Assignment-2(Sandeep sir)

1. Working with java.lang.Boolean

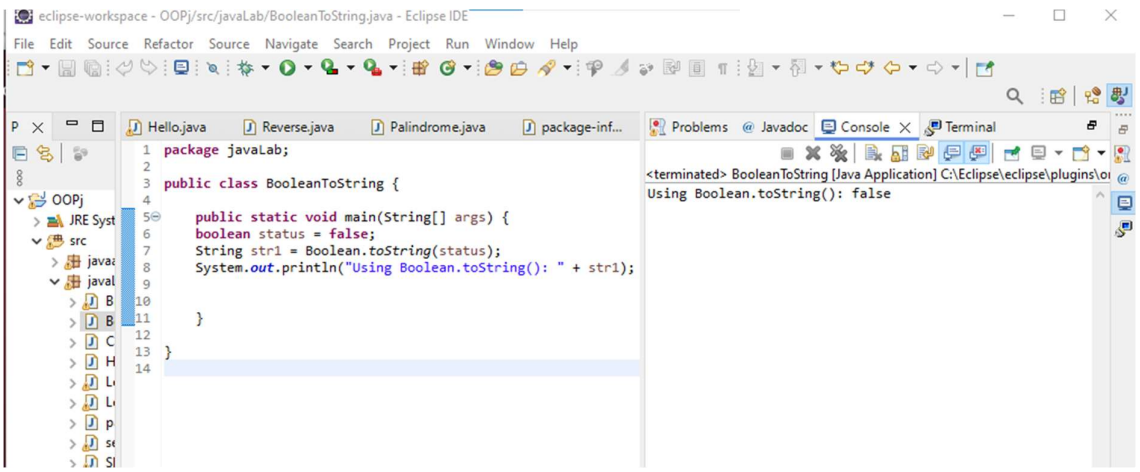
a. Explore the Java API documentation for java.lang.Boolean and observe its modifiers and super types.

Field Summary

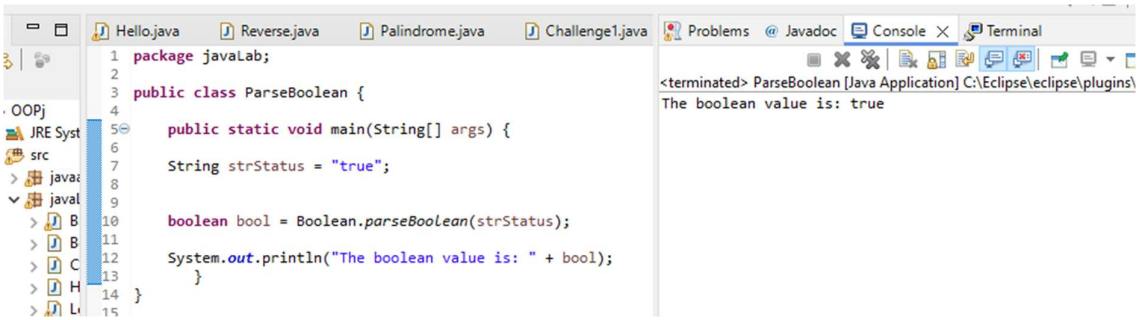
Fields

Modifier and Type	Field and Description
`static Boolean`	**FALSE** The `Boolean` object corresponding to the primitive value `false`.
`static Boolean`	**TRUE** The `Boolean` object corresponding to the primitive value `true`.
`static Class<Boolean>`	**TYPE** The Class object representing the primitive type boolean.

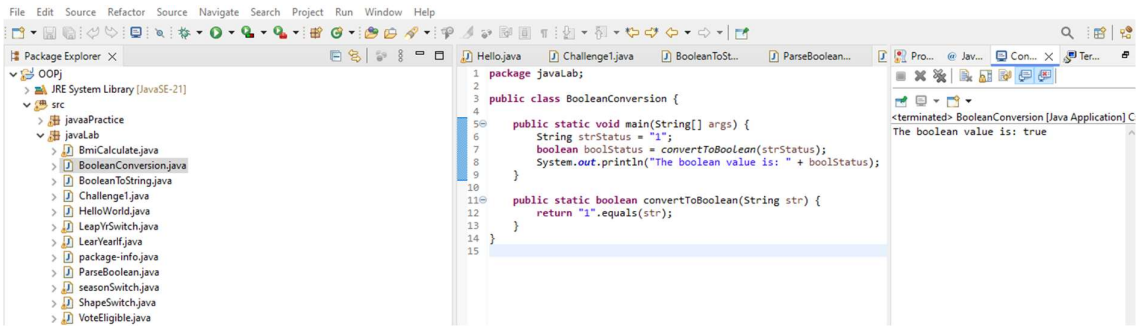
b. Declare a method-local variable status of type boolean with the value true and convert it to a String using the toString method. (Hint: Use Boolean.toString(Boolean)).



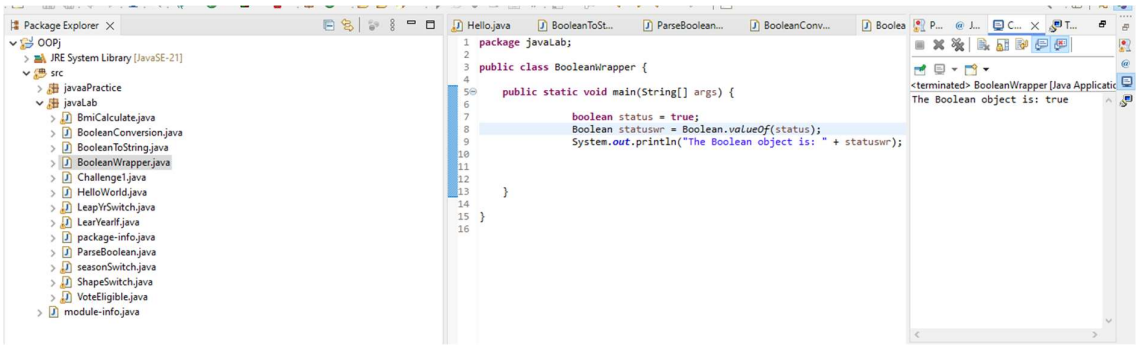
c. Declare a method-local variable strStatus of type String with the value "true" and convert it to a boolean using the parseBoolean method. (Hint: Use Boolean.parseBoolean(String)).



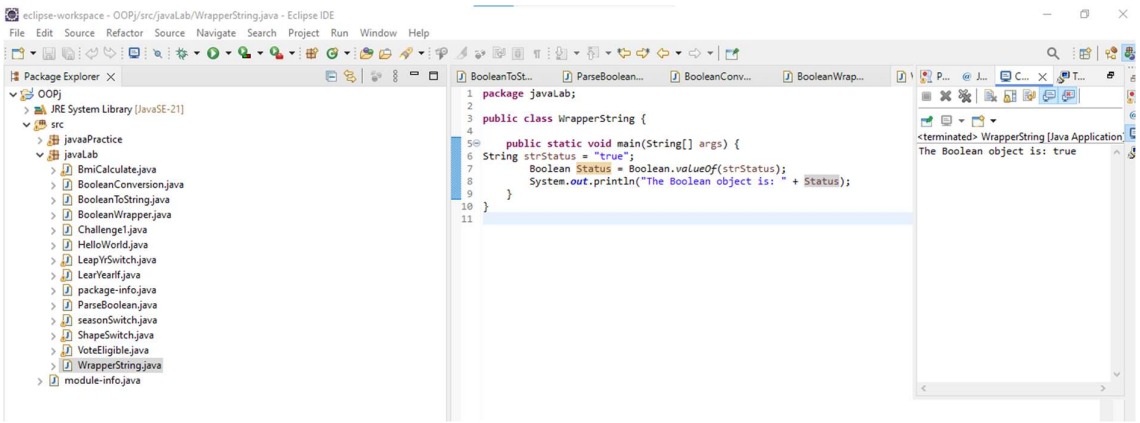
d. Declare a method-local variable strStatus of type String with the value "1" or "0" and attempt to convert it to a boolean. (Hint: parseBoolean method will not work as expected with "1" or "0").



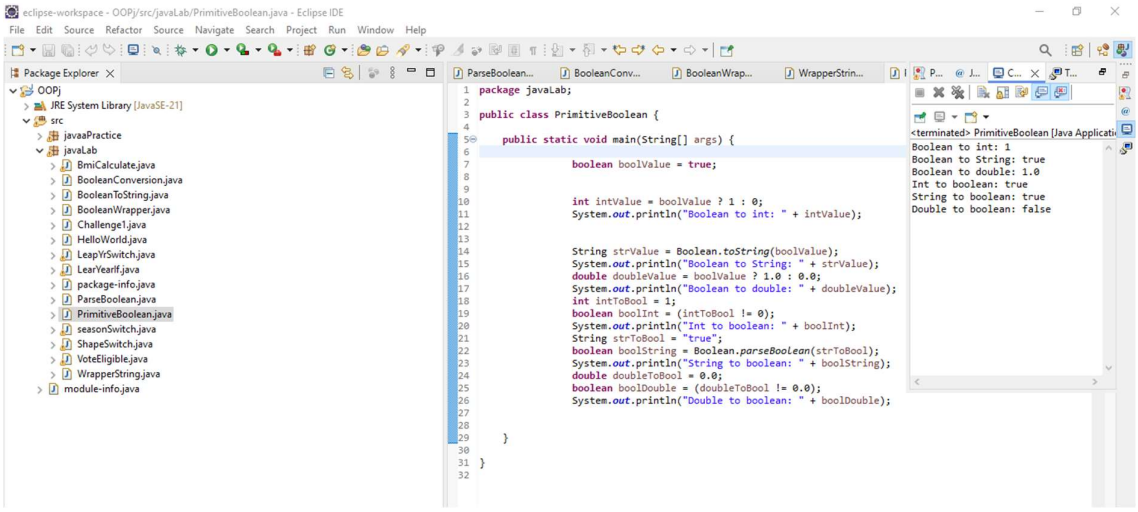
e. Declare a method-local variable status of type boolean with the value true and convert it to the corresponding wrapper class using Boolean.valueOf(). (Hint: Use Boolean.valueOf(boolean)).



f. Declare a method-local variable strStatus of type String with the value "true" and convert it to the corresponding wrapper class using Boolean.valueOf(). (Hint: Use Boolean.valueOf(String)).



g. Experiment with converting a boolean value into other primitive types or vice versa and observe the results.



2. Working with java.lang.Byte

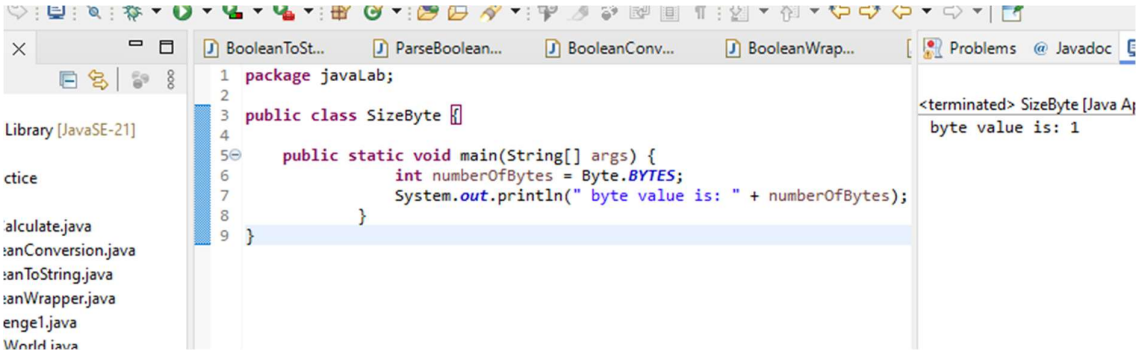
a. Explore the Java API documentation for java.lang.Byte and observe its modifiers and super types.

Field Summary

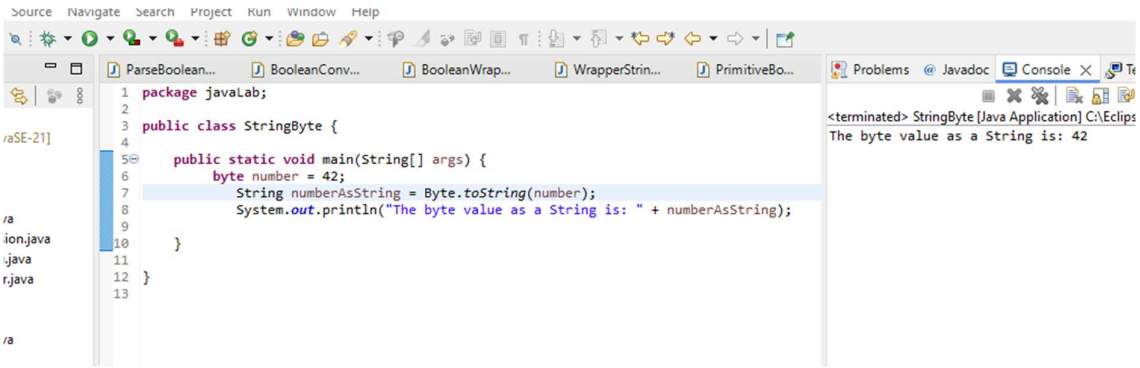
Fields

Modifier and Type	Field and Description
`static int`	**BYTES** The number of bytes used to represent a byte value in two's complement binary form.
`static byte`	**MAX_VALUE** A constant holding the maximum value a byte can have, 2^7-1.
`static byte`	**MIN_VALUE** A constant holding the minimum value a byte can have, -2^7.
`static int`	**SIZE** The number of bits used to represent a byte value in two's complement binary form.
`static Class<Byte>`	**TYPE** The Class instance representing the primitive type byte.

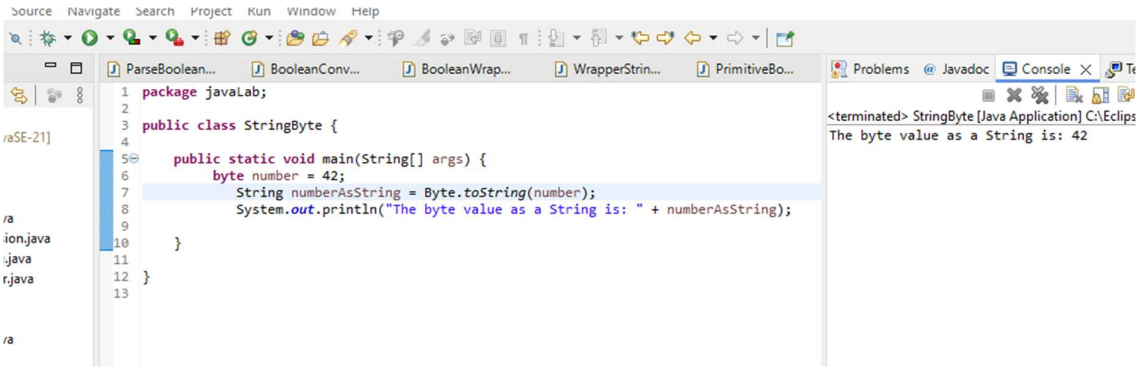
b. Write a program to test how many bytes are used to represent a byte value using the BYTES field. (Hint: Use Byte.BYTES).



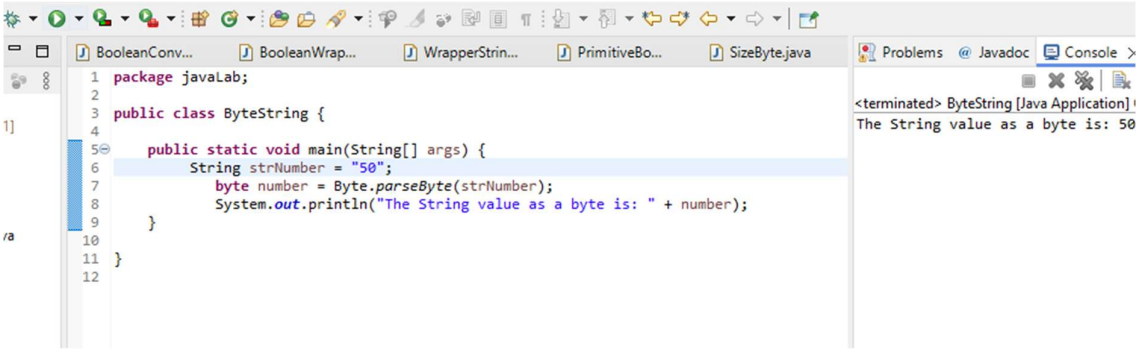
c. Write a program to find the minimum and maximum values of byte using the MIN_VALUE and MAX_VALUE fields. (Hint: Use Byte.MIN_VALUE and Byte.MAX_VALUE).



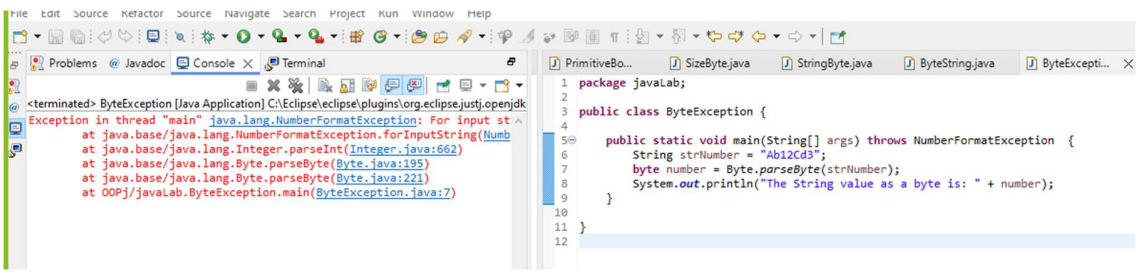
d. Declare a method-local variable number of type byte with some value and convert it to a String using the toString method. (Hint: Use Byte.toString(byte)).



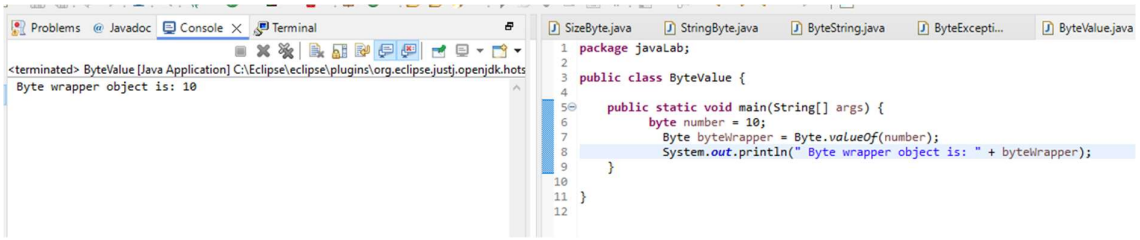
e. Declare a method-local variable strNumber of type String with some value and convert it to a byte value using the parseByte method. (Hint: Use Byte.parseByte(String)).



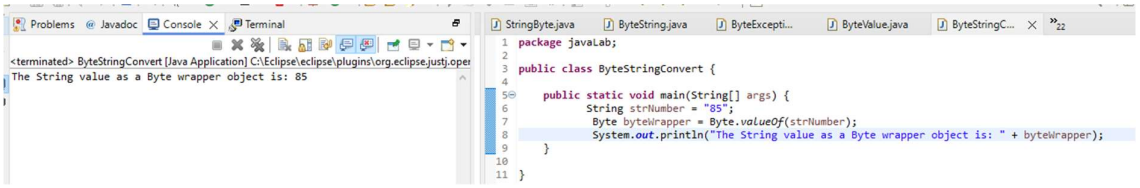
f. Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to a byte value. (Hint: parseByte method will throw a NumberFormatException).



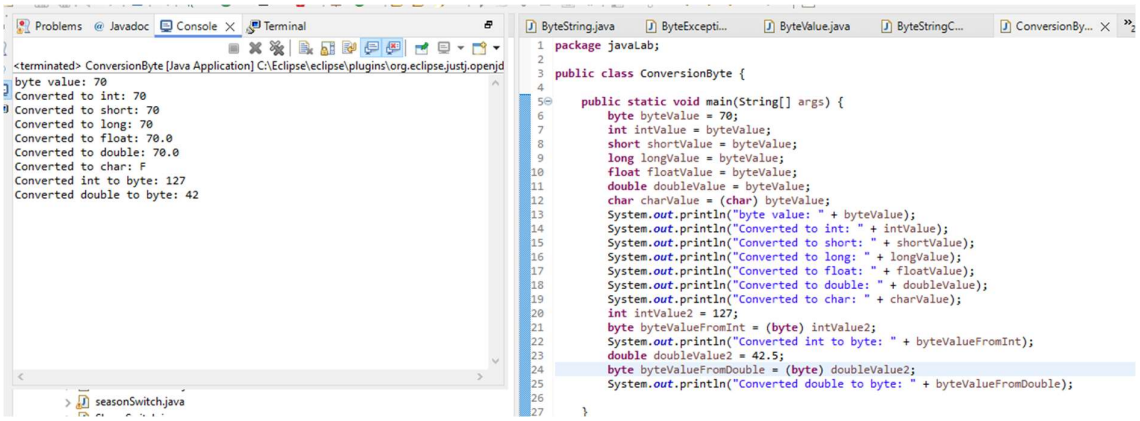
g. Declare a method-local variable number of type byte with some value and convert it to the corresponding wrapper class using Byte.valueOf(). (Hint: Use Byte.valueOf(byte)).



h. Declare a method-local variable strNumber of type String with some byte value and convert it to the corresponding wrapper class using Byte.valueOf(). (Hint: Use Byte.valueOf(String))



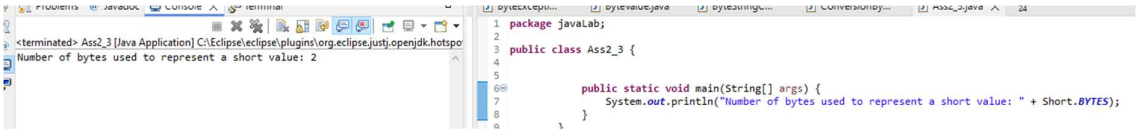
I .Experiment with converting a byte value into other primitive types or vice versa and observe the results.



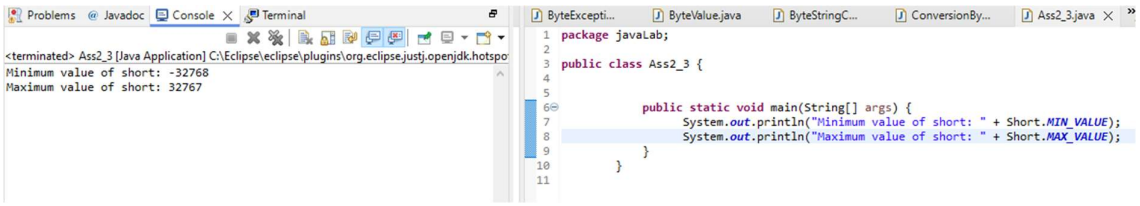
3. Working with java.lang.Short

a. Explore the Java API documentation for java.lang.Short and observe its modifiers and super types.

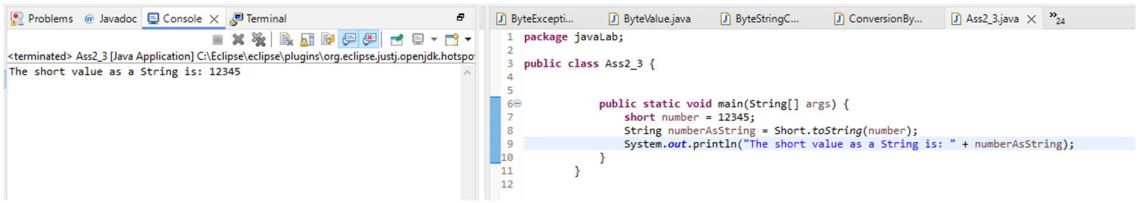
b. Write a program to test how many bytes are used to represent a short value using the BYTES field. (Hint: Use Short.BYTES).



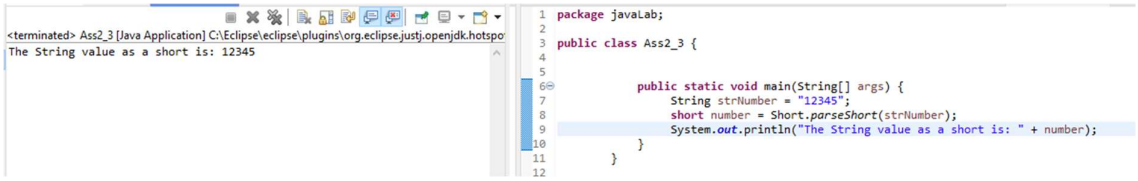
c. Write a program to find the minimum and maximum values of short using the MIN_VALUE and MAX_VALUE fields. (Hint: Use Short.MIN_VALUE and Short.MAX_VALUE).



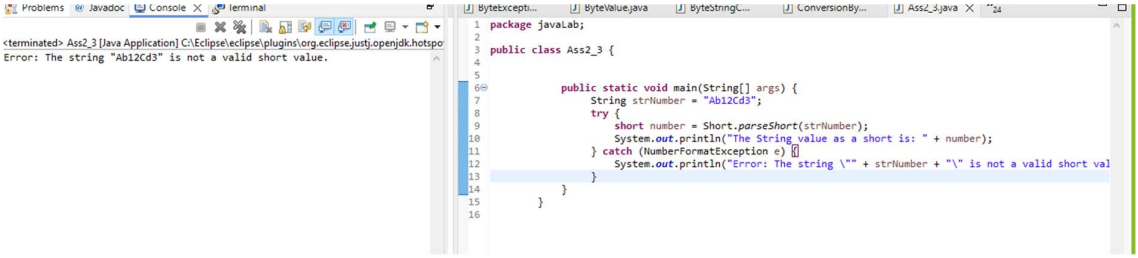
d. Declare a method-local variable number of type short with some value and convert it to a String using the toString method. (Hint: Use Short.toString(short)).



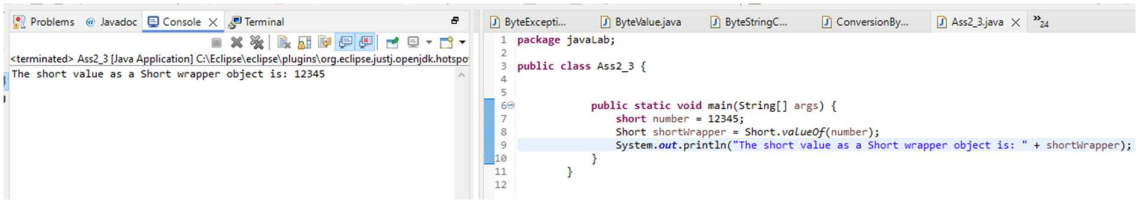
e. Declare a method-local variable strNumber of type String with some value and convert it to a short value using the parseShort method. (Hint: Use Short.parseShort(String)).



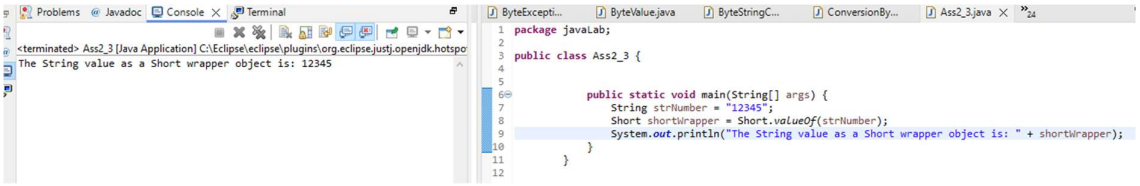
f. Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to a short value. (Hint: parseShort method will throw a NumberFormatException).



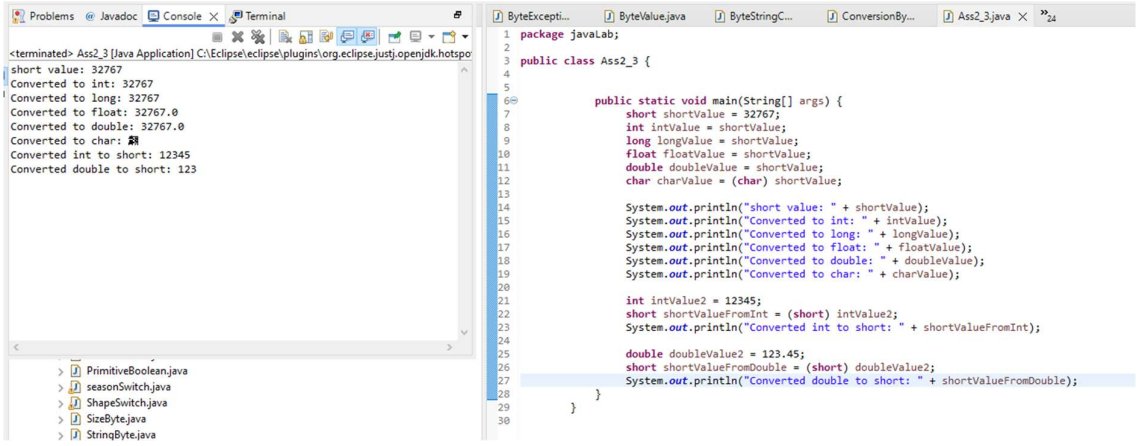
g. Declare a method-local variable number of type short with some value and convert it to the corresponding wrapper class using Short.valueOf(). (Hint: Use Short.valueOf(short)).



h. Declare a method-local variable strNumber of type String with some short value and convert it to the corresponding wrapper class using Short.valueOf(). (Hint: Use Short.valueOf(String)).



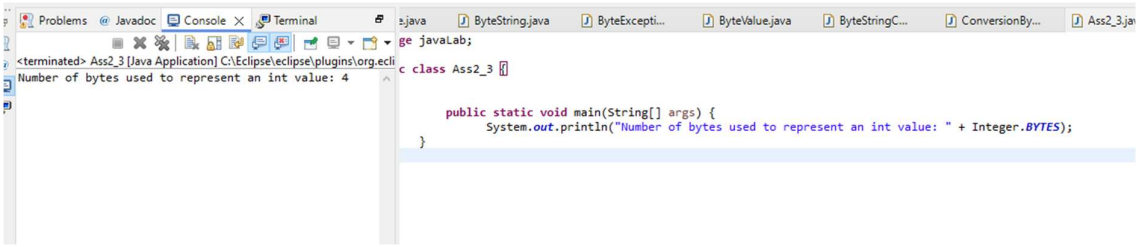
i. Experiment with converting a short value into other primitive types or vice versa and observe the results.



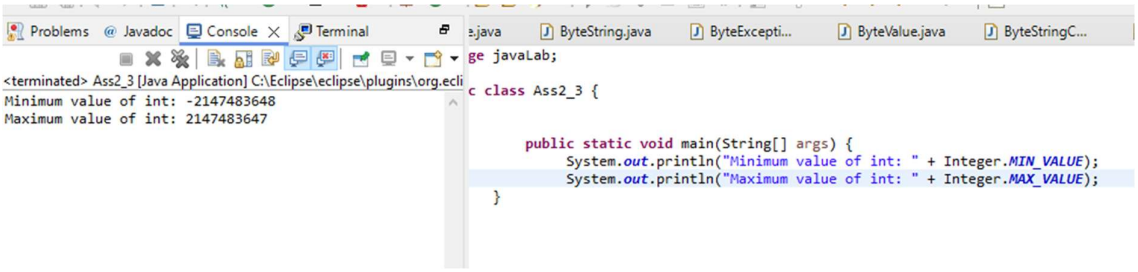
4. Working with java.lang.Integer

a. Explore the Java API documentation for java.lang.Integer and observe its modifiers and super types.

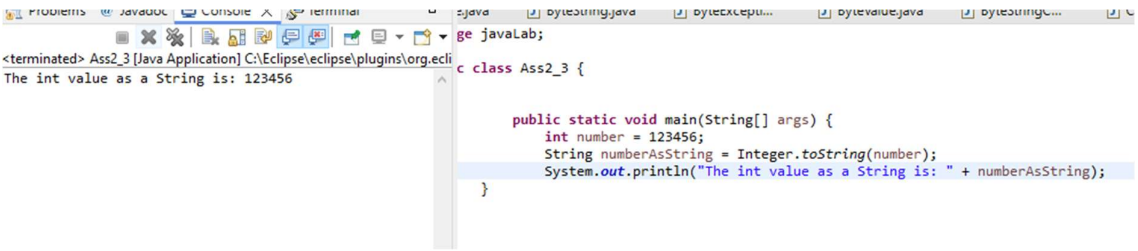
b. Write a program to test how many bytes are used to represent an int value using the BYTES field. (Hint: Use Integer.BYTES).



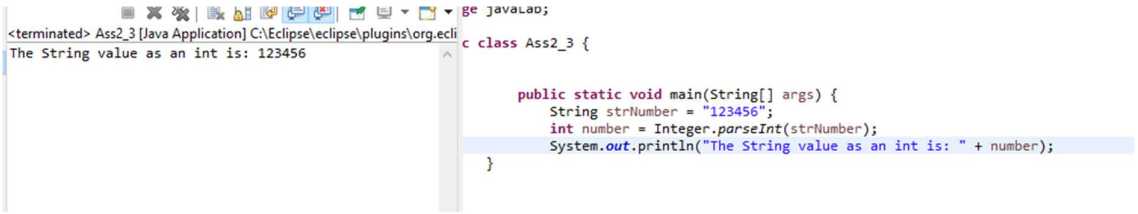
c. Write a program to find the minimum and maximum values of int using the MIN_VALUE and MAX_VALUE fields. (Hint: Use Integer.MIN_VALUE and Integer.MAX_VALUE).



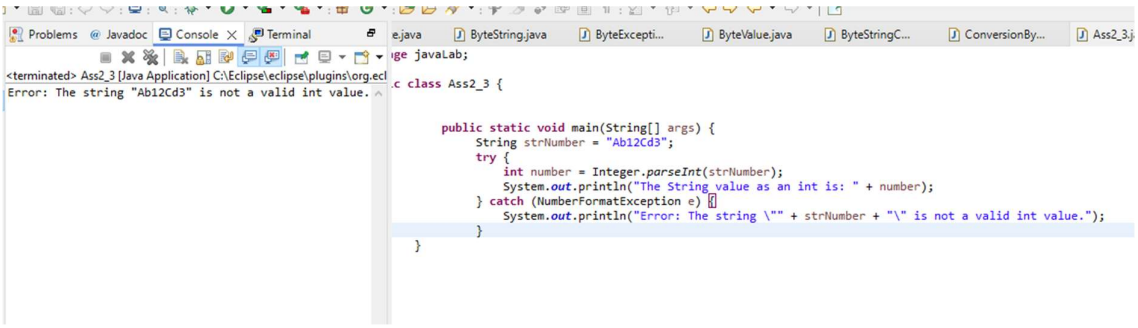
d. Declare a method-local variable number of type int with some value and convert it to a String using the toString method. (Hint: Use Integer.toString(int)).



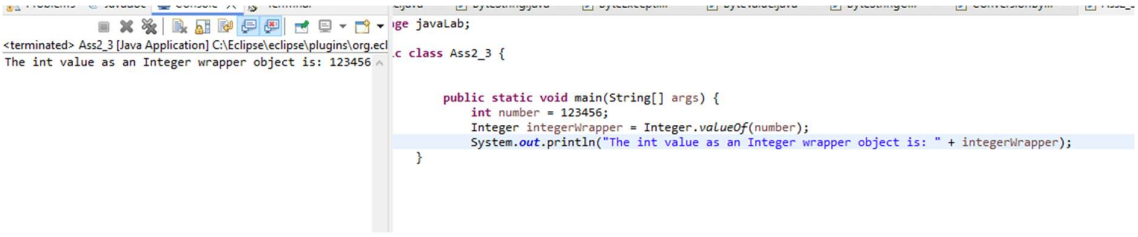
e. Declare a method-local variable strNumber of type String with some value and convert it to an int value using the parseInt method. (Hint: Use Integer.parseInt(String)).



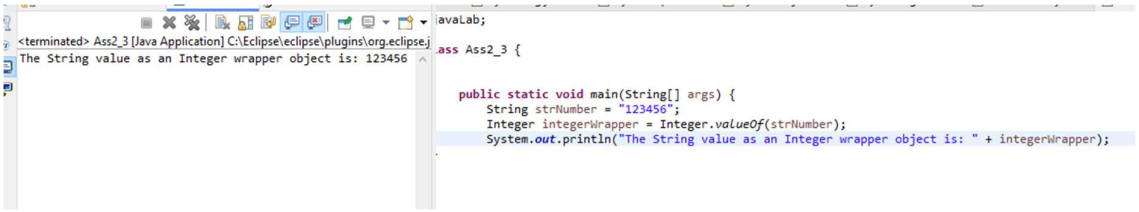
f. Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to an int value. (Hint: parseInt method will throw a NumberFormatException).



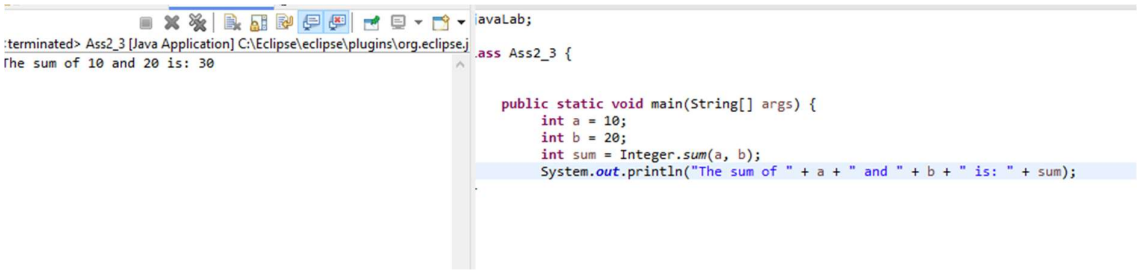
g. Declare a method-local variable number of type int with some value and convert it to the corresponding wrapper class using Integer.valueOf(). (Hint: Use Integer.valueOf(int)).



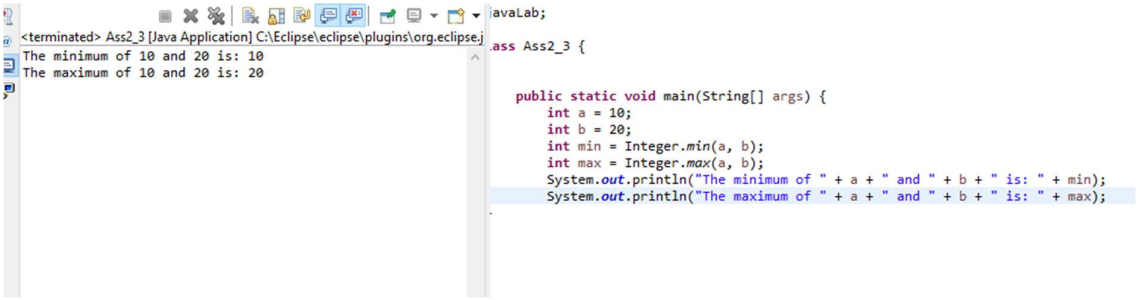
h. Declare a method-local variable strNumber of type String with some integer value and convert it to the corresponding wrapper class using Integer.valueOf(). (Hint: Use Integer.valueOf(String)).



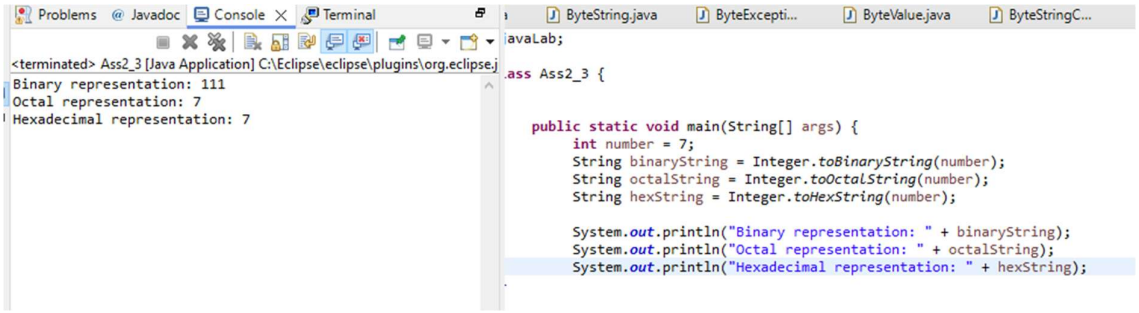
i. Declare two integer variables with values 10 and 20, and add them using a method from the Integer class. (Hint: Use Integer.sum(int, int)).



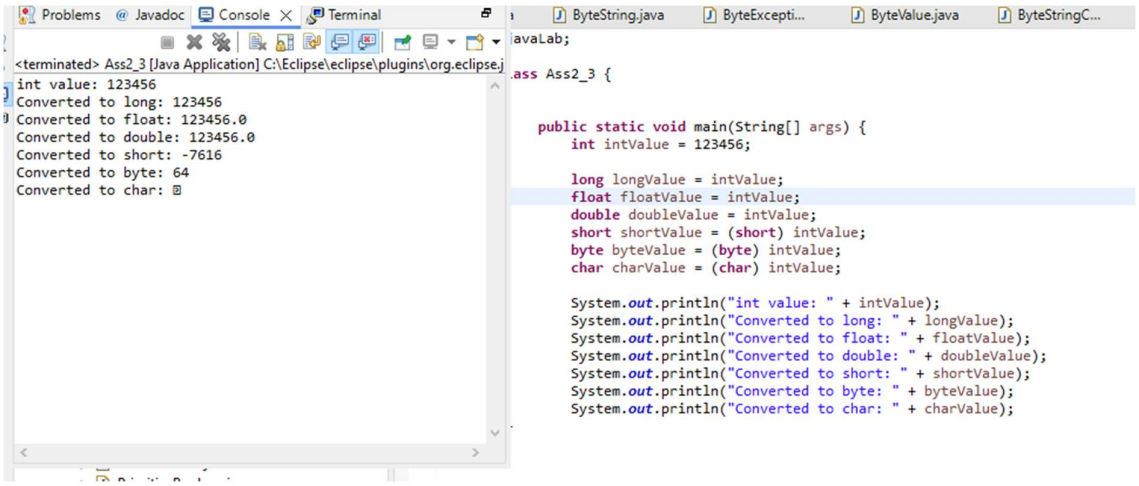
j. Declare two integer variables with values 10 and 20, and find the minimum and maximum values using the Integer class. (Hint: Use Integer.min(int, int) and Integer.max(int, int)).



k. Declare an integer variable with the value 7. Convert it to binary, octal, and hexadecimal strings using methods from the Integer class. (Hint: Use Integer.toBinaryString(int), Integer.toOctalString(int), and Integer.toHexString(int)).



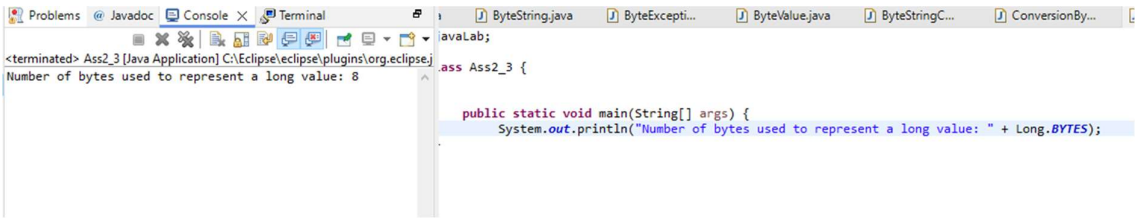
l. Experiment with converting an int value into other primitive types or vice versa and observe the results.



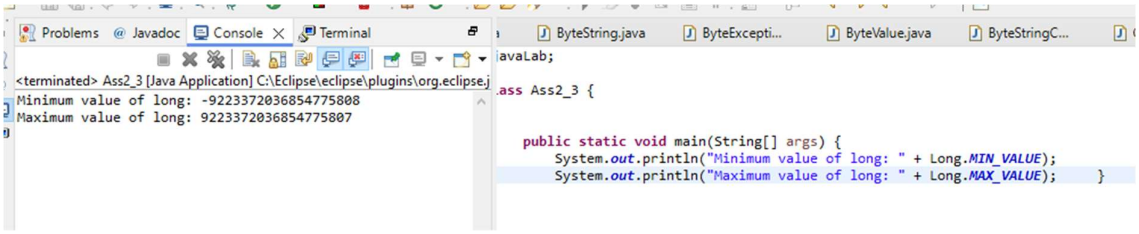
5. Working with java.lang.Long

a. Explore the Java API documentation for java.lang.Long and observe its modifiers and super types.

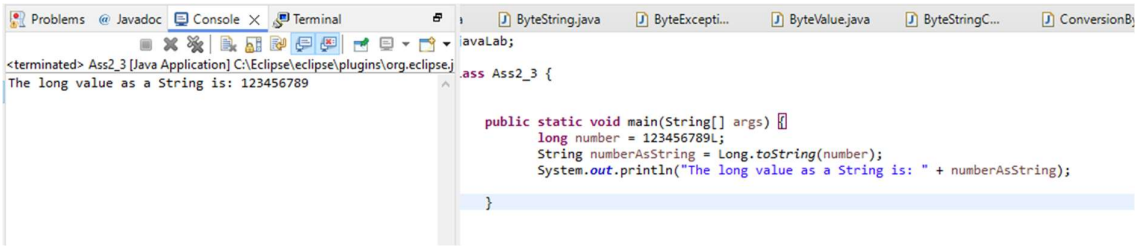
b. Write a program to test how many bytes are used to represent a long value using the BYTES field. (Hint: Use Long.BYTES).



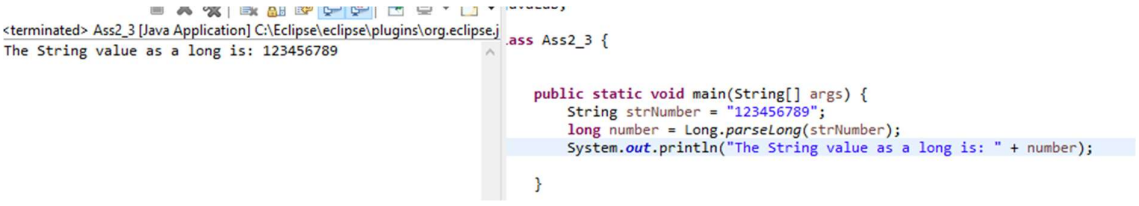
c. Write a program to find the minimum and maximum values of long using the MIN_VALUE and MAX_VALUE fields. (Hint: Use Long.MIN_VALUE and Long.MAX_VALUE).



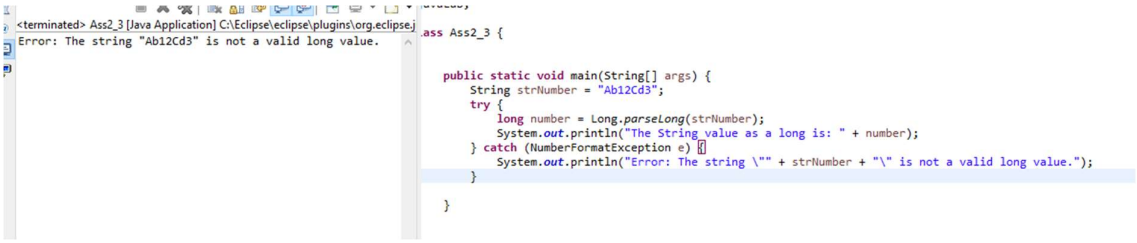
d. Declare a method-local variable number of type long with some value and convert it to a String using the toString method. (Hint: Use Long.toString(long)).



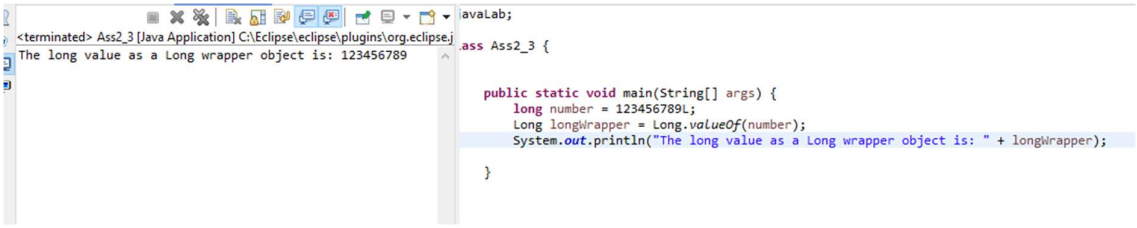
e. Declare a method-local variable strNumber of type String with some value and convert it to a long value using the parseLong method. (Hint: Use Long.parseLong(String)).



f. Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to a long value. (Hint: parseLong method will throw a NumberFormatException).

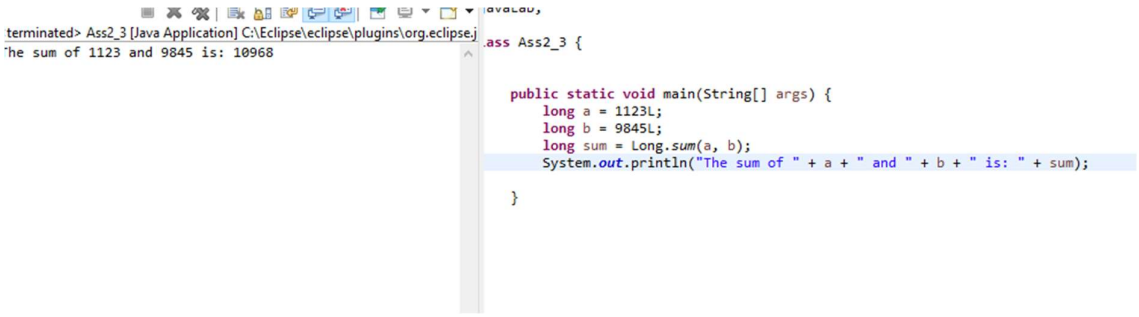


g. Declare a method-local variable number of type long with some value and convert it to the corresponding wrapper class using Long.valueOf(). (Hint: Use Long.valueOf(long)).

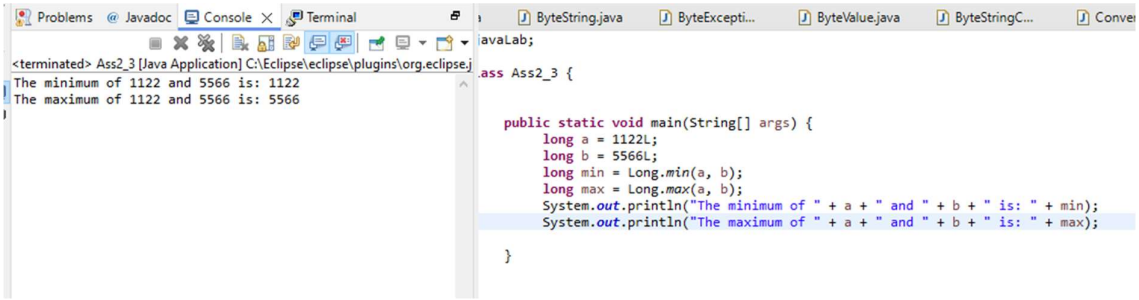


h. Declare a method-local variable strNumber of type String with some long value and convert it to the corresponding wrapper class using Long.valueOf(). (Hint: Use Long.valueOf(String)).

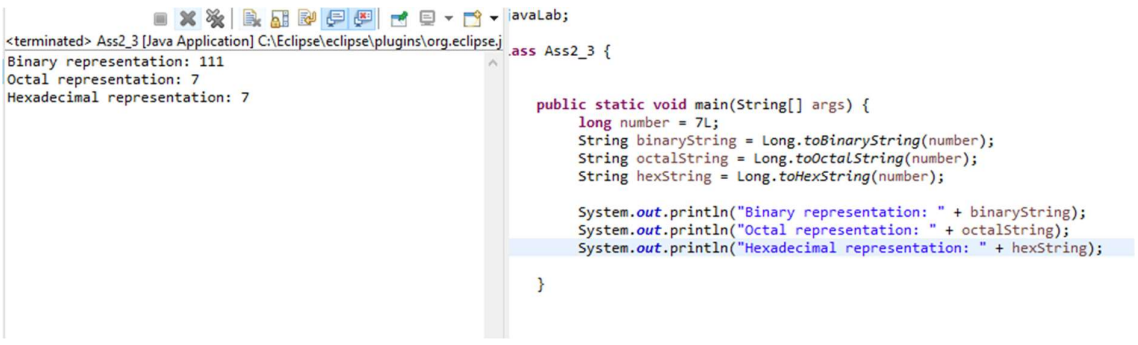
i. Declare two long variables with values 1123 and 9845, and add them using a method from the Long class. (Hint: Use Long.sum(long, long)).



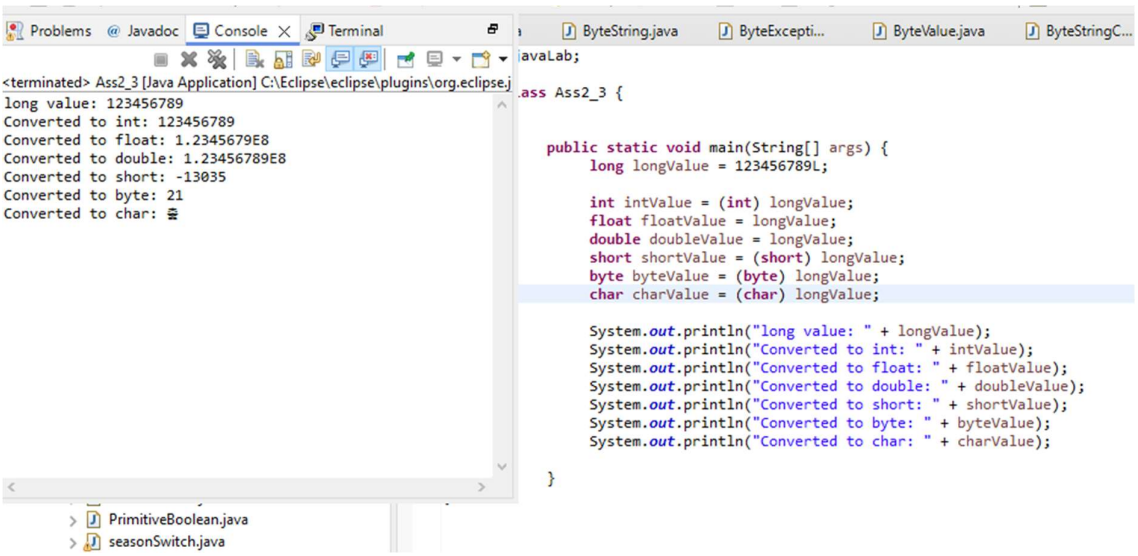
j. Declare two long variables with values 1122 and 5566, and find the minimum and maximum values using the Long class. (Hint: Use Long.min(long, long) and Long.max(long, long)).



k. Declare a long variable with the value 7. Convert it to binary, octal, and hexadecimal strings using methods from the Long class. (Hint: Use Long.toBinaryString(long), Long.toOctalString(long), and Long.toHexString(long)).



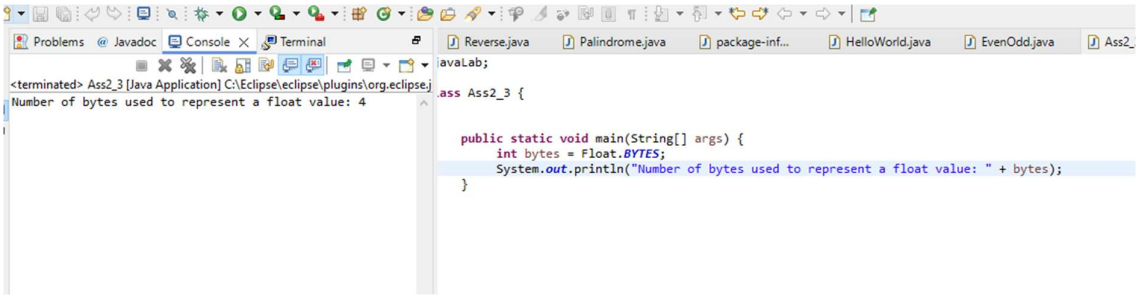
l. Experiment with converting a long value into other primitive types or vice versa and observe the results.



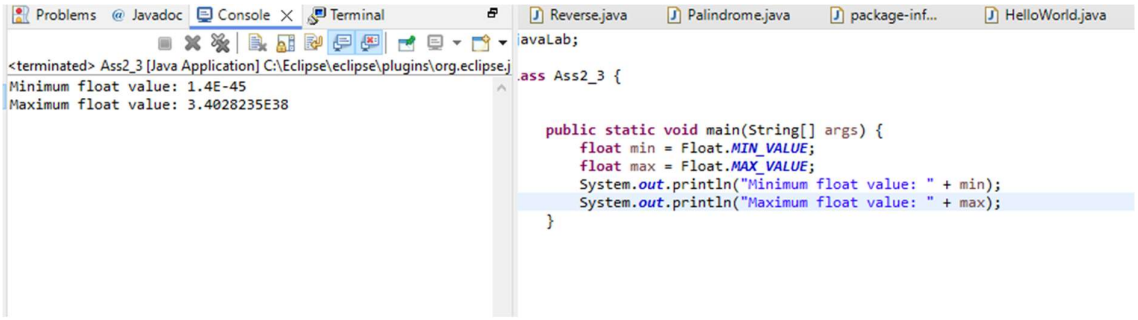
6. Working with java.lang.Float

a. Explore the Java API documentation for java.lang.Float and observe its modifiers and super types.

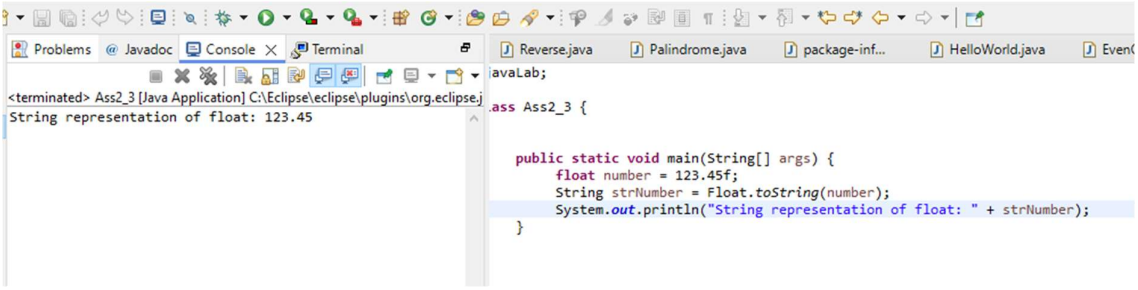
b. Write a program to test how many bytes are used to represent a float value using the BYTES field. (Hint: Use Float.BYTES).



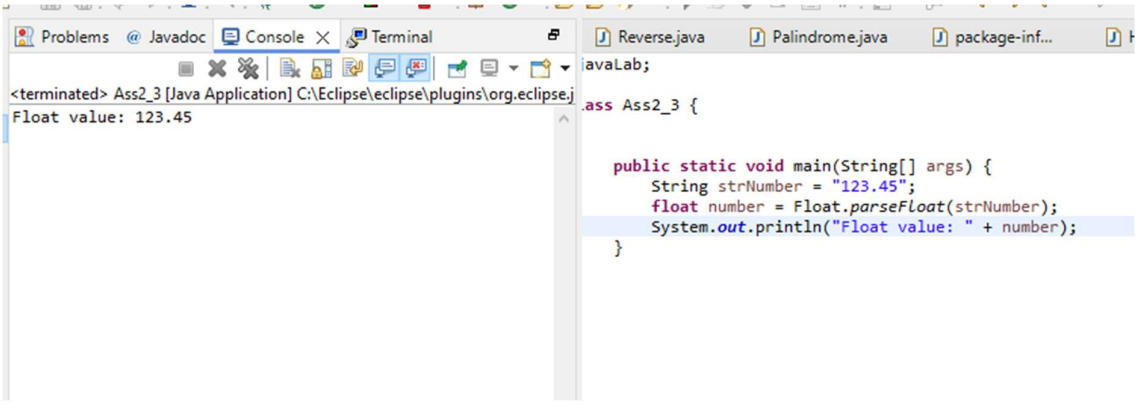
c. Write a program to find the minimum and maximum values of float using the MIN_VALUE and MAX_VALUE fields. (Hint: Use Float.MIN_VALUE and Float.MAX_VALUE).



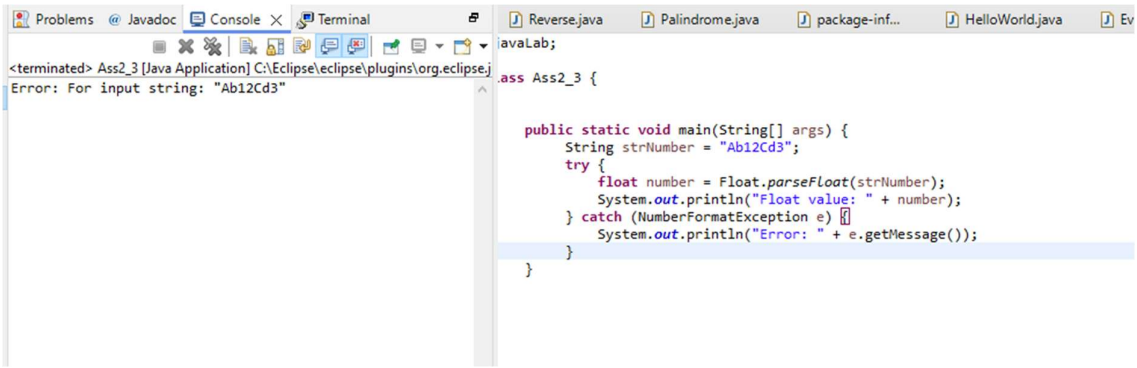
d. Declare a method-local variable number of type float with some value and convert it to a String using the toString method. (Hint: Use Float.toString(float)).



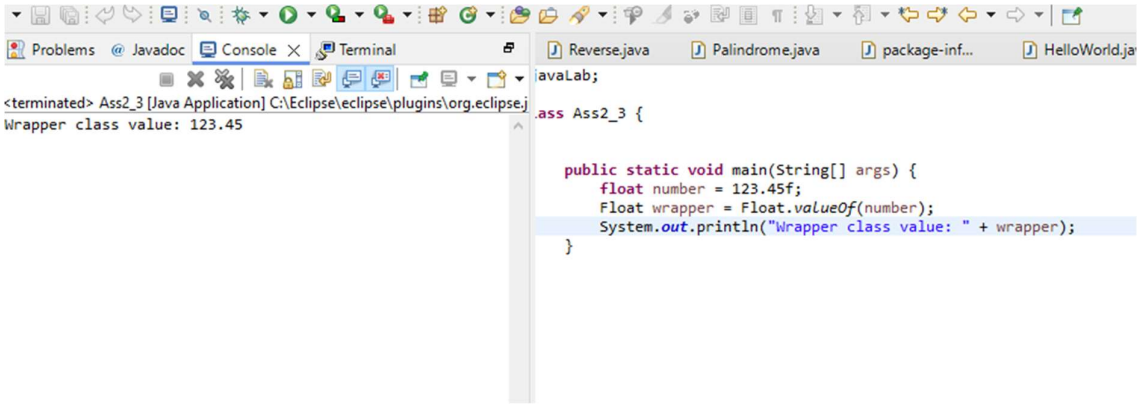
e. Declare a method-local variable strNumber of type String with some value and convert it to a float value using the parseFloat method. (Hint: Use Float.parseFloat(String)).



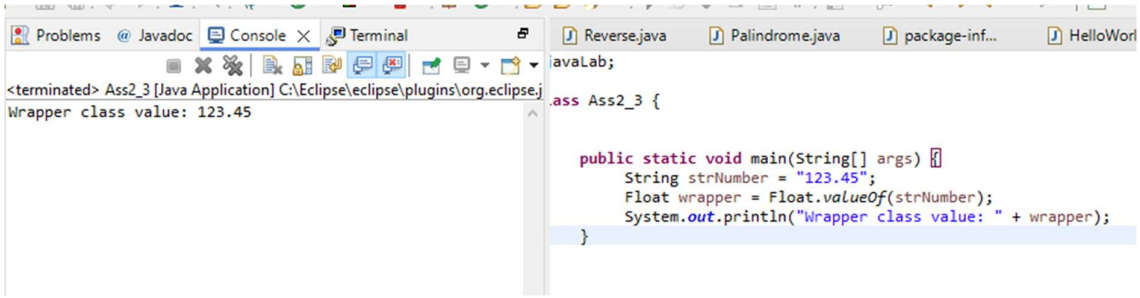
f. Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to a float value. (Hint: parseFloat method will throw a NumberFormatException).



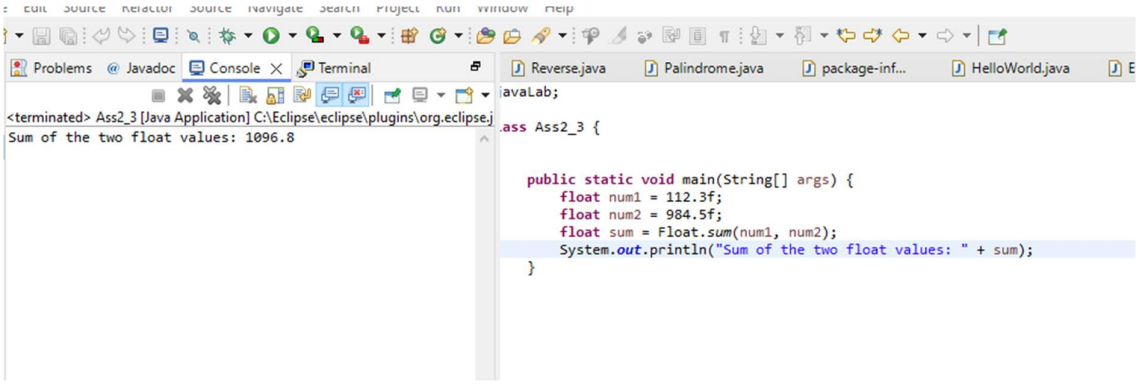
g. Declare a method-local variable number of type float with some value and convert it to the corresponding wrapper class using Float.valueOf(). (Hint: Use Float.valueOf(float)).



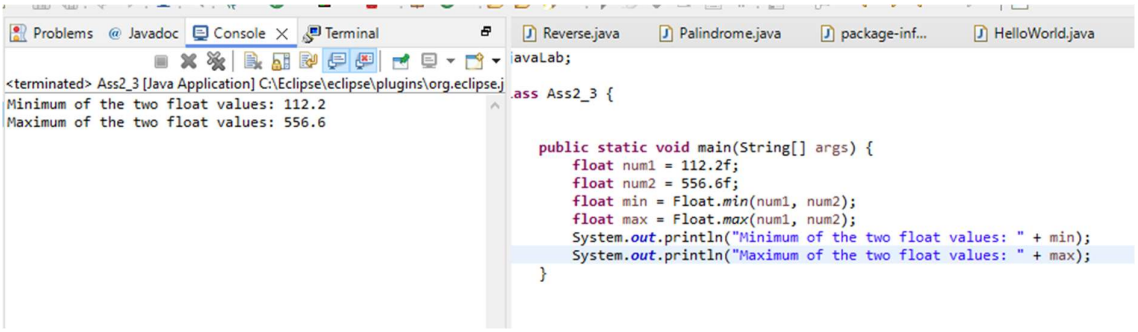
h. Declare a method-local variable strNumber of type String with some float value and convert it to the corresponding wrapper class using Float.valueOf(). (Hint: Use Float.valueOf(String)).



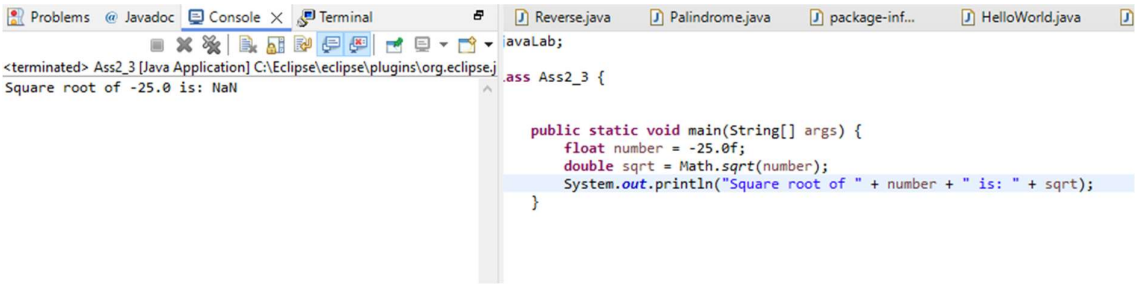
i. Declare two float variables with values 112.3 and 984.5, and add them using a method from the Float class. (Hint: Use Float.sum(float, float)).



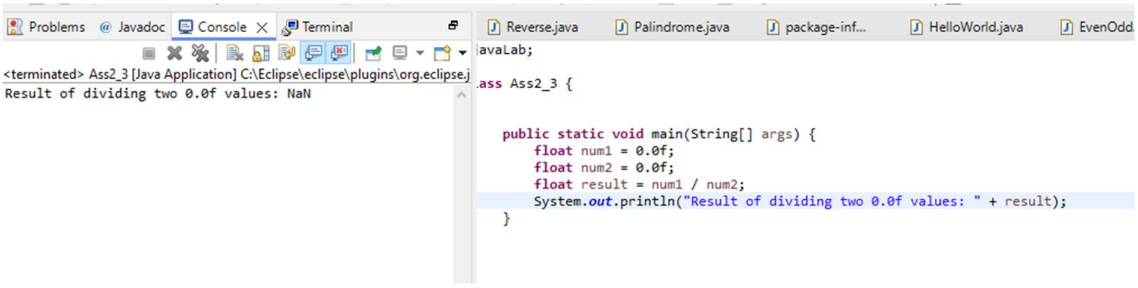
j. Declare two float variables with values 112.2 and 556.6, and find the minimum and maximum values using the Float class. (Hint: Use Float.min(float, float) and Float.max(float, float)).



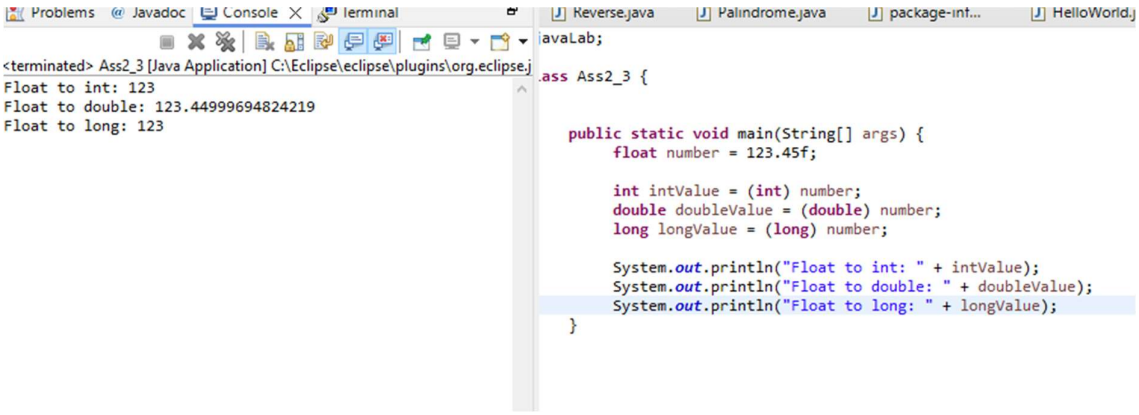
k. Declare a float variable with the value -25.0f. Find the square root of this value. (Hint: Use Math.sqrt() method).



l. Declare two float variables with the same value, 0.0f, and divide them. (Hint: Observe the result and any special floating-point behavior)..



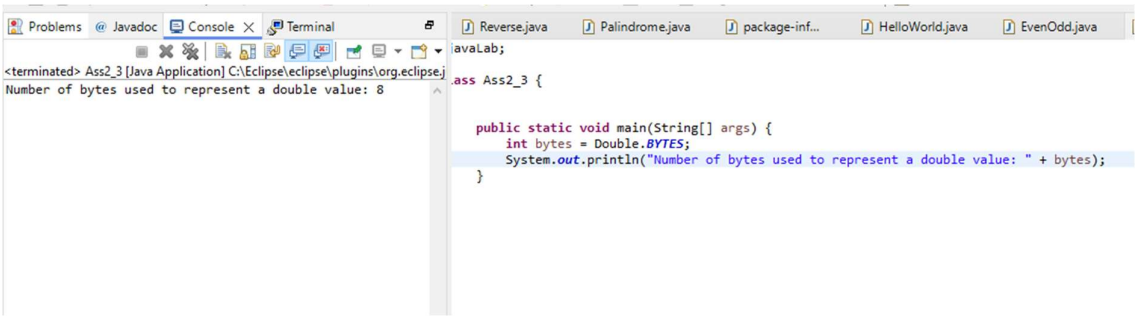
m. Experiment with converting a float value into other primitive types or vice versa and observe the results



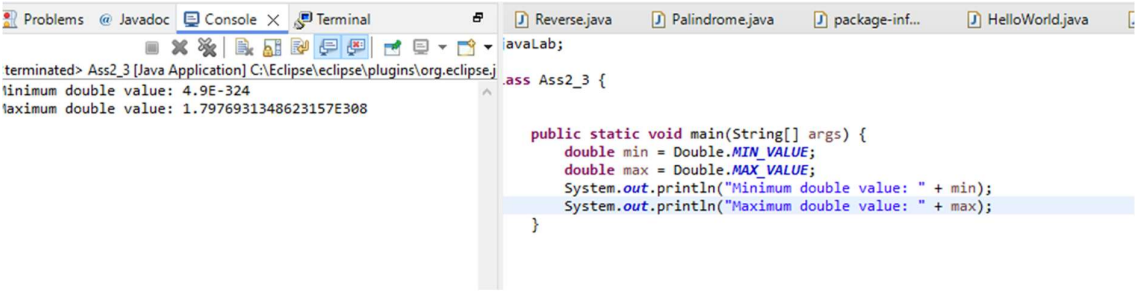
7. Working with java.lang.Double

a. Explore the Java API documentation for java.lang.Double and observe its modifiers and super types.

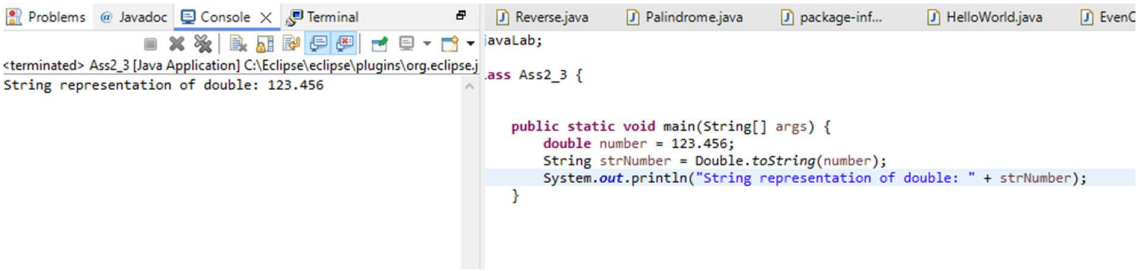
b. Write a program to test how many bytes are used to represent a double value using the BYTES field. (Hint: Use Double.BYTES).



c. Write a program to find the minimum and maximum values of double using the MIN_VALUE and MAX_VALUE fields. (Hint: Use Double.MIN_VALUE and Double.MAX_VALUE).

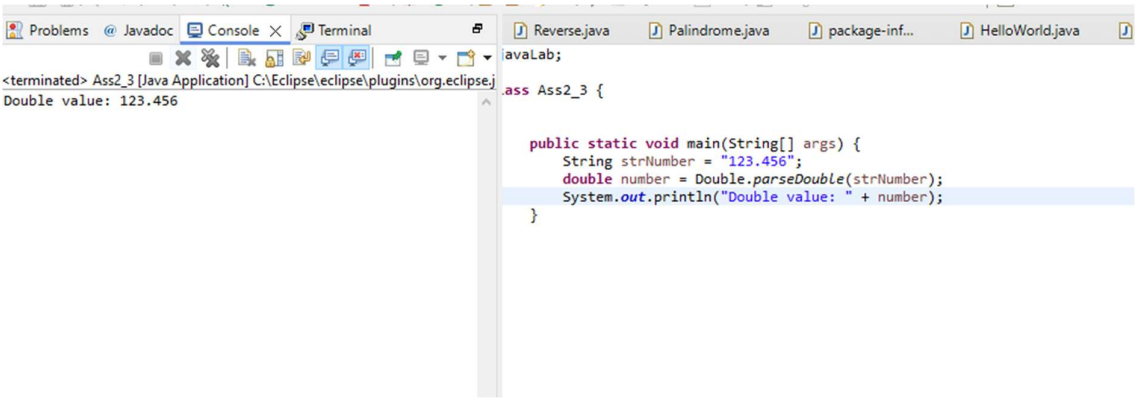


d. Declare a method-local variable number of type double with some value and convert it to a String using the toString method. (Hint: Use Double.toString(double)).

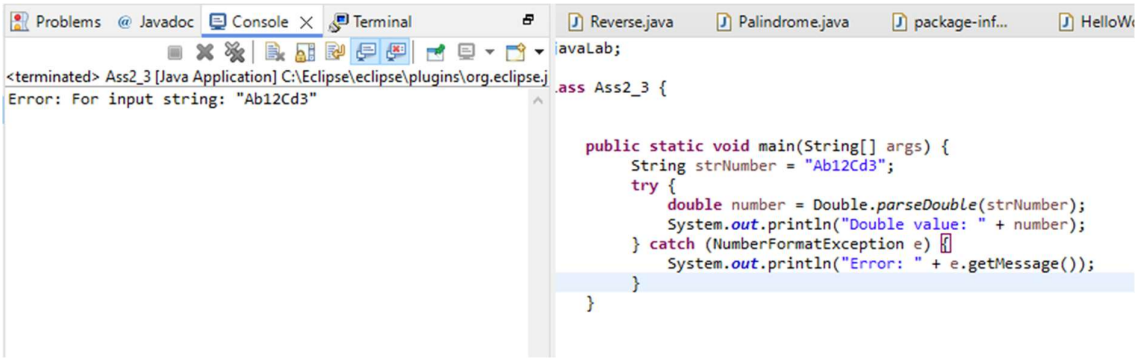


e. Declare a method-local variable strNumber of type String with some value and convert it to a

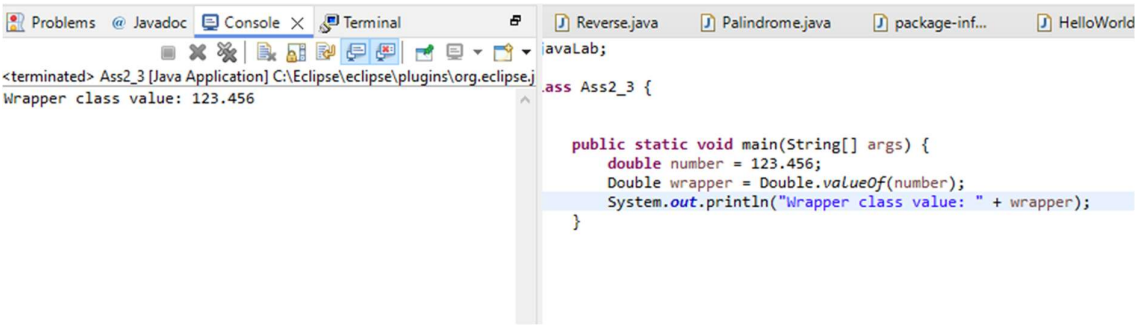
double value using the parseDouble method. (Hint: Use Double.parseDouble(String)).



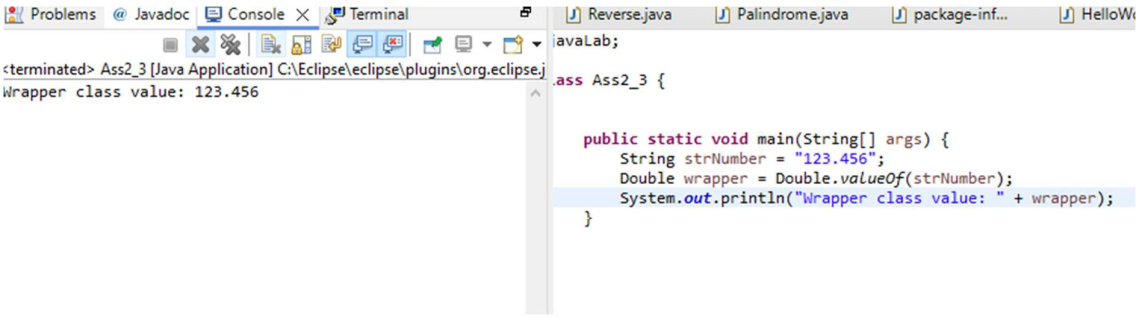
f. Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to a double value. (Hint: parseDouble method will throw a NumberFormatException).



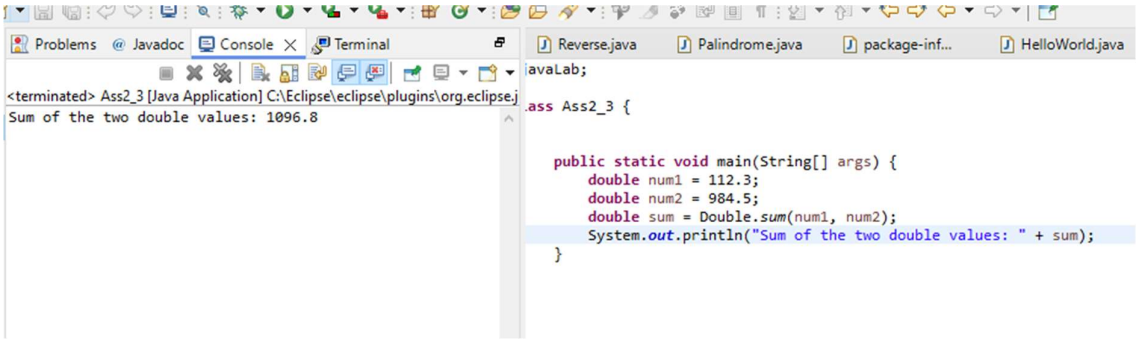
g. Declare a method-local variable number of type double with some value and convert it to the corresponding wrapper class using Double.valueOf(). (Hint: Use Double.valueOf(double)).



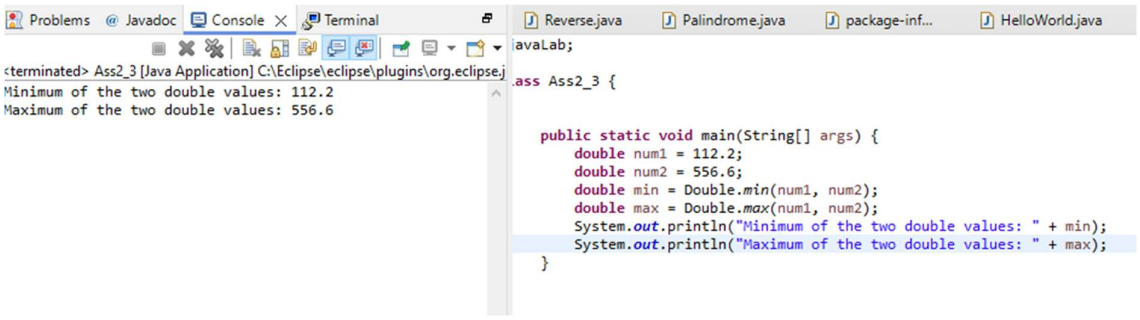
h. Declare a method-local variable strNumber of type String with some double value and convert it to the corresponding wrapper class using Double.valueOf(). (Hint: Use Double.valueOf(String)).



i. Declare two double variables with values 112.3 and 984.5, and add them using a method from the Double class. (Hint: Use Double.sum(double, double)).

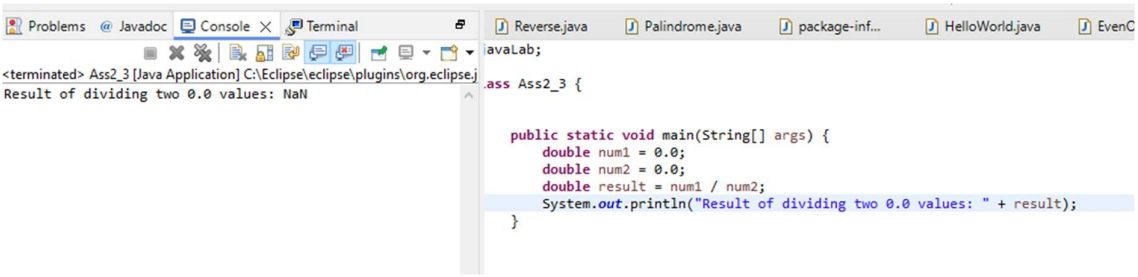


j. Declare two double variables with values and maximum values using the Double 112.2 and class. (Hint: Use , and find the minimum 556.6 Double.min(double, double) and Double.max(double, double)

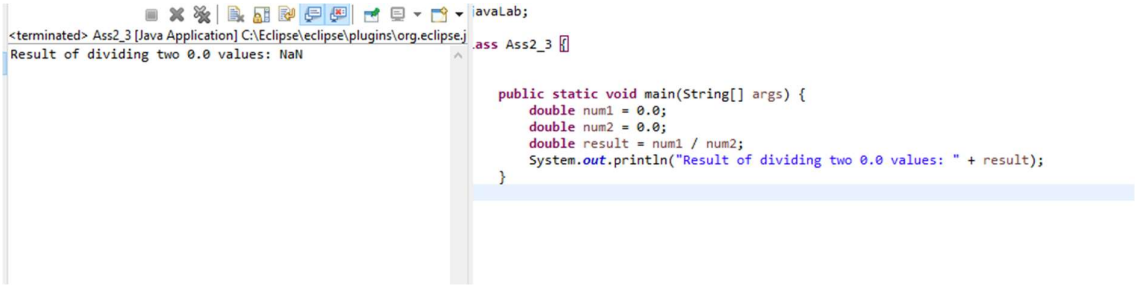


k. Declare a double variable with the value). -25.0 (Hint: Use Math.sqrt() method). . Find the square root of this value.

l. Declare two double variables with the same value, 0.0 , and divide them. (Hint: Observe the result and any special floating-point behavior).



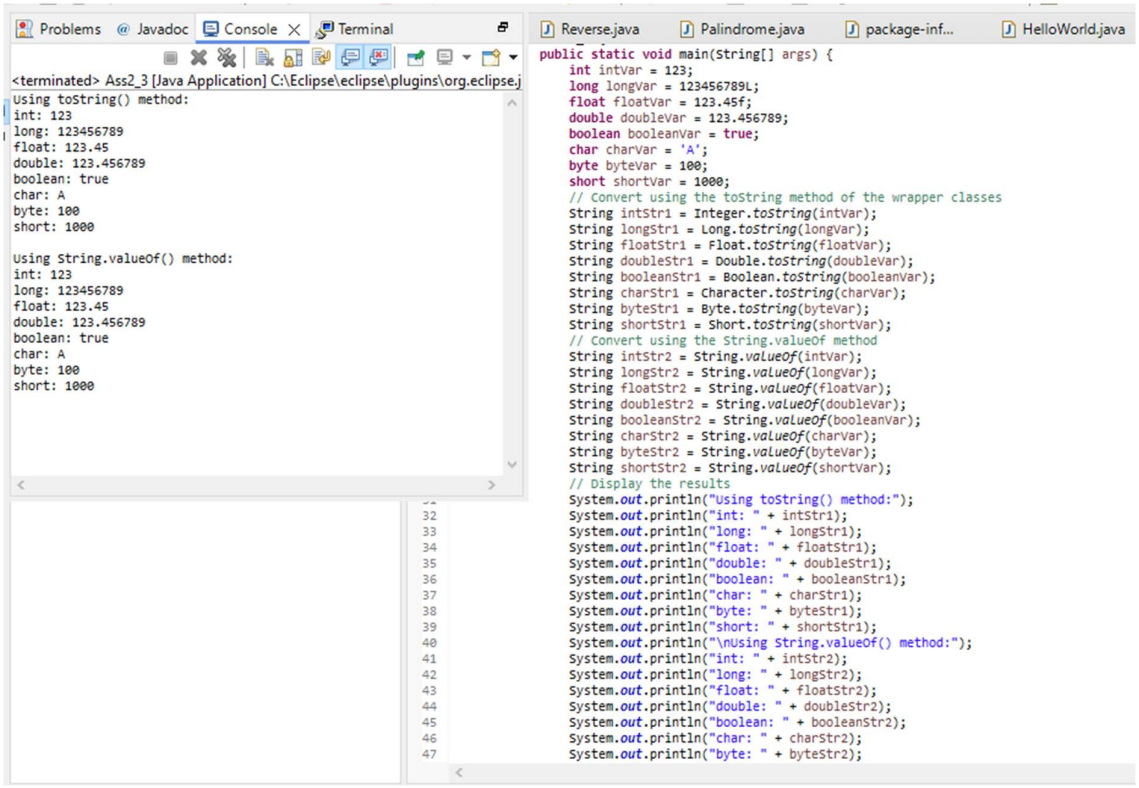
m. Experiment with converting a versa and observe the results. double value into other primitive types or vice



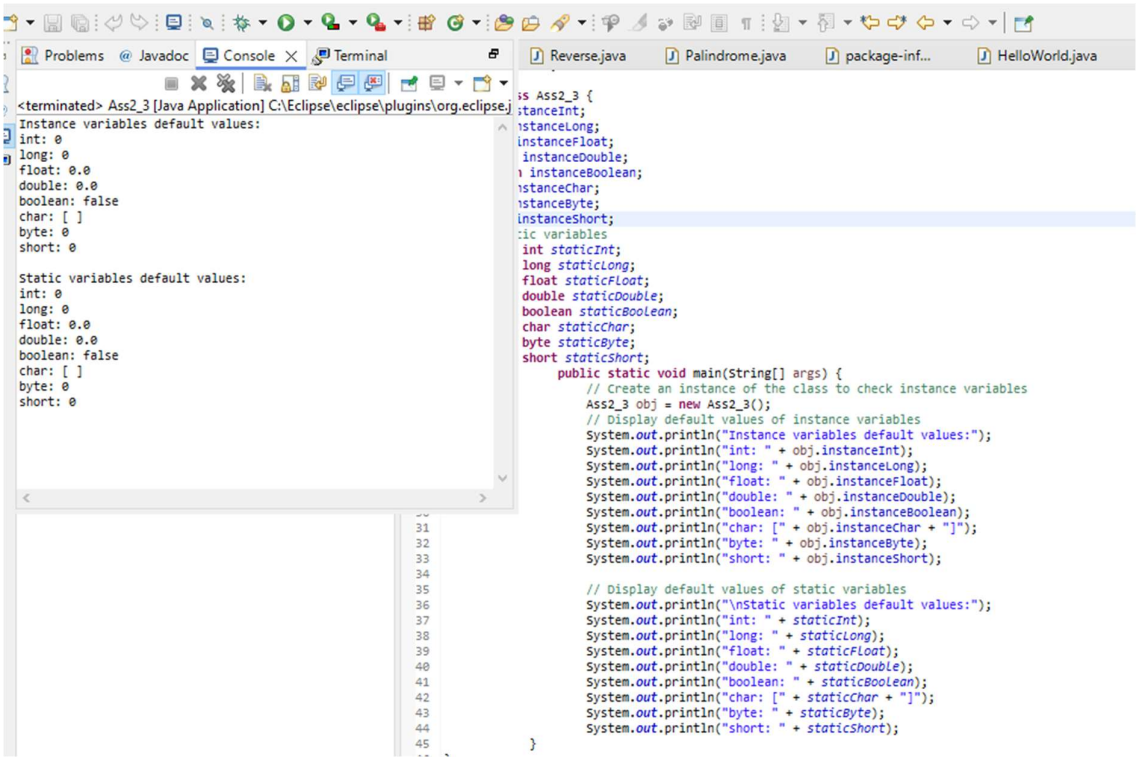
8. Conversion between Primitive Types and Strings String : Initialize a variable of each primitive type with a user-defined value and convert it into

o First, use the toString Integer.toString()).

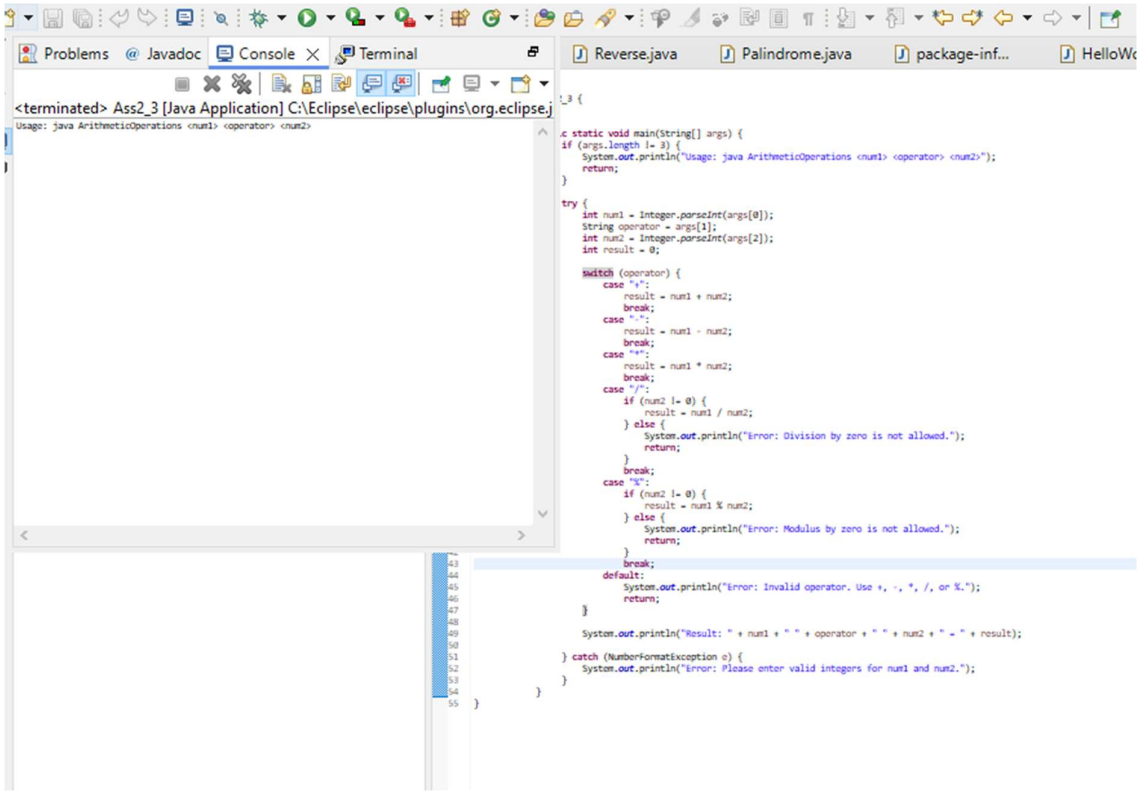
Then, use the valueOf method of the corresponding wrapper class. (e.g., method of the String.valueOf()).



9. Default Values of Primitive Types String class. (e.g., Declare variables of each primitive type as fields of a class and check their default values. (Note: Default values depend on whether the variables are instance variables or static variables).



10. Arithmetic Operations with Command Line Input Write a program that accepts two integers and an arithmetic operator ( ) from the command line. Perform the specified arithmetic operation based on the operator provided. (Hint: Use + switch-case for operations).


Problems
Javadoc
Console
Terminal
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.j
Usage: java ArithmeticOperations <num1> <operator> <num2>
Reverse.java
Palindrome.java
package-inf...
HelloW
_3 {
c static void main(String[] args) {
if (args.length != 3) {
System.out.println("Usage: java ArithmeticOperations <num1> <operator> <num2>");
return;
}
try {
int num1 = Integer.parseInt(args[0]);
String operator = args[1];
int num2 = Integer.parseInt(args[2]);
int result = 0;
switch (operator) {
case "+":
result = num1 + num2;
break;
case "-":
result = num1 - num2;
break;
case "*":
result = num1 * num2;
break;
case "/":
if (num2 != 0) {
result = num1 / num2;
} else {
System.out.println("Error: Division by zero is not allowed.");
return;
}
break;
case "%":
if (num2 != 0) {
result = num1 % num2;
} else {
System.out.println("Error: Modulus by zero is not allowed.");
return;
}
break;
default:
System.out.println("Error: Invalid operator. Use +, -, *, /, or %.");
return;
}
System.out.println("Result: " + num1 + " " + operator + " " + num2 + " = " + result);
} catch (NumberFormatException e) {
System.out.println("Error: Please enter valid integers for num1 and num2.");
}
}
}