HashSet after removing all elements: " + numberSet);

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 78**

**Aim: Write a Java program to copy all mappings from one map to another.**

import java.util.HashMap;

import java.util.Map;

public class CopyMapExample {

public static void main(String[] args) {

// 1. Create and populate the first map

Map<Integer, String> map1 = new HashMap<>();

map1.put(1, "Apple");

map1.put(2, "Banana");

map1.put(3, "Cherry");

// 2. Create a second empty map

Map<Integer, String> map2 = new HashMap<>();

// 3. Copy all mappings from map1 to map2

map2.putAll(map1);

// 4. Display both maps

System.out.println("Original Map (map1): " + map1);

System.out.println("Copied Map (map2): " + map2);

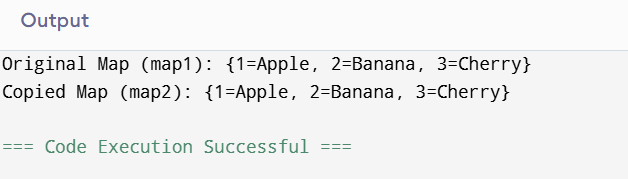
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment -79**

**Aim: Write a Java program to remove all key-value pairs from a map.**

import java.util.HashMap;

import java.util.Map;

public class ClearMapExample {

public static void main(String[] args) {

// 1. Create and populate the map

Map<Integer, String> fruitMap = new HashMap<>();

fruitMap.put(1, "Apple");

fruitMap.put(2, "Banana");

fruitMap.put(3, "Cherry");

// 2. Display the original map

System.out.println("Original Map: " + fruitMap);

// 3. Remove all key-value pairs

fruitMap.clear();

// 4. Display the map after clearing

System.out.println("Map after removing all key-value pairs: " + fruitMap);

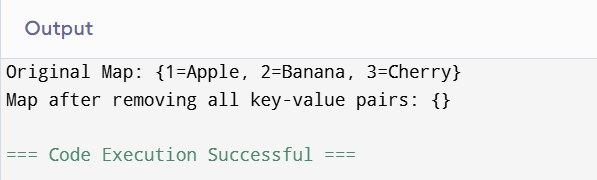
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**

****

**Experiment – 80**

**Aim:** **Write a Java program to check if a map is empty or contains key-value mappings.**

import java.util.HashMap;

import java.util.Map;

public class CheckMapEmpty {

public static void main(String[] args) {

// 1. Create a map

Map<Integer, String> myMap = new HashMap<>();

// 2. Check if the map is empty

if (myMap.isEmpty()) {

System.out.println("The map is currently empty.");

} else {

System.out.println("The map contains key-value pairs: " + myMap);

}

// 3. Add some entries

myMap.put(1, "Java");

myMap.put(2, "Python");

// 4. Check again

if (myMap.isEmpty()) {

System.out.println("The map is empty.");

} else {

System.out.println("Now the map contains: " + myMap);

}

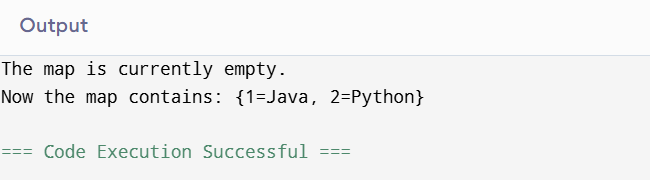
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 81**

**Aim:** **Write a Java program to create a shallow copy of a HashMap instance.**

import java.util.HashMap;

public class ShallowCopyHashMap {

public static void main(String[] args) {

// 1. Create and populate the original HashMap

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

originalMap.put(1, "Red");

originalMap.put(2, "Green");

originalMap.put(3, "Blue");

// 2. Create a shallow copy using clone()

@SuppressWarnings("unchecked")

HashMap<Integer, String> copiedMap = (HashMap<Integer, String>) originalMap.clone();

// 3. Display both maps

System.out.println("Original HashMap: " + originalMap);

System.out.println("Shallow Copied HashMap: " + copiedMap);

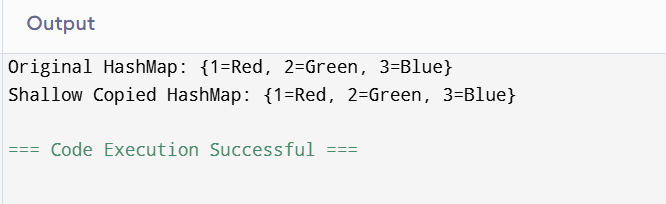
}

}

**Output:**

**Name: Vikki Chandrawanshi**

**Roll No.: 0873CS243D06**



**Experiment – 82**

**Aim: 82. Write a Java program to test whether a specified key exists in the map.**

import java.util.HashMap;

import java.util.Map;

public class KeyExistsInMap {

public static void main(String[] args) {

// 1. Create and populate the map

Map<Integer, String> sampleMap = new HashMap<>();

sampleMap.put(101, "Apple");

sampleMap.put(102, "Banana");

sampleMap.put(103, "Cherry");

// 2. Specify the key to check

int keyToCheck = 102;

// 3. Check if the key exists

if (sampleMap.containsKey(keyToCheck)) {

System.out.println("Key " + keyToCheck + " exists in the map with value: " + sampleMap.get(keyToCheck));

} else {

System.out.println("Key " + keyToCheck + " does not exist in the map.");

}

}

}

**Experiment – 83**

**Aim 83. ) Create a table Item\_dtls (Electronics) Try to insert at least 10 records in the above table Try to insert at least 2 records with null value.**

-- Create the Item\_dtls table for Electronics

CREATE TABLE Item\_dtls (

item\_id INT PRIMARY KEY,

item\_name VARCHAR(100) NOT NULL,

category VARCHAR(50) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

stock\_quantity INT NOT NULL,

manufacturer VARCHAR(100), -- Can be NULL

release\_date DATE -- Can be NULL

);

-- Insert at least 10 records into Item\_dtls

-- Record 1

INSERT INTO Item\_dtls (item\_id, item\_name, category, price, stock\_quantity, manufacturer, release\_date)

VALUES (101, 'Smartphone X', 'Mobile', 799.99, 150, 'TechCorp', '2023-01-15');

-- Record 2

INSERT INTO Item\_dtls (item\_id, item\_name, category, price, stock\_quantity, manufacturer, release\_date)

VALUES (102, 'Laptop Pro 15', 'Laptop', 1200.00, 80, 'GlobalTech', '2022-11-20');

-- Record 3

INSERT INTO Item\_dtls (item\_id, item\_name, category, price, stock\_quantity, manufacturer, release\_date)

VALUES (103, 'Wireless Headphones', 'Audio', 150.50, 300, 'SoundWave', '2023-03-01');

-- Record 4

INSERT INTO Item\_dtls (item\_id, item\_name, category, price, stock\_quantity, manufacturer, release\_date)

VALUES (104, 'Smart TV 55"', 'Television', 950.00, 60, 'Visionary', '2023-02-10');

-- Record 5

INSERT INTO Item\_dtls (item\_id, item\_name, category, price, stock\_quantity, manufacturer, release\_date)

VALUES (105, 'Gaming Console Z', 'Gaming', 499.00, 100, 'PlayGen', '2022-10-05');

-- Record 6 (with NULL manufacturer)

INSERT INTO Item\_dtls (item\_id, item\_name, category, price, stock\_quantity, manufacturer, release\_date)

VALUES (106, 'Portable Bluetooth Speaker', 'Audio', 75.00, 250, NULL, '2023-04-22');

-- Record 7 (with NULL release\_date)

INSERT INTO Item\_dtls (item\_id, item\_name, category, price, stock\_quantity, manufacturer, release\_date)

VALUES (107, 'Smartwatch Series 5', 'Wearable', 299.99, 120, 'TimeGear', NULL);

-- Record 8

INSERT INTO Item\_dtls (item\_id, item\_name, category, price, stock\_quantity, manufacturer, release\_date)

VALUES (108, 'Tablet Mini 8', 'Tablet', 350.00, 90, 'SlimTabs', '2023-01-01');

-- Record 9

INSERT INTO Item\_dtls (item\_id, item\_name, category, price, stock\_quantity, manufacturer, release\_date)

VALUES (109, 'Drone Explorer', 'Drone', 850.75, 40, 'SkyHigh', '2022-12-18');

-- Record 10

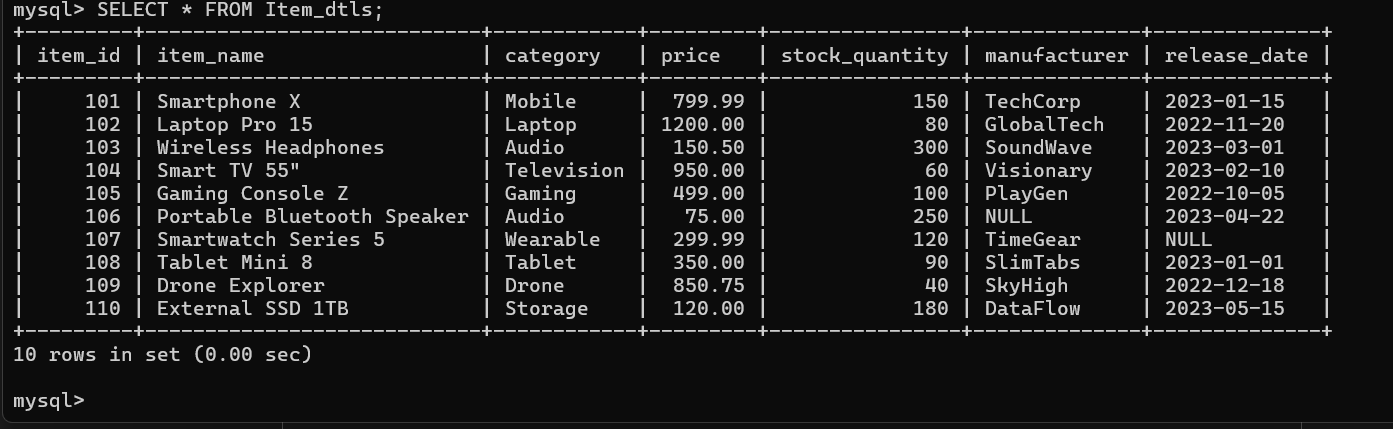
INSERT INTO Item\_dtls (item\_id, item\_name, category, price, stock\_quantity, manufacturer, release\_date)

VALUES (110, 'External SSD 1TB', 'Storage', 120.00, 180, 'DataFlow', '2023-05-15');

**OUTPUT :-**

Name :- Vikki Chandrawanshi

Roll No:- 0873CS243D06



**Experiment – 84**

**Aim 84). Create a table Sales\_dtls Try to insert at least 10 records in the above table Try to insert at least 2 records with null value.**

CREATE TABLE Sales\_dtls (

SaleID INT PRIMARY KEY,

ProductName VARCHAR(50),

Quantity INT,

SaleDate DATE,

Price DECIMAL(10, 2)

);

INSERT INTO Sales\_dtls (SaleID, ProductName, Quantity, SaleDate, Price) VALUES

(1, 'Laptop', 5, '2025-06-01', 1200.00),

(2, 'Smartphone', 10, '2025-06-02', 600.00),

(3, 'Tablet', 7, '2025-06-03', 300.00),

(4, 'Monitor', 3, '2025-06-04', 200.00),

(5, 'Keyboard', 15, '2025-06-05', 50.00),

(6, 'Mouse', 20, '2025-06-06', 25.00),

(7, NULL, 8, '2025-06-07', 100.00), -- NULL ProductName

(8, 'Printer', NULL, '2025-06-08', 150.00), -- NULL Quantity

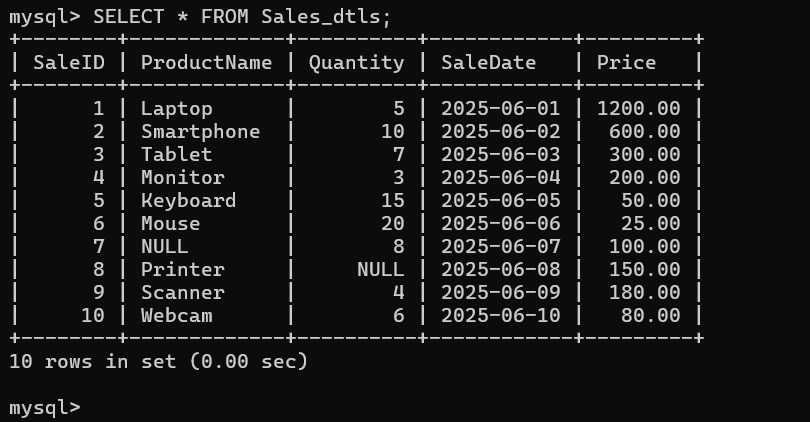
(9, 'Scanner', 4, '2025-06-09', 180.00),

(10, 'Webcam', 6, '2025-06-10', 80.00);

**OUTPUT:-**

Name :- Vikki Chandrawanshi

Roll No:- 0873CS243D06



**Experiment – 85**

**Aim 85)create a table manufacturers Try to insert at least 10 records in the above table Try to insert at least 2 records with null value.**

CREATE TABLE CUST\_DTLS (

CustID INT PRIMARY KEY,

CustName VARCHAR(100),

CustEmail VARCHAR(100),

CustPhone VARCHAR(15),

CustAddress VARCHAR(255)

);

INSERT INTO CUST\_DTLS VALUES

(1, 'John Doe', 'john@example.com', '1234567890', '123 Elm Street'),

(2, 'Jane Smith', 'jane@example.com', '2345678901', '456 Oak Avenue'),

(3, 'Mike Johnson', NULL, '3456789012', '789 Pine Road'),

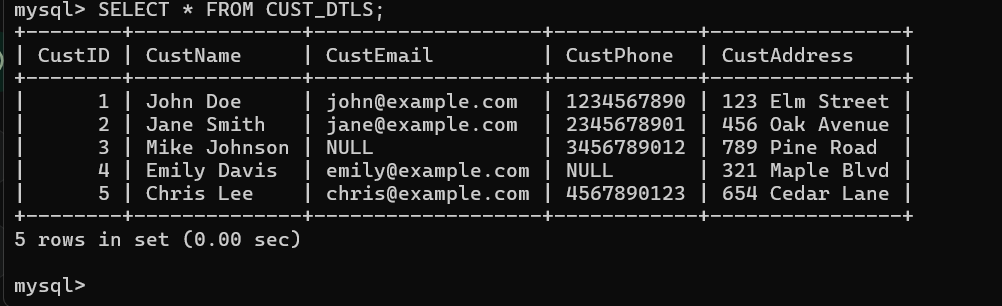
(4, 'Emily Davis', 'emily@example.com', NULL, '321 Maple Blvd'),

(5, 'Chris Lee', 'chris@example.com', '4567890123', '654 Cedar Lane');

**OUTPUT:-**

Name :- Vikki Chandrawanshi

Roll No:- 0873CS243D06



**Consider the below tables with estimated columns and then practise below questions.**

**CUST DTLS**

**CUST Act DTLS**

**ACT\_TYPES\_INFO**

**PROD\_DTLS**

**EMP**

**DEPT**

CODE:- -- Customer Details Table

CREATE TABLE CUST\_DTLS (

CustID INT PRIMARY KEY,

CustName VARCHAR(100),

Mobile VARCHAR(15),

City VARCHAR(50)

);

-- Account Type Info Table

CREATE TABLE ACT\_TYPES\_INFO (

ActTypeID INT PRIMARY KEY,

ActTypeName VARCHAR(50),

InterestRate DECIMAL(5,2)

);

-- Customer Account Details Table

CREATE TABLE CUST\_ACT\_DTLS (

ActID INT PRIMARY KEY,

CustID INT,

ActTypeID INT,

Balance DECIMAL(10,2),

OpenDate DATE,

FOREIGN KEY (CustID) REFERENCES CUST\_DTLS(CustID),

FOREIGN KEY (ActTypeID) REFERENCES ACT\_TYPES\_INFO(ActTypeID)

);

-- Product Details Table

CREATE TABLE PROD\_DTLS (

ProdID INT PRIMARY KEY,

ProdName VARCHAR(100),

ManufacturerID INT,

Price DECIMAL(10,2),

ManufactureDate DATE,

ExpiryDate DATE

);

-- Department Table

CREATE TABLE DEPT (

DeptID INT PRIMARY KEY,

DeptName VARCHAR(50),

Location VARCHAR(50)

);

-- Employee Table

CREATE TABLE EMP (

EmpID INT PRIMARY KEY,

EmpName VARCHAR(100),

DeptID INT,

JobTitle VARCHAR(50),

Salary DECIMAL(10,2),

FOREIGN KEY (DeptID) REFERENCES DEPT(DeptID)

);

-- Customers

INSERT INTO CUST\_DTLS VALUES

(1, 'John Doe', '9876543210', 'Texas'),

(2, 'Jane Smith', '9123456780', 'New York'),

(3, 'Sam Wilson', '9988776655', 'Texas');

-- Account Types

INSERT INTO ACT\_TYPES\_INFO VALUES

(101, 'SAVINGS', 3.5),

(102, 'SALARY', 4.0),

(103, 'TRADING', 5.0);

-- Customer Accounts

INSERT INTO CUST\_ACT\_DTLS VALUES

(201, 1, 101, 1000.00, '2023-02-10'),

(202, 2, 102, 5000.00, '2024-12-01'),

(203, 3, 103, 2000.00, '2022-05-01');

-- Products

INSERT INTO PROD\_DTLS VALUES

(301, 'Laptop', 1, 45000.00, '2025-01-15', '2027-01-15'),

(302, 'Mouse', 2, 500.00, '2024-01-01', '2025-01-01'),

(303, 'Keyboard', 2, 800.00, '2022-06-20', '2023-06-20');

-- Departments

INSERT INTO DEPT VALUES

(10, 'Sales', 'CHICAGO'),

(20, 'IT', 'NEW YORK');

-- Employees

INSERT INTO EMP VALUES

(401, 'Ravi Kumar', 10, 'Clerk', 25000.00),

(402, 'Anjali Mehra', 20, 'Manager', 60000.00),

(403, 'Vikash Sharma', 10, 'Clerk', 24000.00);

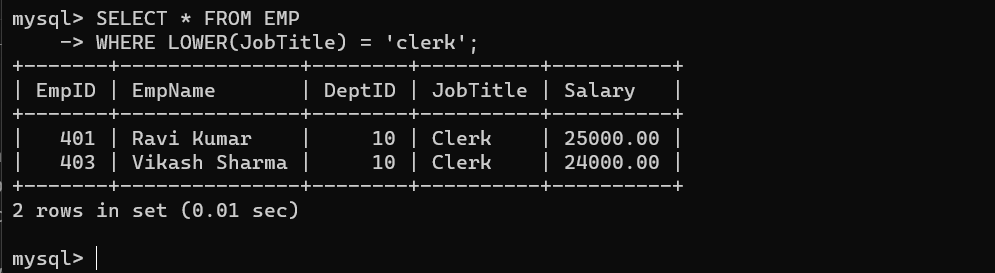
**Experiment – 86**

**Aim 86) Fetch all clerks information?**

**OUTPUT:-**

Name :- Vikki Chandrawanshi

Roll No:- 0873CS243D06

****

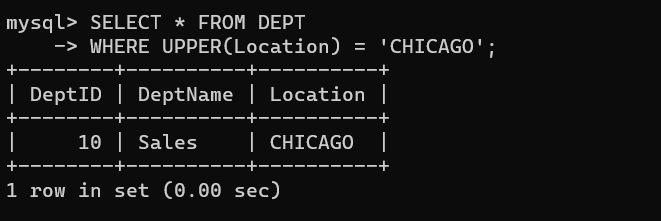
**Experiment – 87**

**Aim 87) Display all departments information located at CHICAGO?**

**OUTPUT:-**

Name :- Vikki Chandrawanshi

Roll No:- 0873CS243D06

****

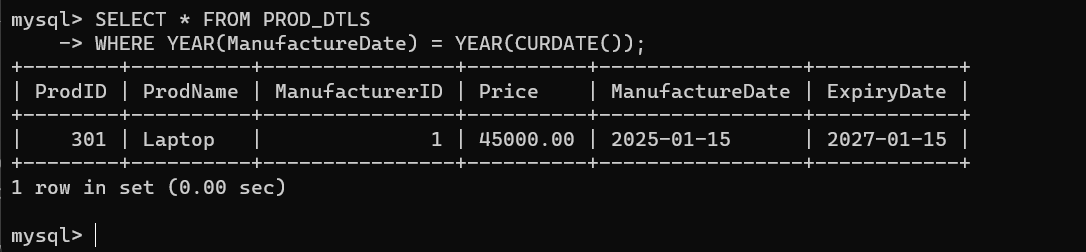
**Experiment – 88**

**Aim 88) Display product details manufactured in the current year only?**

**OUTPUT:-**

Name :- Vikki Chandrawanshi

Roll No:- 0873CS243D06

****

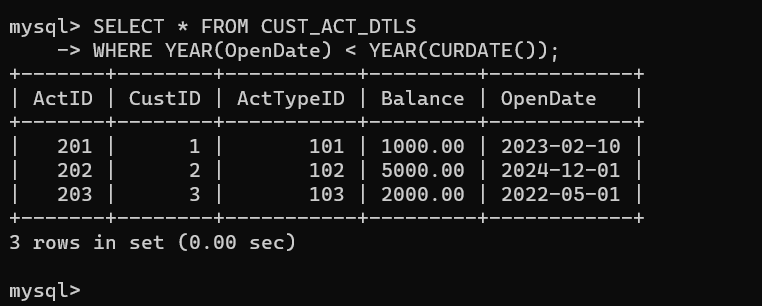
**Experiment – 89**

**Aim 89) Get the details of cutomers accounts who opened the accounts before this year?**

**OUTPUT:-**

Name :- Vikki Chandrawanshi

Roll No:- 0873CS243D06

****

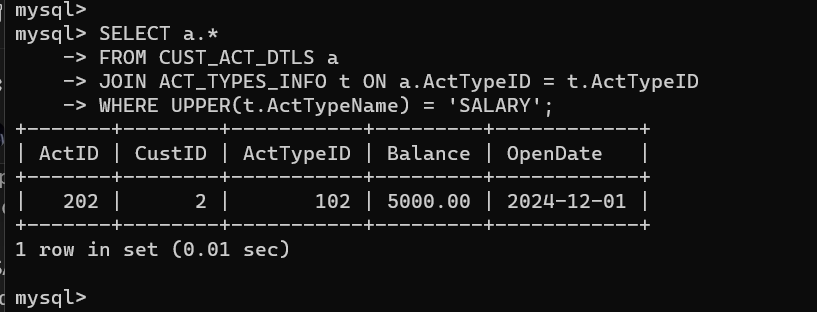
**Experiment – 90**

**Aim 90) Get all SALARY account details?**

**OUTPUT:-**

Name :- Vikki Chandrawanshi

Roll No:- 0873CS243D06

****

**Experiment – 91**

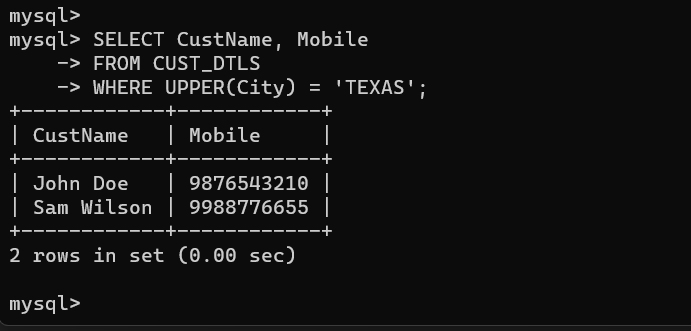
**Aim 91) Display customer names and mobile numbers from the city 'Texas'?**

**select cname, mobile from cust\_dtls where city='Texas';**

**OUTPUT:-**

Name :- Vikki Chandrawanshi

Roll No:- 0873CS243D06

****

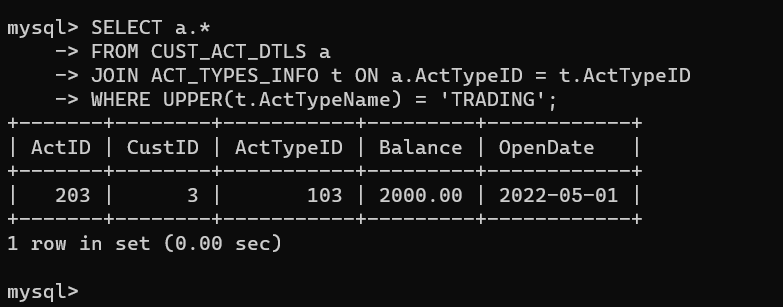
**Experiment – 92**

**Aim 92) Get the information of Trading account?**

**OUTPUT:-**

Name :- Vikki Chandrawanshi

Roll No:- 0873CS243D06

****

**Experiment – 93**

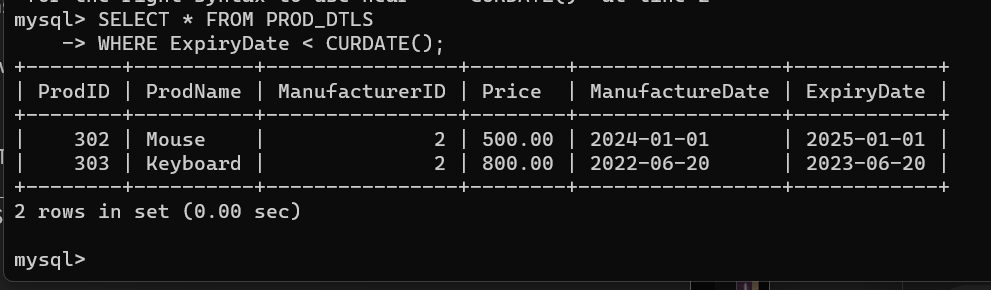
**Aim 93) Display only Expired product details?**

**select \* from prod\_dtls where exp<sysdate;**

**OUTPUT:-**

Name :- Vikki Chandrawanshi

Roll No:- 0873CS243D06

****