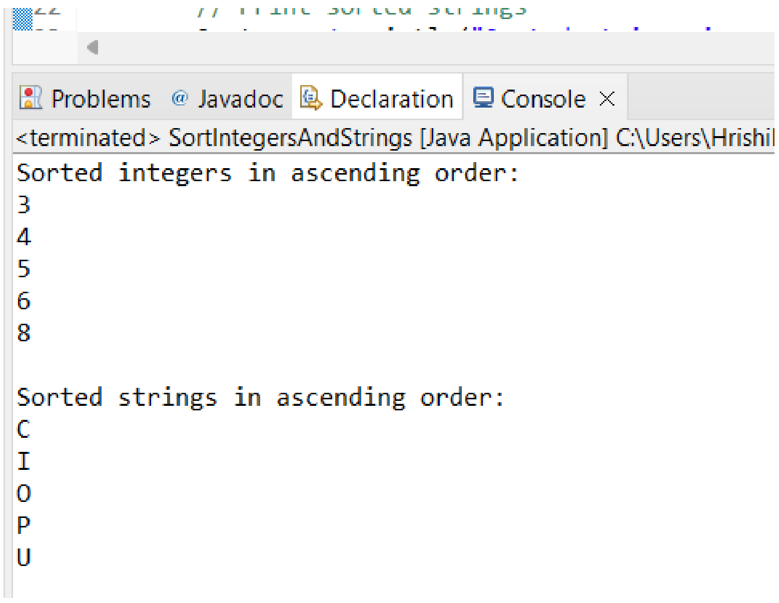
Poorvaja Bhorunde AF0403155

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 demo;
import java.util.Arrays;
public class SortIntegersAndStrings {

public static void main(String[] args) {
// TODO Auto-generated method stub
// Sort integers
int[] numbers = {8, 4, 3, 5, 6};
Arrays.sort(numbers);

// Sort strings (alphabetically)
String[] letters = {"C", "O", "I", "P", "U"};
Arrays.sort(letters);

// Print sorted integers
System.out.println("Sorted integers in ascending order: ");
for (int number : numbers) {
System.out.println(number + " ");
}
System.out.println();

// Print sorted strings
System.out.println("Sorted strings in ascending order: ");
for (String letter : letters) {
System.out.println(letter + " ");
}
}
}
```

Output :



```
<terminated> SortIntegersAndStrings [Java Application] C:\Users\Hrishil
Sorted integers in ascending order:
3
4
5
6
8

Sorted strings in ascending order:
C
I
O
P
U
```

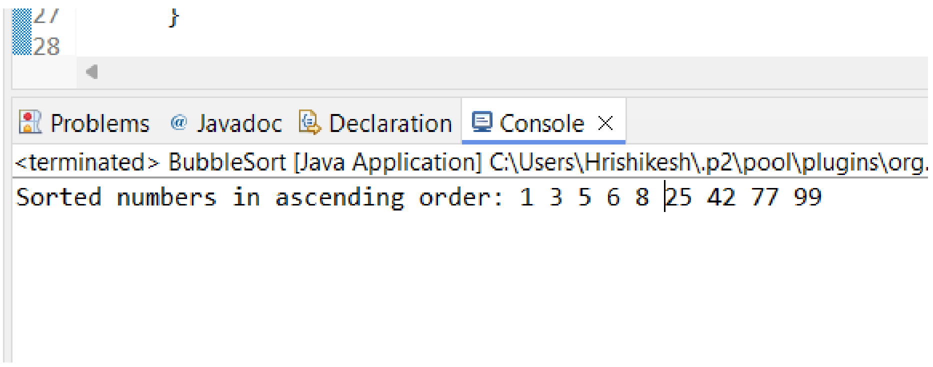
Poorvaja Bhorunde AF0403155

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

Program :

```
package demo;

public class BubbleSort {

public static void bubbleSort(int[] arr) {
int n = arr.length;
for (int i = 0; i < n - 1; i++) {
for (int j = 0; j < n - i - 1; j++) {
if (arr[j] > arr[j + 1]) {
// Swapping elements
int temp = arr[j];
arr[j] = arr[j + 1];
arr[j + 1] = temp;
}
}
}
}

public static void main(String[] args) {
int[] numbers = {8, 42, 3,25, 5, 6, 1, 77, 99};
bubbleSort(numbers);

System.out.print("Sorted numbers in ascending order: ");
for (int number : numbers) {
System.out.print(number + " ");
}
}

}
```

Output :



```
}
```

```
<terminated> BubbleSort [Java Application] C:\Users\Hrishikesh\.p2\pool\plugins\org.
Sorted numbers in ascending order: 1 3 5 6 8 25 42 77 99
```

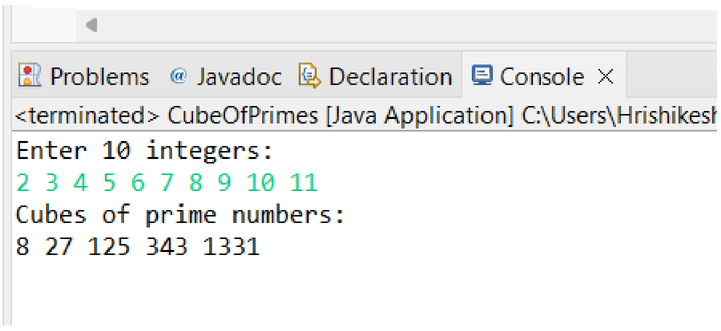
Poorvaja Bhorunde AF0403155

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

Program :

```
package demo;
import java.util.Scanner;
public class CubeOfPrimes {
public static boolean isPrime(int num) {
if (num <= 1) {
return false;
}
for (int i = 2; i * i <= num; i++) {
if (num % i == 0) {
return false;
}
}
return true;
}
public static void main(String[] args) {
Scanner s = new Scanner(System.in);
int[] numbers = new int[10];

System.out.println("Enter 10 integers:");
for (int i = 0; i < 10; i++) {
numbers[i] = s.nextInt();
}
System.out.println("Cubes of prime numbers:");
for (int num : numbers) {
if (isPrime(num)) {
System.out.print(num * num * num + " ");
}
}
System.out.println();
s.close();
}
}
```

Output :



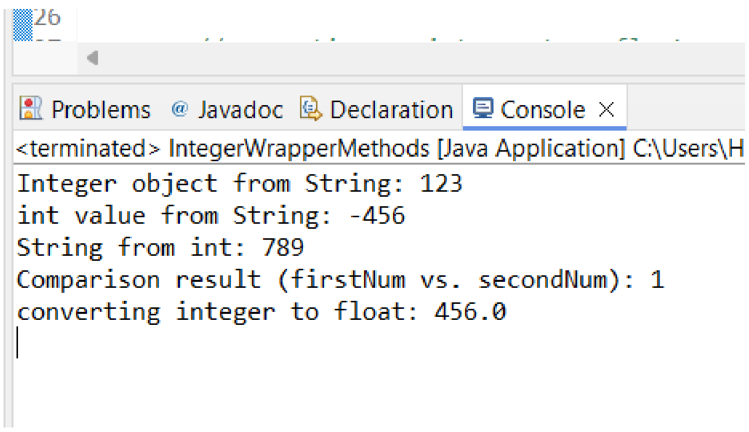
```
<terminated> CubeOfPrimes [Java Application] C:\Users\Hrishikesh
Enter 10 integers:
2 3 4 5 6 7 8 9 10 11
Cubes of prime numbers:
8 27 125 343 1331
```

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

Program :

```
package demo;

public class IntegerWrapperMethods {

public static void main(String[] args) {
// valueOf() Converting a string to an Integer
String str = "123";
Integer num = Integer.valueOf(str);
System.out.println("Integer object from String: " + num);

// using parseInt() converting  string to an int
String anotherStr = "-456";
int parsedNum = Integer.parseInt(anotherStr);
System.out.println("int value from String: " + parsedNum);

//  Converting int to a String
int number = 789;
String stringNum = Integer.toString(number);
System.out.println("String from int: " + stringNum);

// compareTo() - Compares two Integers
Integer firstNum = 50;
Integer secondNum = 30;
int comparison = firstNum.compareTo(secondNum);
System.out.println("Comparison result (firstNum vs. secondNum): " + comparison);

//converting an integer to a float
int IntegerNum=456;
float floatNum=Integer.valueOf(IntegerNum).floatValue();
System.out.println("converting integer to float: "+floatNum);
}
}
```

Output :



```
<terminated> IntegerWrapperMethods [Java Application] C:\Users\H
Integer object from String: 123
int value from String: -456
String from int: 789
Comparison result (firstNum vs. secondNum): 1
converting integer to float: 456.0
```

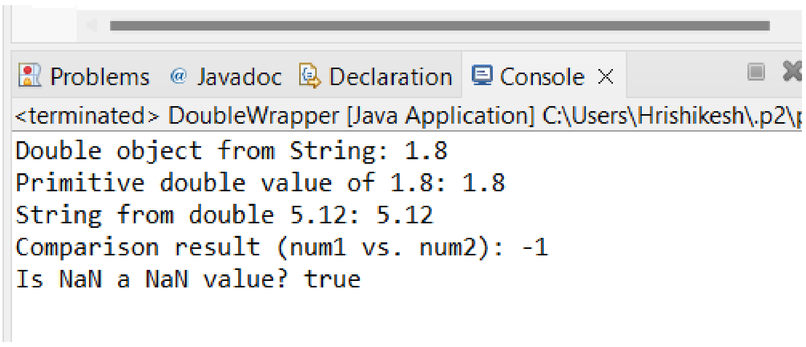
Poorvaja Bhorunde AF0403155

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

Program :

```
package demo;

public class DoubleWrapper {

public static void main(String[] args) {
// TODO Auto-generated method stub
double num1 = 5.12;
double num2 = 13.6;

// valueOf(String s) - Converting a string to a Double object
String str = "1.8";
Double doubleObj = Double.valueOf(str);
System.out.println("Double object from String: " + doubleObj);

//  doubleValue() - Returns the primitive double value
System.out.println("Primitive double value of " + doubleObj + ": " +
doubleObj.doubleValue());

//  toString(double d) - Converts a double to a String representation
System.out.println("String from double " + num1 + ": " + Double.toString(num1));

//  compareTo(Double anotherDouble) - Compares two Double objects
int comparison = Double.compare(num1, num2);
System.out.println("Comparison result (num1 vs. num2): " + comparison);
// Output will be 1 (positive because num1 is greater)

//  isNaN(double d) - Checks if a double value is Not a Number
double NaNvalue = Double.NaN;
System.out.println("Is " + NaNvalue + " a NaN value? " + Double.isNaN(NaNvalue));
}

}
```

Output :



```
<terminated> DoubleWrapper [Java Application] C:\Users\Hrishikesh\.p2\
Double object from String: 1.8
Primitive double value of 1.8: 1.8
String from double 5.12: 5.12
Comparison result (num1 vs. num2): -1
Is NaN a NaN value? true
```

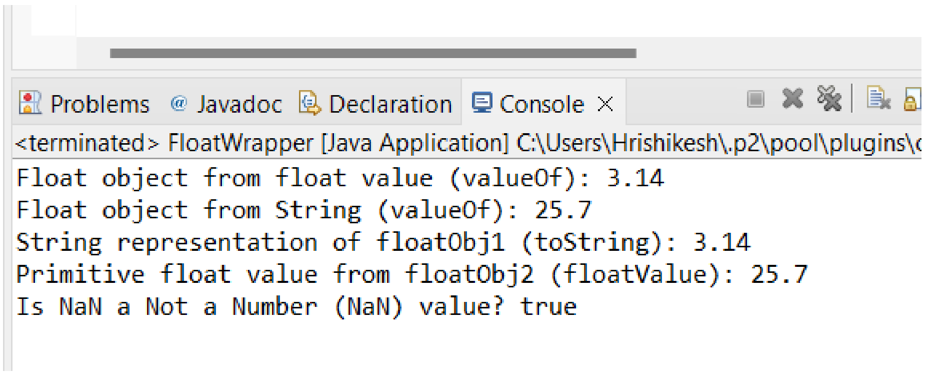
Poorvaja Bhorunde AF0403155

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

Program :

```java
package demo;

public class FloatWrapper {

public static void main(String[] args) {
// TODO Auto-generated method stub
// 1. valueOf(float): Create a Float object from a float value
float num1 = 3.14f;
Float floatObj1 = Float.valueOf(num1);
System.out.println("Float object from float value (valueOf): " + floatObj1);

// 2. valueOf(String): Create a Float object from a String representation
String strNum = "25.7";
Float floatObj2 = Float.valueOf(strNum);
System.out.println("Float object from String (valueOf): " + floatObj2);

// 3. toString(): Convert a Float object to its String representation
System.out.println("String representation of floatObj1 (toString): " +
floatObj1.toString());

// 4. floatValue(): Get the primitive float value from a Float object
float primitiveFloat = floatObj2.floatValue();
System.out.println("Primitive float value from floatObj2 (floatValue): " +
primitiveFloat);

// 5. isNaN(float): Check if a float value is Not a Number (NaN)
System.out.println("Is " + Float.NaN + " a Not a Number (NaN) value? " +
Float.isNaN(Float.NaN)); // Create new NaN each time
}

}
```

Output :



```
<terminated> FloatWrapper [Java Application] C:\Users\Hrishikesh\.p2\pool\plugins\
Float object from float value (valueOf): 3.14
Float object from String (valueOf): 25.7
String representation of floatObj1 (toString): 3.14
Primitive float value from floatObj2 (floatValue): 25.7
Is NaN a Not a Number (NaN) value? true
```

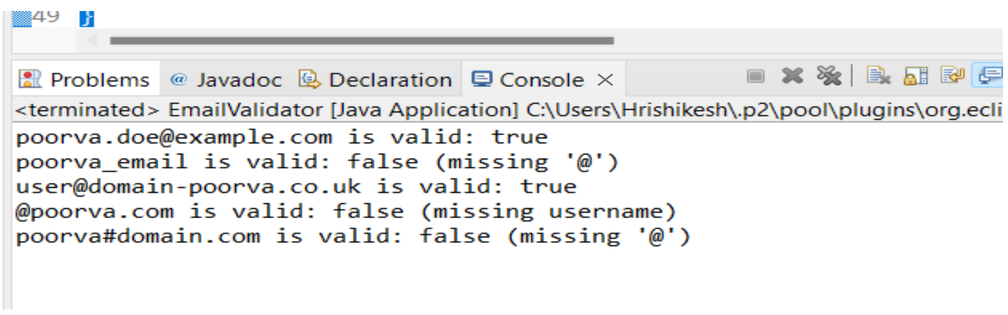
Poorvaja Bhorunde AF0403155

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 demo;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class EmailValidator {
public static void main(String[] args) {
// Regular expression for validating email addresses
String regex = "^[\\w!#$%&'*+/=?^_`{|}~-]+(?:\\.[\\w!#$%&'*+/=?^_`{|}~-
]+)*@(?:[\\w](?:[\\w-]*[\\w])?\\.)+[\\w](?:[\\w-]*[\\w])?$";
// Array of email addresses to validate
String[] emails = {
"poorva.doe@example.com",
"poorva_email",
"user@domain-poorva.co.uk",
"@poorva.com",
"poorva#domain.com"
};
// Iterate over each email address
for (String email : emails) {
// Compile the regex pattern
Pattern pattern = Pattern.compile(regex);
// Match the email against the pattern
Matcher matcher = pattern.matcher(email);
if (matcher.matches()) {
System.out.println(email + " is valid: true");
} else {
System.out.print(email + " is valid: false (");
// Specific checks for reasons
if (!email.contains("@")) {
System.out.println("missing '@')");
} else {
String[] parts = email.split("@");
if (parts[0].isEmpty()) {
System.out.println("missing username)");
} else if (!parts[1].contains(".")) {
System.out.println("missing domain)");
} else if (email.contains("#") || email.contains(" ")) {
System.out.println("invalid characters)");
} else {
System.out.println("general invalid format)");
}}}}}}
```

Output :



```
<terminated> EmailValidator [Java Application] C:\Users\Hrishikesh\.p2\pool\plugins\org.ecli
poorva.doe@example.com is valid: true
poorva_email is valid: false (missing '@')
user@domain-poorva.co.uk is valid: true
@poorva.com is valid: false (missing username)
poorva#domain.com is valid: false (missing '@')
```

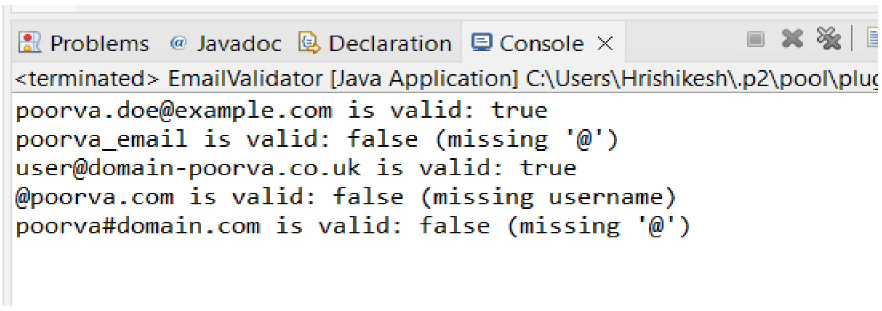
Poorvaja Bhorunde AF0403155

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

Program :

```
package demo;

import java.util.regex.Pattern;

import java.util.regex.Matcher;

public class PhoneNumberValidator {

public static boolean isValidPhoneNumber(String phoneNumber) {

String regex = "^\\(91[0-9]{2}\\) [0-9]{3}-[0-9]{4}$";

Pattern pattern = Pattern.compile(regex);

Matcher matcher = pattern.matcher(phoneNumber);

return matcher.matches();

}

public static void main(String[] args) {

String[] phoneNumbers = {

"(9123) 456-7890",  // Valid

"123-456-7890",     // Invalid (missing 91)

"(123) 456-789a",   // Invalid (non-digit character)

"(9145) 6789012",    // Valid

};

for (String phoneNumber : phoneNumbers) {

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

}}}
```

Output :



```
<terminated> EmailValidator [Java Application] C:\Users\Hrishikesh\.p2\pool\plu
poorva.doe@example.com is valid: true
poorva_email is valid: false (missing '@')
user@domain-poorva.co.uk is valid: true
@poorva.com is valid: false (missing username)
poorva#domain.com is valid: false (missing '@')
```