LAB 6 ANP-C7781

Solve following questions:

1. Write a java 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 Day07;

import java.util.Arrays;

public class ArraySortingProgram {

public static void main(String[] args) {

// Declare and initialize an integer array

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

// Sort the integer array in ascending order

Arrays.*sort*(integers);

// Print the sorted integer array

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

// Declare and initialize a string array

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

// Sort the string array in ascending order

Arrays.*sort*(strings);

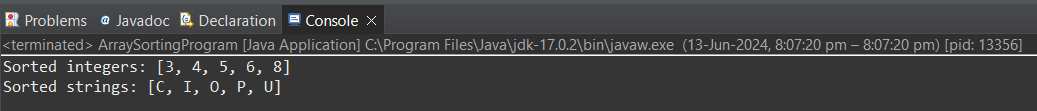
// Print the sorted string array

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

}

}

Output :-



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

Program :-

package Day07;

public class BubbleSortProgram {

public static void main(String[] args) {

// Define and initialize an array of integers

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

// Print the original array

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

*printArray*(arr);

// Call bubbleSort to sort the array

*bubbleSort*(arr);

// Print the sorted array

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

*printArray*(arr);

}

// Function to implement bubble sort

public static void bubbleSort(int[] arr) {

int n = arr.length; // Get the length of the array

// Outer loop for passes

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

// Inner loop for comparison and swapping

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

// Compare and swap if the current element is greater than the next element

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

// Swap arr[j] and arr[j + 1]

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

}

// Function to print the array

public static void printArray(int[] arr) {

// Iterate through the array and print each element

for (int i : arr) {

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

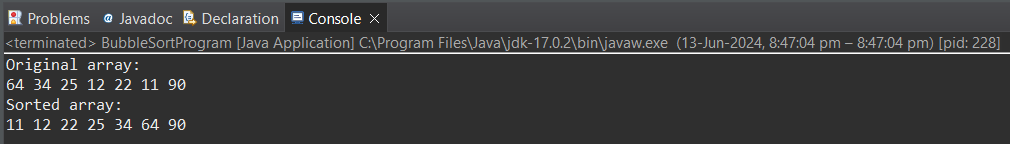
}

System.***out***.println(); // Print a new line at the end

}

}

Output :-



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

Program :-

package Day07;

import java.util.Scanner;

public class CubePrimeNumbers {

// Create a scanner object for input

public static void main(String[] args) {

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

int[] arr = new int[10];

// Input 10 elements from the user

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

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

arr[i] = scanner.nextInt();

}

// Print the cube of prime numbers

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

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

if (*isPrime*(arr[i])) {

System.***out***.println("Cube of " + arr[i] + " is " + (arr[i] \* arr[i] \* arr[i]));

}

}

// Close the scanner

scanner.close();

}

// Function to check if a number is prime

public static boolean isPrime(int num) {

// Numbers less than 2 are not prime

if (num < 2) {

return false;

}

// Check divisibility from 2 to the square root of the number

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

if (num % i == 0) {

return false;

}

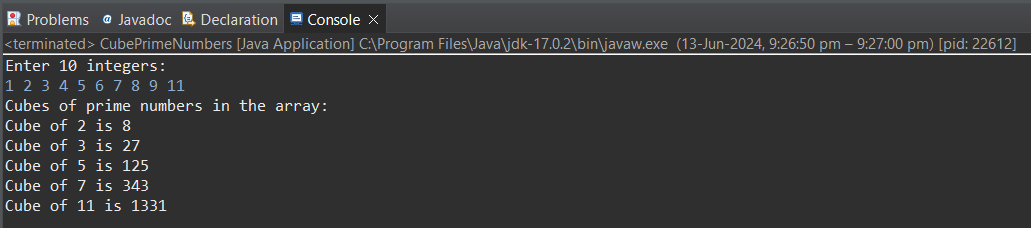
}

return true;

}

}

Output :-



1. Write a java program to implement integer wrapper class methods. (Any 5 methods)

Program :-

package Day07;

public class IntegerWrapperProgram {

public static void main(String[] args) {

// Parsing an int from a String

// The parseInt method converts a string representation of a number into an int.

String numberStr = "123";

int parsedNumber = Integer.*parseInt*(numberStr);

System.***out***.println("Parsed number from string: " + parsedNumber);

// Getting the maximum value of an int

// The MAX\_VALUE constant holds the maximum value an int can have, which is 2^31

// - 1.

int maxInt = Integer.***MAX\_VALUE***;

System.***out***.println("Maximum value of int: " + maxInt);

// Converting an int to a String

// The toString method converts an int to its string representation.

int number = 456;

String numberToStr = Integer.*toString*(number);

System.***out***.println("Integer converted to string: " + numberToStr);

// Comparing two integers

// The compare method compares two int values and returns:

// - a negative value if the first is less than the second,

// - zero if they are equal,

// - a positive value if the first is greater than the second.

Integer num1 = 10;

Integer num2 = 20;

int comparisonResult = Integer.*compare*(num1, num2);

if (comparisonResult < 0) {

System.***out***.println(num1 + " is less than " + num2);

} else if (comparisonResult > 0) {

System.***out***.println(num1 + " is greater than " + num2);

} else {

System.***out***.println(num1 + " is equal to " + num2);

}

// Decoding a String into an Integer

// The decode method decodes a string into an Integer. The string can be in

// decimal,

// hexadecimal (prefix 0x or 0X), or octal (prefix 0).

String hexStr = "0x1A";

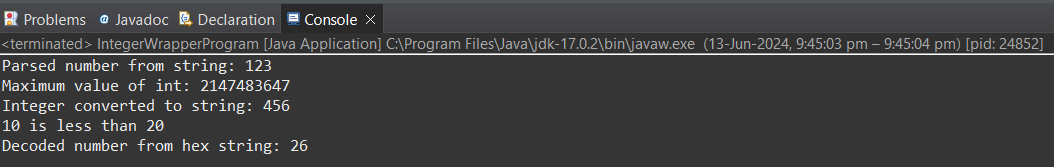
Integer decodedNumber = Integer.*decode*(hexStr);

System.***out***.println("Decoded number from hex string: " + decodedNumber);

}

}

Output :-



1. Write a java program to implement double wrapper class methods. (Any 5 methods)

Program :-

package Day07;

public class DoubleWrapperProgram {

public static void main(String[] args) {

// Parsing a double from a String

// The parseDouble method converts a string representation of a number into a

// double.

String numberStr = "123.45";

double parsedNumber = Double.*parseDouble*(numberStr);

System.***out***.println("Parsed number from string: " + parsedNumber);

// Getting the maximum value of a double

// The MAX\_VALUE constant holds the maximum value a double can have.

double maxDouble = Double.***MAX\_VALUE***;

System.***out***.println("Maximum value of double: " + maxDouble);

// Checking if a value is NaN (Not-a-Number)

// The isNaN method returns true if the specified number is NaN.

double nanValue = Double.***NaN***;

System.***out***.println("Is NaN value: " + Double.*isNaN*(nanValue));

// Converting a double to a String

// The toString method converts a double to its string representation.

double number = 456.78;

String numberToStr = Double.*toString*(number);

System.***out***.println("Double converted to string: " + numberToStr);

// Comparing two double values

// The compare method compares two double values and returns:

// - a negative value if the first is less than the second,

// - zero if they are equal,

// - a positive value if the first is greater than the second.

Double num1 = 10.5;

Double num2 = 20.5;

int comparisonResult = Double.*compare*(num1, num2);

if (comparisonResult < 0) {

System.***out***.println(num1 + " is less than " + num2);

} else if (comparisonResult > 0) {

System.***out***.println(num1 + " is greater than " + num2);

} else {

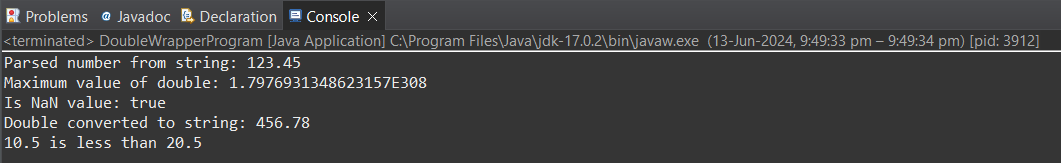
System.***out***.println(num1 + " is equal to " + num2);

}

}

}

Output :-



1. Write a java program to implement float wrapper class methods. (Any 5 methods)

Program :-

package Day07;

public class FloatWrapperClass {

public static void main(String[] args) {

// Parsing a float from a String

// The parseFloat method converts a string representation of a number into a

// float.

String numberStr = "123.45f";

float parsedNumber = Float.*parseFloat*(numberStr);

System.***out***.println("Parsed number from string: " + parsedNumber);

// Getting the maximum value of a float

// The MAX\_VALUE constant holds the maximum value a float can have.

float maxFloat = Float.***MAX\_VALUE***;

System.***out***.println("Maximum value of float: " + maxFloat);

// Checking if a value is NaN (Not-a-Number)

// The isNaN method returns true if the specified number is NaN.

float nanValue = Float.***NaN***;

System.***out***.println("Is NaN value: " + Float.*isNaN*(nanValue));

// Converting a float to a String

// The toString method converts a float to its string representation.

float number = 456.78f;

String numberToStr = Float.*toString*(number);

System.***out***.println("Float converted to string: " + numberToStr);

// Comparing two float values

// The compare method compares two float values and returns:

// - a negative value if the first is less than the second,

// - zero if they are equal,

// - a positive value if the first is greater than the second.

Float num1 = 10.5f;

Float num2 = 20.5f;

int comparisonResult = Float.*compare*(num1, num2);

if (comparisonResult < 0) {

System.***out***.println(num1 + " is less than " + num2);

} else if (comparisonResult > 0) {

System.***out***.println(num1 + " is greater than " + num2);

} else {

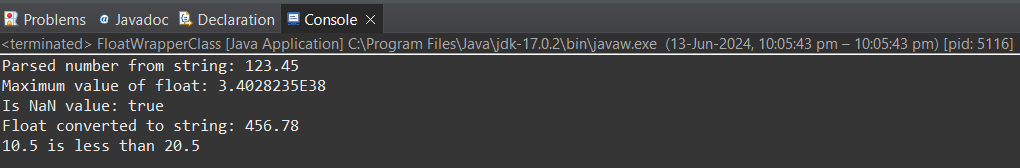
System.***out***.println(num1 + " is equal to " + num2);

}

}

}

Output :-



1. 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 Day07;

import java.util.regex.Pattern;

import java.util.regex.Matcher;

import java.util.Scanner;

public class RegexEmailValidator {

// Regular expression for validating email address

private static final String ***EMAIL\_REGEX*** = "^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,6}$";

public static void main(String[] args) {

// Create a scanner object for input

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

// Prompt the user to enter an email address

System.***out***.println("Enter an email address to validate:");

String email = scanner.nextLine();

// Validate the email address

boolean isValid = *validateEmail*(email);

// Print the validation result

if (isValid) {

System.***out***.println("The email address " + email + " is valid.");

} else {

System.***out***.println("The email address " + email + " is invalid.");

}

// Close the scanner

scanner.close();

}

// Method to validate the email address using regular expression

public static boolean validateEmail(String email) {

// Compile the regular expression

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

// Match the input email with the pattern

Matcher matcher = pattern.matcher(email);

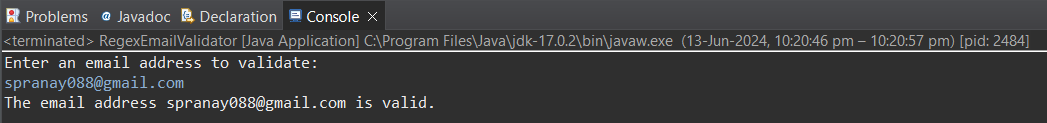
// Return whether the email matches the pattern

return matcher.matches();

}

}

Output :-



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

Program :-

package Day07;

import java.util.Scanner;

import java.util.regex.Pattern;

import java.util.regex.Matcher;

public class PhoneNumberValidator {

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

private static final String ***PHONE\_NUMBER\_REGEX*** = "^\\(?([0-9]{3})\\)?[-. ]?([0-9]{3})[-. ]?([0-9]{4})$";

public static void main(String[] args) {

// Create a scanner object for input

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

// Prompt the user to enter a phone number

System.***out***.println("Enter a phone number to validate :");

String phoneNumber = scanner.nextLine();

// Validate the phone number

boolean isValid = *validatePhoneNumber*(phoneNumber);

// Print the validation result

if (isValid) {

System.***out***.println("The phone number " + phoneNumber + " is valid.");

} else {

System.***out***.println("The phone number " + phoneNumber + " is invalid.");

}

// Close the scanner

scanner.close();

}

// Method to validate the phone number using regular expression

public static boolean validatePhoneNumber(String phoneNumber) {

// Compile the regular expression

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

// Match the input phone number with the pattern

Matcher matcher = pattern.matcher(phoneNumber);

// Return whether the phone number matches the pattern

return matcher.matches();

}

}

Output :-

