1.     Write the programme to sort the integers 8, 4, 3,5,6 and the alphabetical string C, O, I, P, U, in ascending order. Show the resulting output.

Program :

package Lab\_6;

import java.util.Arrays;

public class Q1\_SortExample {

public static void main(String[] args) {

// Define the arrays

int[] integersArray = { 8, 4, 3, 5, 6 };

String[] alphabeticalStringsArray = { "C", "O", "I", "P", "U" };

// Sort the arrays

Arrays.*sort*(integersArray);

Arrays.*sort*(alphabeticalStringsArray);

// Print the sorted arrays

System.***out***.println("Sorted integers: " + Arrays.*toString*(integersArray));

System.***out***.println("Sorted alphabetical strings: " + Arrays.*toString*(alphabeticalStringsArray));

}

}

Output :

A screenshot of a computer

Description automatically generated

2.     Write a Java program to implement the bubble sort algorithm to sort an array of integers in ascending order.

Program :

package Lab\_6;

public class Q2\_BubbleSort {

public static void main(String[] args) {

// Define an array of integers to be sorted

int[] array = {64, 34, 25, 12, 22, 11, 90};

// Print the original array

System.***out***.println("Original array:");

*printArray*(array);

// Perform Bubble Sort

*bubbleSort*(array);

// Print the sorted array

System.***out***.println("Sorted array:");

*printArray*(array);

}

// Method to perform Bubble Sort

public static void bubbleSort(int[] arr) {

int n = arr.length;

boolean swapped;

for (int i = 0; i < n - 1; i++) {

swapped = false;

// Last i elements are already in place

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

// Swap if the element found is greater than the next element

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

swapped = true;

}

}

// If no elements were swapped, the array is already sorted

if (!swapped) break;

}

}

// Method to print the elements of the array

public static void printArray(int[] arr) {

for (int num : arr) {

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

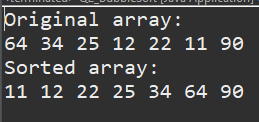
}

System.***out***.println();

}

}

Output :



3.     Write a program to input an array 10 elements and print the cube of prime numbers in it.

Program :

package Lab\_6;

import java.util.Scanner;

public class Q3\_CubeOfPrimeNumbers {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

// Define the array and read input

int[] array = new int[10];

System.***out***.println("Enter 10 integers:");

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

array[i] = scanner.nextInt();

}

// Close the scanner

scanner.close();

// Print cubes of prime numbers

System.***out***.println("Cubes of prime numbers in the array:");

for (int num : array) {

if (*isPrime*(num)) {

System.***out***.println("Number: " + num + ", Cube: " + (num \* num \* num));

}

}

}

// Method to check if a number is prime

public static boolean isPrime(int num) {

if (num <= 1) {

return false;

}

for (int i = 2; i \* i <= num; i += 6) {

if (num % i == 0 || num % (i + 2) == 0) {

return false;

}

}

return true;

}

}

Output :

A screenshot of a computer

Description automatically generated

4.     Write a java program to implement integer wrapper class methods.(any 3 methods)

Program :

package Lab\_6;

public class Q4\_IntegerWrapperExample {

public static void main(String[] args) {

// Method 1: parseInt(String s)

String numberString = "1234";

int number = Integer.*parseInt*(numberString);

System.***out***.println("Parsed integer from string \"" + numberString + "\": " + number);

// Method 2: toString(int i)

int intValue = 5678;

String intString = Integer.*toString*(intValue);

System.***out***.println("String representation of integer " + intValue + ": " + intString);

// Method 3: compare(int x, int y)

int x = 42;

int y = 27;

int comparisonResult = Integer.*compare*(x, y);

if (comparisonResult < 0) {

System.***out***.println(x + " is less than " + y);

} else if (comparisonResult > 0) {

System.***out***.println(x + " is greater than " + y);

} else {

System.***out***.println(x + " is equal to " + y);

}

}

}

Output :

A screenshot of a computer

Description automatically generated

5.     Write a java program to implement double wrapper class methods.(any 3 methods)

Program :

package Lab\_6;

public class Q5\_DoubleWrapper {

public static void main(String[] args) {

// Method 1: parseDouble(String s)

String doubleString = "123.45";

double parsedDouble = Double.*parseDouble*(doubleString);

System.***out***.println("Parsed double from string \"" + doubleString + "\": " + parsedDouble);

// Method 2: toString(double d)

double doubleValue = 678.90;

String doubleStringFromDouble = Double.*toString*(doubleValue);

System.***out***.println("String representation of double " + doubleValue + ": " + doubleStringFromDouble);

// Method 3: compare(double d1, double d2)

double d1 = 3.14;

double d2 = 2.71;

int comparisonResult = Double.*compare*(d1, d2);

if (comparisonResult < 0) {

System.***out***.println(d1 + " is less than " + d2);

} else if (comparisonResult > 0) {

System.***out***.println(d1 + " is greater than " + d2);

} else {

System.***out***.println(d1 + " is equal to " + d2);

}

}

}

Output :

A screenshot of a computer

Description automatically generated

6.     Write a java program to implement float wrapper class methods.(any 3 methods)

Program :

package Lab\_6;

public class Q6\_FloatWrapper {

public static void main(String[] args) {

// Method 1: parseFloat(String s)

String floatString = "15.14";

float parsedFloat = Float.*parseFloat*(floatString);

System.***out***.println("Parsed float from string \"" + floatString + "\": " + parsedFloat);

// Method 2: toString(float f)

float floatValue = 48.45f;

String floatStringFromFloat = Float.*toString*(floatValue);

System.***out***.println("String representation of float " + floatValue + ": " + floatStringFromFloat);

// Method 3: compare(float f1, float f2)

float f1 = 1.23f;

float f2 = 2.35f;

int comparisonResult = Float.*compare*(f1, f2);

if (comparisonResult < 0) {

System.***out***.println(f1 + " is less than " + f2);

} else if (comparisonResult > 0) {

System.***out***.println(f1 + " is greater than " + f2);

} else {

System.***out***.println(f1 + " is equal to " + f2);

}

}

}

Output :

A screenshot of a computer

Description automatically generated

7.     Write a Java program to validate email addresses using regular expressions. The email should have the format username@domain.com where username and domain can contain alphanumeric characters, dots, and hyphens.

Program :

package Lab\_6;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class Q7\_EmailValidator {

// Regular expression for validating email addresses

// email id is valid if in format username@domain.com

// username and domain can contain alphanumeric characters, dots, and hyphens.

// and end should be .com , any other is not valid as question say.

private static final String ***EMAIL\_REGEX*** = "^[a-zA-Z0-9.-]+@[a-zA-Z0-9.-]+.com";

public static void main(String[] args) {

// Test email addresses

String[] testEmails = { "user.name@example.com", "user-name@example.co.uk", "username@domain",

"username@domain.c", "username@domain@com", "user..name@example.com", "user-name@domain.co.in",

"user@domain", "@domain.com", "user@domain.com", "user123@domain.com" };

// Validate each email address

for (String email : testEmails) {

boolean isValid = *validateEmail*(email);

System.***out***.println(email + " is " + (isValid ? "valid" : "invalid"));

}

}

// Method to validate email address using regular expression

public static boolean validateEmail(String email) {

Pattern pattern = Pattern.*compile*(***EMAIL\_REGEX***);

Matcher matcher = pattern.matcher(email);

return matcher.matches();

}

}

Output :

A screen shot of a computer

Description automatically generated

8.     Create a Java program to validate phone numbers. The format should be (xxx) xxx-xxxx where x is a digit.

Program :

package Lab\_6;

import java.util.regex.Pattern;

import java.util.regex.Matcher;

public class Q8\_PhoneNumberValidator {

// Regular expression for validating phone numbers in the format (xxx) xxx-xxxx

private static final String PHONE\_NUMBER\_REGEX = "\\(\\d{3}\\) \\d{3}-\\d{4}";

public static void main(String[] args) {

// Test phone numbers

String[] testPhoneNumbers = {

"(123) 456-7890",

"(987) 654-3210",

"(123)456-7890",

"(123) 4567-890",

"123-456-7890",

"(123) 456-78901",

"(12) 345-6789",

"(123) 456-789"

};

// Validate each phone number

for (String phoneNumber : testPhoneNumbers) {

boolean isValid = validatePhoneNumber(phoneNumber);

System.out.println(phoneNumber + " is " + (isValid ? "valid" : "invalid"));

}

}

// Method to validate phone number using regular expression

public static boolean validatePhoneNumber(String phoneNumber) {

Pattern pattern = Pattern.compile(PHONE\_NUMBER\_REGEX);

Matcher matcher = pattern.matcher(phoneNumber);

return matcher.matches();

}

}

Output :

A screen shot of a computer

Description automatically generated