

# Assignment-2(Sandeep sir )

## 1. Working with java.lang.Boolean

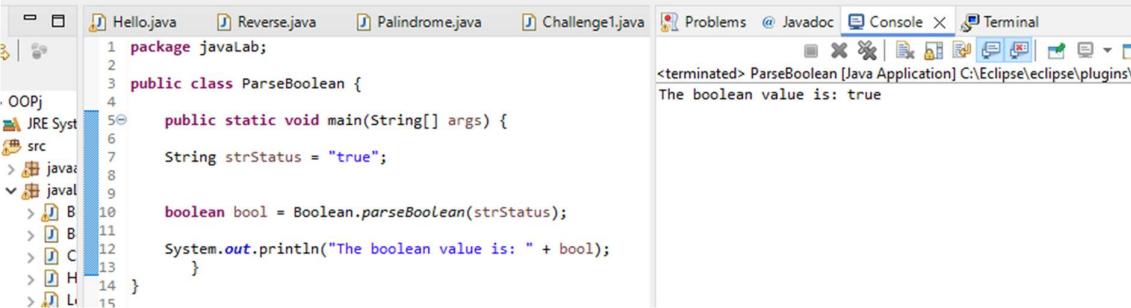
- Explore the Java API documentation for `java.lang.Boolean` and observe its modifiers and super types.

Field Summary	
Fields	Modifier and Type
	Field and Description
	<code>static Boolean FALSE</code>
	The Boolean object corresponding to the primitive value false.
	<code>static Boolean TRUE</code>
	The Boolean object corresponding to the primitive value true.
	<code>static Class&lt;Boolean&gt; TYPE</code>
	The Class object representing the primitive type boolean.

- 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)` ).

```
File Edit Source Refactor Source Navigate Project Run Window Help
P X  Hello.java Reverse.java Palindrome.java package-info.java Problems Javadoc Console Terminal
eclipse-workspace - OOPj/src/javaLab/BooleanToString.java - Eclipse IDE
Hello.java
public class BooleanToString {
    public static void main(String[] args) {
        boolean status = false;
        String str1 = Boolean.toString(status);
        System.out.println("Using Boolean.toString(): " + str1);
    }
}
<terminated> BooleanToString [Java Application] C:\Eclipse\plugins\@ Using Boolean.toString(): false
```

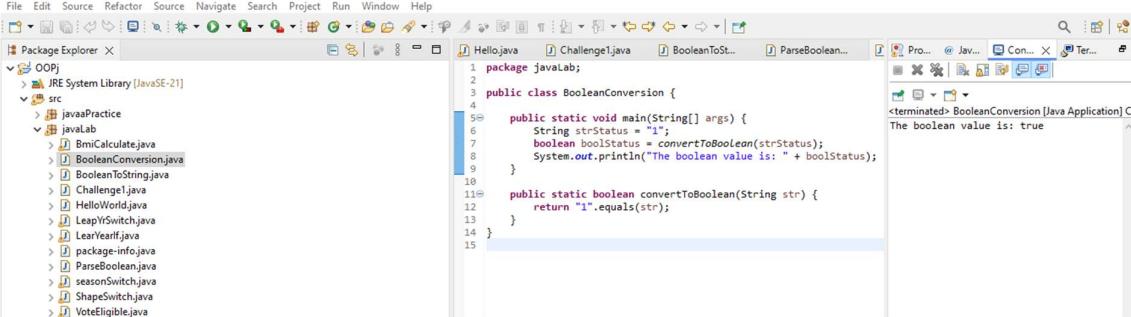
c. Declare a method-local variable strStatus of type String with the value "true" and convert it to a boolean using the parseBoolean method. (Hint: Use Boolean.parseBoolean(String)).



```
1 package javaLab;
2
3 public class ParseBoolean {
4
5     public static void main(String[] args) {
6
7         String strStatus = "true";
8
9         boolean bool = Boolean.parseBoolean(strStatus);
10
11         System.out.println("The boolean value is: " + bool);
12     }
13
14 }
15
```

The boolean value is: true

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").



```
1 package javaLab;
2
3 public class BooleanConversion {
4
5     public static void main(String[] args) {
6         String strStatus = "1";
7         boolean boolStatus = convertToBoolean(strStatus);
8         System.out.println("The boolean value is: " + boolStatus);
9     }
10
11     public static boolean convertToBoolean(String str) {
12         return "1".equals(str);
13     }
14 }
15
```

The boolean value is: true

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)).

The screenshot shows the Eclipse IDE interface. On the left is the Package Explorer view, which lists several Java files under the OOP project. In the center is the code editor showing BooleanWrapper.java:

```
1 package javaLab;
2
3 public class BooleanWrapper {
4
5     public static void main(String[] args) {
6
7         boolean status = true;
8         Boolean statusWr = Boolean.valueOf(status);
9         System.out.println("The Boolean object is: " + statusWr);
10    }
11
12 }
13
14 }
15 }
```

On the right is the Run View, which displays the output of the application: "The Boolean object is: true".

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)).

The screenshot shows the Eclipse IDE interface. On the left is the Package Explorer view, which lists several Java files under the OOP project. In the center is the code editor showing WrapperString.java:

```
1 package javaLab;
2
3 public class WrapperString {
4
5     public static void main(String[] args) {
6         String strStatus = "true";
7         Boolean Status = Boolean.valueOf(strStatus);
8         System.out.println("The Boolean object is: " + Status);
9     }
10 }
11
```

On the right is the Run View, which displays the output of the application: "The Boolean object is: true".

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



The screenshot shows the Eclipse IDE interface. In the center is a code editor with the following Java code:

```
1 package javaLab;
2
3 public class SizeByte {
4     public static void main(String[] args) {
5         int numberOfBytes = Byte.BYTES;
6         System.out.println(" byte value is: " + numberOfBytes);
7     }
8 }
9 
```

To the right of the code editor is a terminal window showing the output of the program:

```
<terminated> SizeByte [Java Application]
byte value is: 1
```

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).

The screenshot shows the Eclipse IDE interface. In the center is a code editor with the following Java code:

```
1 package javaLab;
2
3 public class StringByte {
4     public static void main(String[] args) {
5         byte number = 42;
6         String numberAsString = Byte.toString(number);
7         System.out.println("The byte value as a String is: " + numberAsString);
8     }
9 }
10
11
12 }
13 
```

To the right of the code editor is a terminal window showing the output of the program:

```
<terminated> StringByte [Java Application] C:\Eclips
The byte value as a String is: 42
```

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)).

```
Source Navigate Search Project Run Window Help
ParseBoolean... BooleanConv... BooleanWrap... WrapperStrin... PrimitiveBo...
/aSE-21]
/a ion.java ijava rjava
/a

1 package javaLab;
2
3 public class StringByte {
4
5     public static void main(String[] args) {
6         byte number = 42;
7         String numberAsString = Byte.toString(number);
8         System.out.println("The byte value as a String is: " + numberAsString);
9     }
10 }
11
12 }

/a
```

The byte value as a String is: 42

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)).

```
Source Navigate Search Project Run Window Help
BooleanConv... BooleanWrap... WrapperStrin... PrimitiveBo...
/aSE-21]
/a ion.java ijava rjava
/a

1 package javaLab;
2
3 public class ByteString {
4
5     public static void main(String[] args) {
6         String strNumber = "50";
7         byte number = Byte.parseByte(strNumber);
8         System.out.println("The String value as a byte is: " + number);
9     }
10 }
11
12 
```

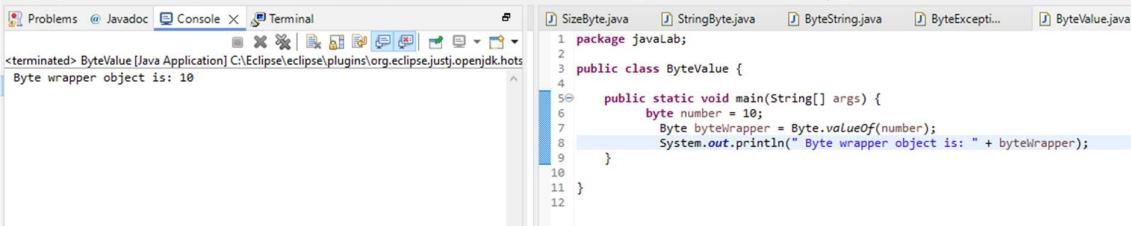
The String value as a byte is: 50

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).

```
File Edit Source Refactor Source Navigate Project Run Window Help
File Edit Source Refactor Source Navigate Project Run Window Help
Problems @ Javadoc Console Terminal
<terminated> ByteException [Java Application] C:\Eclipse\workspace\javaLab\ByteException.java:1: error: cannot find symbol
1 package javaLab;
^
symbol:   class ByteException
location: class ByteException
2
3 public class ByteException {
4
5     public static void main(String[] args) throws NumberFormatException {
6         String strNumber = "Ab12Cd3";
7         byte number = Byte.parseByte(strNumber);
8         System.out.println("The String value as a byte is: " + number);
9     }
10 }
11
12 
```

Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3"  
at java.base/java.lang.NumberFormatException.forInputString(Numb  
at java.base/java.lang.Integer.parseInt(Integer.java:662)  
at java.base/java.lang.Byte.parseDouble(Byte.java:195)  
at java.base/java.lang.Byte.parseDouble(Byte.java:221)  
at OOPj/javaLab/ByteException.main(ByteException.java:7)

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)).



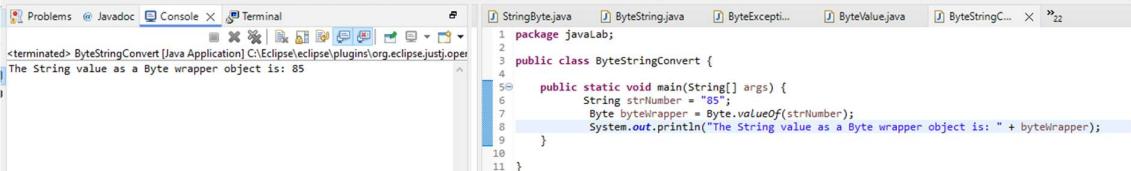
```

package javaLab;
public class ByteValue {
    public static void main(String[] args) {
        byte number = 10;
        Byte byteWrapper = Byte.valueOf(number);
        System.out.println(" Byte wrapper object is: " + byteWrapper);
    }
}

```

The console output shows: Byte wrapper object is: 10

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))



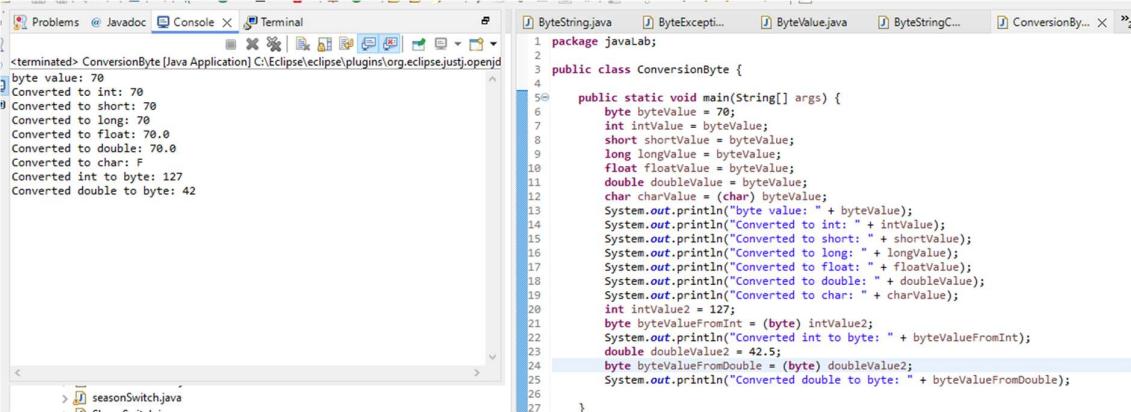
```

package javaLab;
public class ByteStringConvert {
    public static void main(String[] args) {
        String strNumber = "85";
        Byte byteWrapper = Byte.valueOf(strNumber);
        System.out.println("The String value as a Byte wrapper object is: " + byteWrapper);
    }
}

```

The console output shows: The String value as a Byte wrapper object is: 85

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



```

package javaLab;
public class ConversionByte {
    public static void main(String[] args) {
        byte byteValue = 70;
        int intValue = byteValue;
        short shortValue = byteValue;
        long longValue = byteValue;
        float floatValue = byteValue;
        double doubleValue = byteValue;
        char charValue = (char) byteValue;
        System.out.println("byte value: " + byteValue);
        System.out.println("Converted to int: " + intValue);
        System.out.println("Converted to short: " + shortValue);
        System.out.println("Converted to long: " + longValue);
        System.out.println("Converted to float: " + floatValue);
        System.out.println("Converted to double: " + doubleValue);
        System.out.println("Converted to char: " + charValue);
        int intValue2 = 127;
        byte byteValueFromInt = (byte) intValue2;
        System.out.println("Converted int to byte: " + byteValueFromInt);
        double doubleValue2 = 42.5;
        byte byteValueFromDouble = (byte) doubleValue2;
        System.out.println("Converted double to byte: " + byteValueFromDouble);
    }
}

```

The console output shows:

```

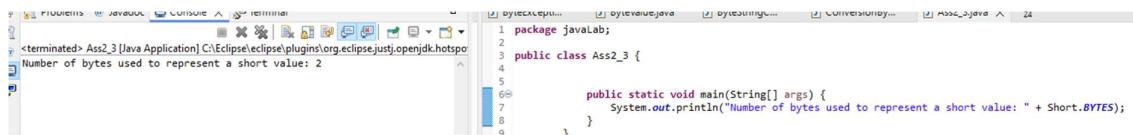
byte value: 70
Converted to int: 70
Converted to short: 70
Converted to long: 70
Converted to float: 70.0
Converted to double: 70.0
Converted to char: F
Converted int to byte: 127
Converted double to byte: 42

```

### 3. Working with java.lang.Short

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

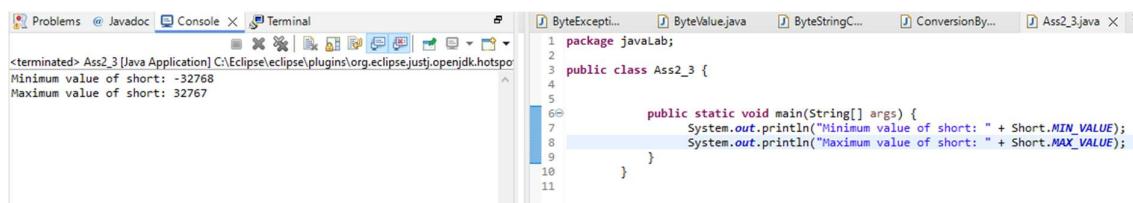
b. Write a program to test how many bytes are used to represent a short value using the `BYTES` field. (Hint: Use `Short.BYTES`).



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\plugins\org.eclipse.jdt.openjdk.hotspot
Number of bytes used to represent a short value: 2
```

```
1 package javaLab;
2
3 public class Ass2_3 {
4
5     public static void main(String[] args) {
6         System.out.println("Number of bytes used to represent a short value: " + Short.BYTES);
7     }
8
9
10}
```

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`).



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\plugins\org.eclipse.jdt.openjdk.hotspot
Minimum value of short: -32768
Maximum value of short: 32767
```

```
1 package javaLab;
2
3 public class Ass2_3 {
4
5
6     public static void main(String[] args) {
7         System.out.println("Minimum value of short: " + Short.MIN_VALUE);
8         System.out.println("Maximum value of short: " + Short.MAX_VALUE);
9     }
10}
```

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)`).

```
ByteException.java ByteValue.java ByteStringC... ConversionBy... Ass2_3.java X 24
1 package javaLab;
2
3 public class Ass2_3 {
4
5     public static void main(String[] args) {
6         short number = 12345;
7         String numberAsString = Short.toString(number);
8         System.out.println("The short value as a String is: " + numberAsString);
9     }
10
11 }
12
```

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)).

```
ByteException.java ByteValue.java ByteStringC... ConversionBy... Ass2_3.java X 24
1 package javaLab;
2
3 public class Ass2_3 {
4
5     public static void main(String[] args) {
6         String strNumber = "12345";
7         short number = Short.parseShort(strNumber);
8         System.out.println("The String value as a short is: " + number);
9     }
10
11 }
12
```

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).

```
ByteException.java ByteValue.java ByteStringC... ConversionBy... Ass2_3.java X 24
1 package javaLab;
2
3 public class Ass2_3 {
4
5     public static void main(String[] args) {
6         String strNumber = "Ab12Cd3";
7         try {
8             short number = Short.parseShort(strNumber);
9             System.out.println("The String value as a short is: " + number);
10        } catch (NumberFormatException e) {
11            System.out.println("Error: The string '" + strNumber + "' is not a valid short value");
12        }
13    }
14
15 }
16
```

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)).

```

package javaLab;
public class Ass2_3 {
    public static void main(String[] args) {
        short number = 12345;
        Short shortWrapper = Short.valueOf(number);
        System.out.println("The short value as a Short wrapper object is: " + shortWrapper);
    }
}
  
```

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)).

```

package javaLab;
public class Ass2_3 {
    public static void main(String[] args) {
        String strNumber = "12345";
        Short shortWrapper = Short.valueOf(strNumber);
        System.out.println("The String value as a Short wrapper object is: " + shortWrapper);
    }
}
  
```

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

```

package javaLab;
public class Ass2_3 {
    public static void main(String[] args) {
        short shortValue = 32767;
        int intValue = shortValue;
        long longValue = shortValue;
        float floatValue = shortValue;
        double doubleValue = shortValue;
        char charValue = (char) shortValue;

        System.out.println("short value: " + shortValue);
        System.out.println("Converted to int: " + intValue);
        System.out.println("Converted to long: " + longValue);
        System.out.println("Converted to float: " + floatValue);
        System.out.println("Converted to double: " + doubleValue);
        System.out.println("Converted to char: " + charValue);

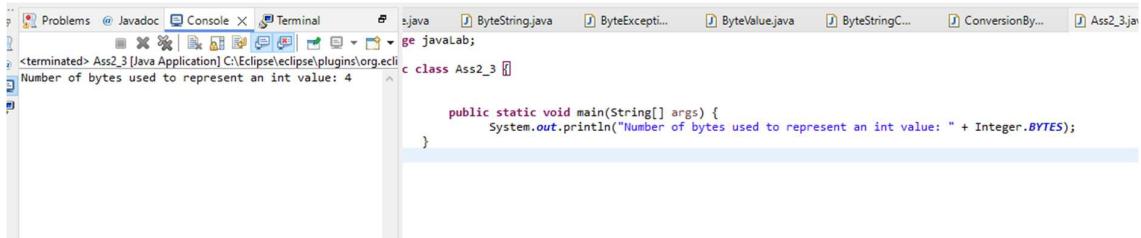
        int intValue2 = 12345;
        short shortValueFromInt = (short) intValue2;
        System.out.println("Converted int to short: " + shortValueFromInt);

        double doubleValue2 = 123.45;
        short shortValueFromDouble = (short) doubleValue2;
        System.out.println("Converted double to short: " + shortValueFromDouble);
    }
}
  
```

## 4. Working with java.lang.Integer

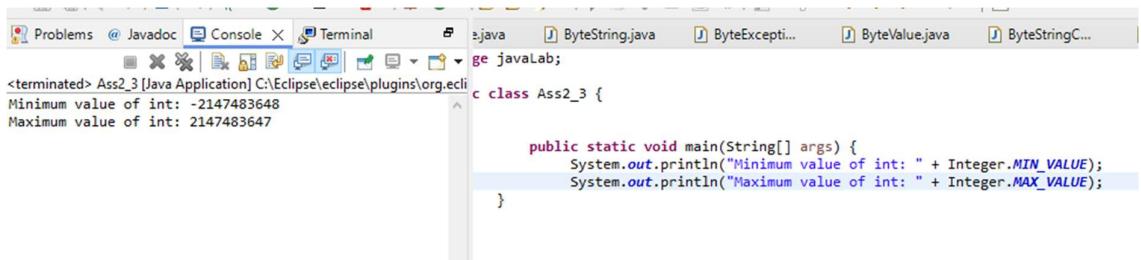
a. Explore the Java API documentation for java.lang.Integer and observe its modifiers and super types.

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



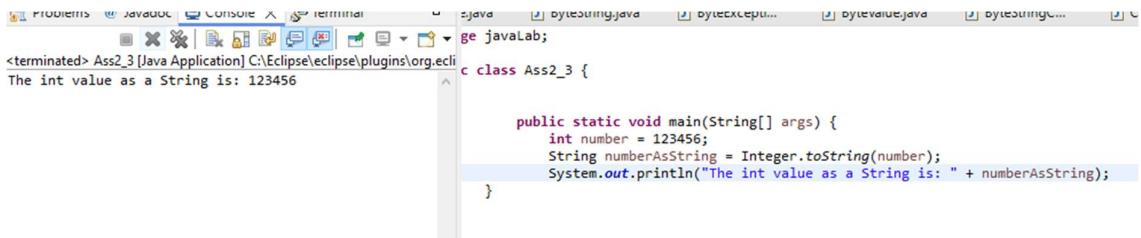
```
Problems @ Javadoc Console Terminal sjava ByteString.java ByteExcepti... ByteValue.java ByteStringC... ConversionBy... Ass2_3.jar  
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\javaLab;  
c class Ass2_3 {  
  
    public static void main(String[] args) {  
        System.out.println("Number of bytes used to represent an int value: " + 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).



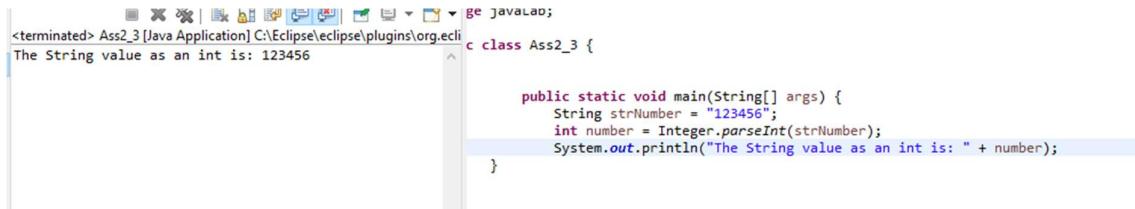
```
Problems @ Javadoc Console Terminal sjava ByteString.java ByteExcepti... ByteValue.java ByteStringC... Ass2_3.jar  
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\javaLab;  
Minimum value of int: -2147483648  
Maximum value of int: 2147483647  
c class Ass2_3 {  
  
    public static void main(String[] args) {  
        System.out.println("Minimum value of int: " + Integer.MIN_VALUE);  
        System.out.println("Maximum value of int: " + 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)`).



```
Problems @ Javadoc Console Terminal sjava ByteString.java ByteExcepti... ByteValue.java ByteStringC... Ass2_3.jar  
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\javaLab;  
The int value as a String is: 123456  
c class Ass2_3 {  
  
    public static void main(String[] args) {  
        int number = 123456;  
        String numberAsString = Integer.toString(number);  
        System.out.println("The int value as a String is: " + numberAsString);  
    }  
}
```

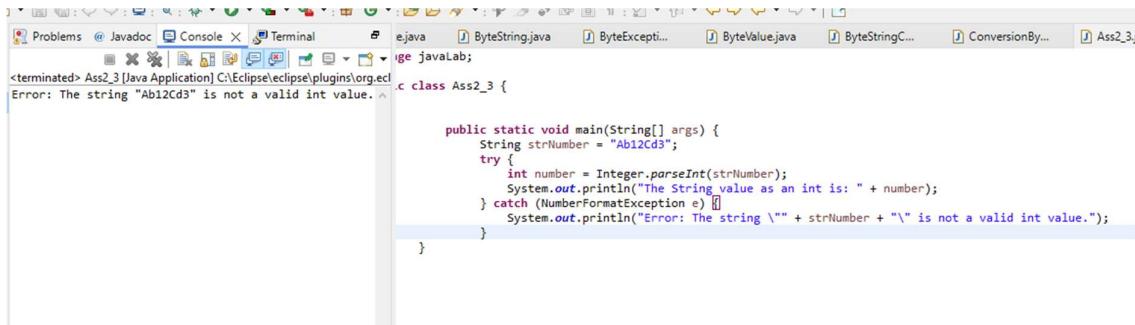
e. Declare a method-local variable strNumber of type String with some value and convert it to an int value using the parseInt method. (Hint: Use Integer.parseInt(String)).



The screenshot shows the Eclipse IDE interface with the Java perspective selected. A terminal window at the top displays the output of a run: "The String value as an int is: 123456". Below the terminal is the Java code for a class named Ass2\_3:

```
class Ass2_3 {
    public static void main(String[] args) {
        String strNumber = "123456";
        int number = Integer.parseInt(strNumber);
        System.out.println("The String value as an int is: " + number);
    }
}
```

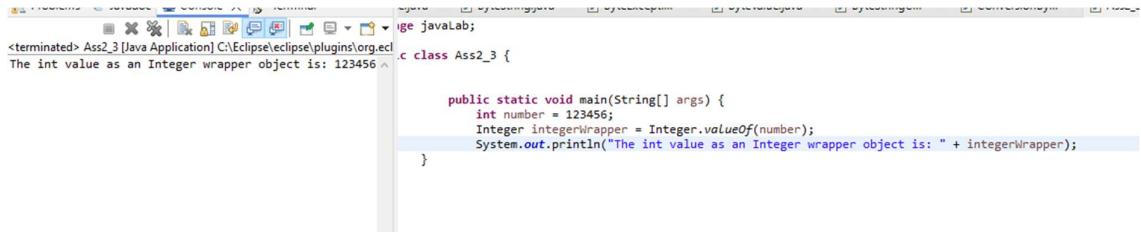
f. Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to an int value. (Hint: parseInt method will throw a NumberFormatException).



The screenshot shows the Eclipse IDE interface with the Java perspective selected. A terminal window at the top displays the error message: "Error: The string \"Ab12Cd3\" is not a valid int value.". Below the terminal is the Java code for a class named Ass2\_3, which includes a try-catch block to handle the exception:

```
class Ass2_3 {
    public static void main(String[] args) {
        String strNumber = "Ab12Cd3";
        try {
            int number = Integer.parseInt(strNumber);
            System.out.println("The String value as an int is: " + number);
        } catch (NumberFormatException e) {
            System.out.println("Error: The string \"" + strNumber + "\" is not a valid int value.");
        }
    }
}
```

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)).

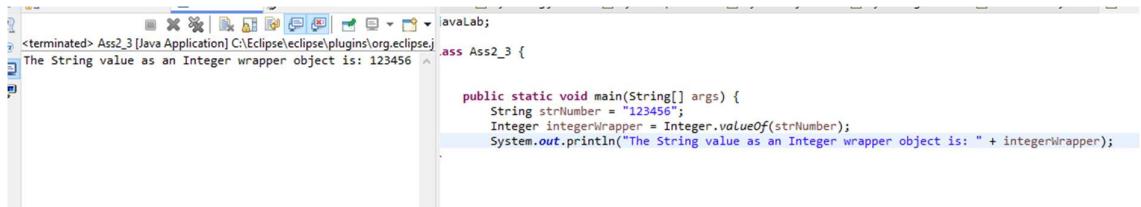


```
javaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\Ass2_3.java
The int value as an Integer wrapper object is: 123456

class Ass2_3 {

    public static void main(String[] args) {
        int number = 123456;
        Integer integerWrapper = Integer.valueOf(number);
        System.out.println("The int value as an Integer wrapper object is: " + integerWrapper);
    }
}
```

- 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)).

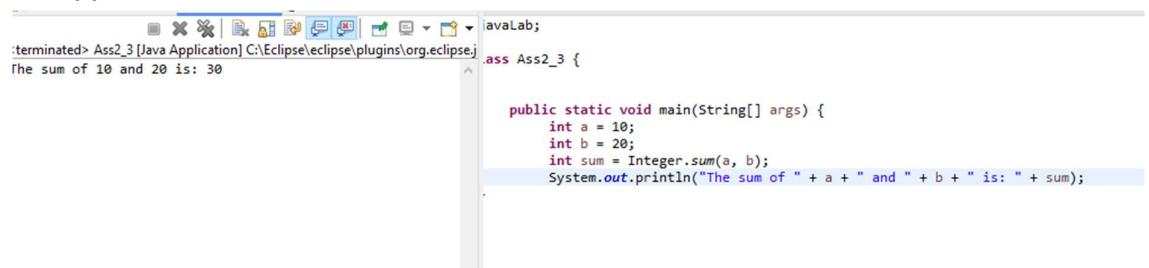


```
javaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\Ass2_3.java
The String value as an Integer wrapper object is: 123456

class Ass2_3 {

    public static void main(String[] args) {
        String strNumber = "123456";
        Integer integerWrapper = Integer.valueOf(strNumber);
        System.out.println("The String value as an Integer wrapper object is: " + integerWrapper);
    }
}
```

- 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)).

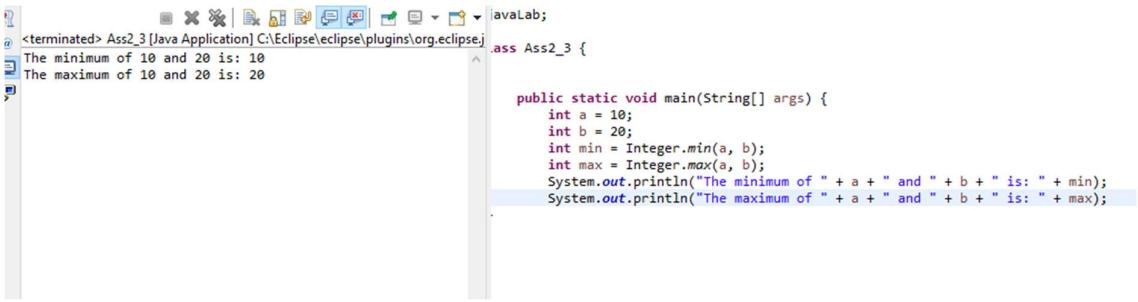


```
javaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\Ass2_3.java
The sum of 10 and 20 is: 30

class Ass2_3 {

    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        int sum = Integer.sum(a, b);
        System.out.println("The sum of " + a + " and " + b + " is: " + sum);
    }
}
```

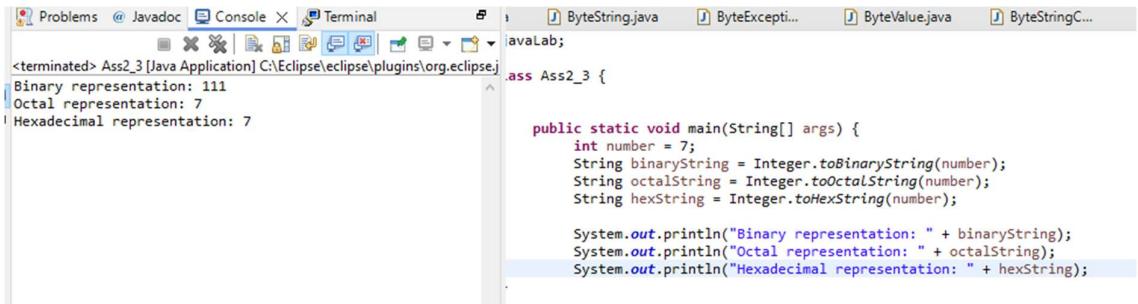
- j. Declare two integer variables with values 10 and 20, and find the minimum and maximum values using the Integer class. (Hint: Use Integer.min(int, int) and Integer.max(int, int)).



```
avaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.j... .ass Ass2_3 {
The minimum of 10 and 20 is: 10
The maximum of 10 and 20 is: 20
}

public static void main(String[] args) {
    int a = 10;
    int b = 20;
    int min = Integer.min(a, b);
    int max = Integer.max(a, b);
    System.out.println("The minimum of " + a + " and " + b + " is: " + min);
    System.out.println("The maximum of " + a + " and " + b + " is: " + max);
```

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)).



```
Problems @ Javadoc Console X Terminal
avaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.j... .ass Ass2_3 {
Binary representation: 111
Octal representation: 7
Hexadecimal representation: 7

public static void main(String[] args) {
    int number = 7;
    String binaryString = Integer.toBinaryString(number);
    String octalString = Integer.toOctalString(number);
    String hexString = Integer.toHexString(number);

    System.out.println("Binary representation: " + binaryString);
    System.out.println("Octal representation: " + octalString);
    System.out.println("Hexadecimal representation: " + hexString);
```

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

```
int value: 123456
Converted to long: 123456
Converted to float: 123456.0
Converted to double: 123456.0
Converted to short: -7616
Converted to byte: 64
Converted to char: ☺
```

```
avaLab;
class Ass2_3 {
    public static void main(String[] args) {
        int intValue = 123456;

        long longValue = intValue;
        float floatValue = intValue;
        double doubleValue = intValue;
        short shortValue = (short) intValue;
        byte byteValue = (byte) intValue;
        char charValue = (char) intValue;

        System.out.println("int value: " + intValue);
        System.out.println("Converted to long: " + longValue);
        System.out.println("Converted to float: " + floatValue);
        System.out.println("Converted to double: " + doubleValue);
        System.out.println("Converted to short: " + shortValue);
        System.out.println("Converted to byte: " + byteValue);
        System.out.println("Converted to char: " + charValue);
    }
}
```

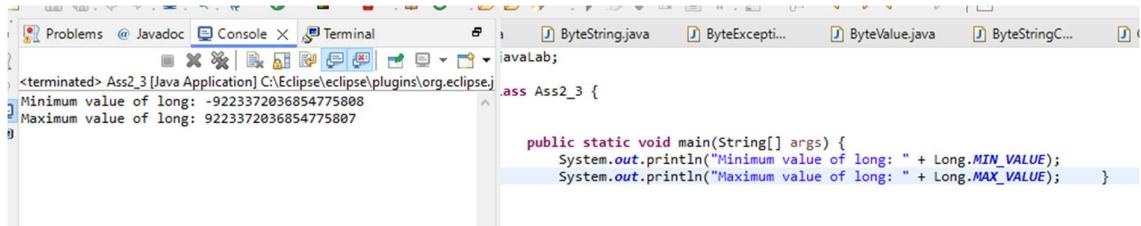
## 5. Working with java.lang.Long

- Explore the Java API documentation for `java.lang.Long` and observe its modifiers and super types.
- Write a program to test how many bytes are used to represent a long value using the `BYTES` field. (Hint: Use `Long.BYTES`).

```
Number of bytes used to represent a long value: 8
```

```
avaLab;
class Ass2_3 {
    public static void main(String[] args) {
        System.out.println("Number of bytes used to represent a long value: " + Long.BYTES);
    }
}
```

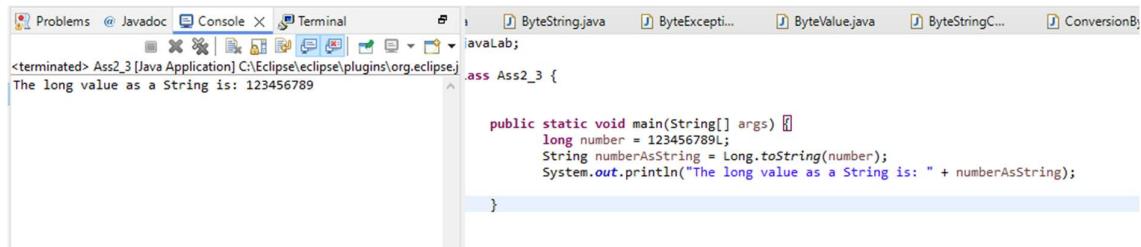
- Write a program to find the minimum and maximum values of long using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Long.MIN_VALUE` and `Long.MAX_VALUE`).



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\javaLab;
Minimum value of long: -9223372036854775808
Maximum value of long: 9223372036854775807
```

```
public static void main(String[] args) {
    System.out.println("Minimum value of long: " + Long.MIN_VALUE);
    System.out.println("Maximum value of long: " + Long.MAX_VALUE); }
```

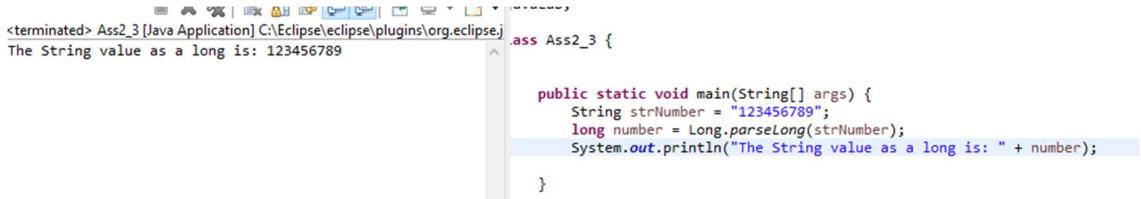
d. Declare a method-local variable number of type long with some value and convert it to a String using the `toString` method. (Hint: Use `Long.toString(long)`).



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\javaLab;
The long value as a String is: 123456789
```

```
public static void main(String[] args) {
    long number = 123456789L;
    String numberAsString = Long.toString(number);
    System.out.println("The long value as a String is: " + numberAsString);
}
```

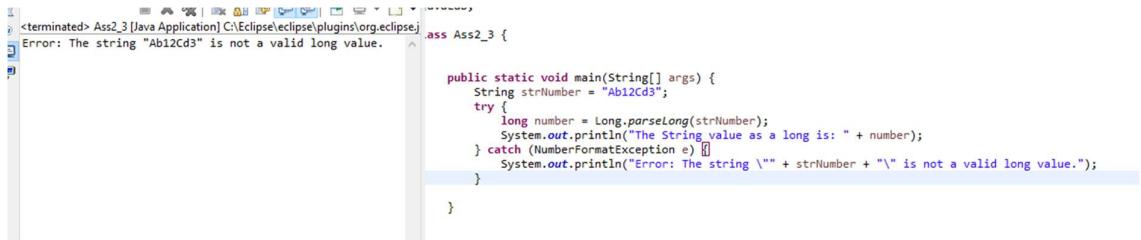
e. Declare a method-local variable `strNumber` of type String with some value and convert it to a long value using the `parseLong` method. (Hint: Use `Long.parseLong(String)`).



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\javaLab;
The String value as a long is: 123456789
```

```
public static void main(String[] args) {
    String strNumber = "123456789";
    long number = Long.parseLong(strNumber);
    System.out.println("The String value as a long is: " + number);
}
```

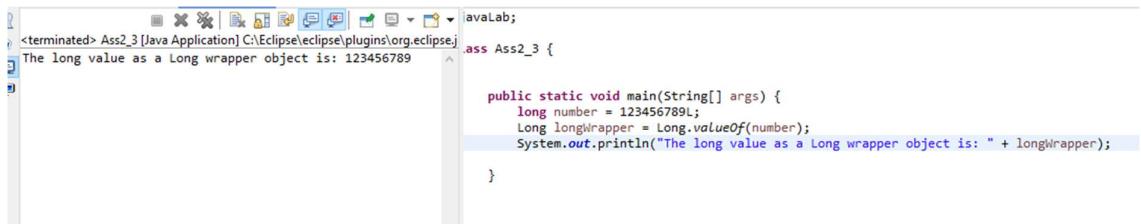
f. Declare a method-local variable `strNumber` of type String with the value "Ab12Cd3" and attempt to convert it to a long value. (Hint: `parseLong` method will throw a `NumberFormatException`).



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\plugins\org.eclipse.jdt.core\src\javaLab\Ass2_3.java
Error: The string "Ab12Cd3" is not a valid long value.

public class Ass2_3 {
    public static void main(String[] args) {
        String strNumber = "Ab12Cd3";
        try {
            long number = Long.parseLong(strNumber);
            System.out.println("The String value as a long is: " + number);
        } catch (NumberFormatException e) {
            System.out.println("Error: The string '" + strNumber + "' is not a valid long value.");
        }
    }
}
```

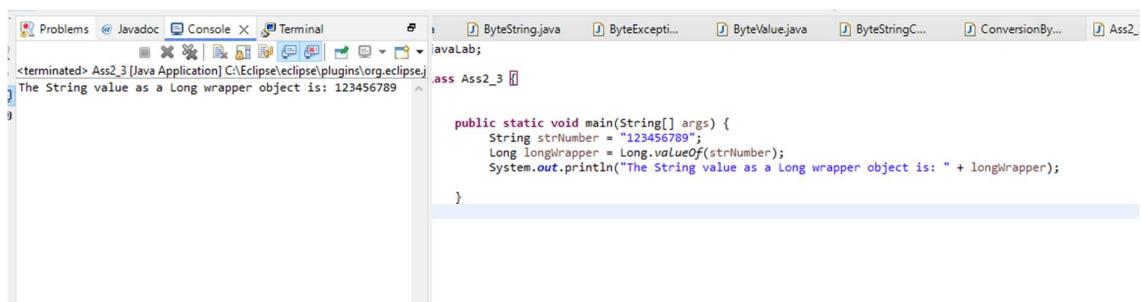
g. Declare a method-local variable number of type long with some value and convert it to the corresponding wrapper class using Long.valueOf(). (Hint: Use Long.valueOf(long)).



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\plugins\org.eclipse.jdt.core\src\javaLab\Ass2_3.java
The long value as a Long wrapper object is: 123456789

public class Ass2_3 {
    public static void main(String[] args) {
        long number = 123456789L;
        Long longWrapper = Long.valueOf(number);
        System.out.println("The long value as a Long wrapper object is: " + longWrapper);
    }
}
```

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)).



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\plugins\org.eclipse.jdt.core\src\javaLab\Ass2_3.java
The String value as a Long wrapper object is: 123456789

public class Ass2_3 {
    public static void main(String[] args) {
        String strNumber = "123456789";
        Long longWrapper = Long.valueOf(strNumber);
        System.out.println("The String value as a Long wrapper object is: " + longWrapper);
    }
}
```

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)).

```
terminated> Ass2_3 [Java Application] C:\Eclipse\workspace\org.eclipse.jdt.core\src\javaLab\ass Ass2_3 {  
    public static void main(String[] args) {  
        long a = 1123L;  
        long b = 9845L;  
        long sum = Long.sum(a, b);  
        System.out.println("The sum of " + a + " and " + b + " is: " + sum);  
    }  
}
```

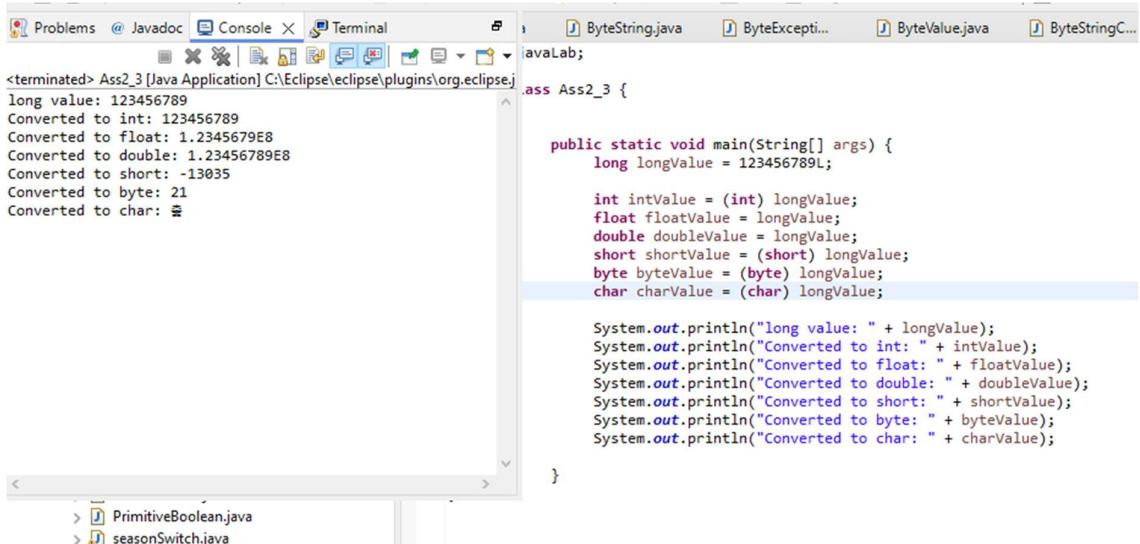
j. Declare two long variables with values 1122 and 5566, and find the minimum and maximum values using the Long class. (Hint: Use Long.min(long, long) and Long.max(long, long)).

```
terminated> Ass2_3 [Java Application] C:\Eclipse\workspace\org.eclipse.jdt.core\src\javaLab\ass Ass2_3 {  
    public static void main(String[] args) {  
        long a = 1122L;  
        long b = 5566L;  
        long min = Long.min(a, b);  
        long max = Long.max(a, b);  
        System.out.println("The minimum of " + a + " and " + b + " is: " + min);  
        System.out.println("The maximum of " + a + " and " + b + " is: " + max);  
    }  
}
```

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)).

```
terminated> Ass2_3 [Java Application] C:\Eclipse\workspace\org.eclipse.jdt.core\src\javaLab\ass Ass2_3 {  
    public static void main(String[] args) {  
        long number = 7L;  
        String binaryString = Long.toBinaryString(number);  
        String octalString = Long.toOctalString(number);  
        String hexString = Long.toHexString(number);  
  
        System.out.println("Binary representation: " + binaryString);  
        System.out.println("Octal representation: " + octalString);  
        System.out.println("Hexadecimal representation: " + hexString);  
    }  
}
```

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



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\plugins\org.eclipse.jdt.core\src\javaLab
long value: 123456789
Converted to int: 123456789
Converted to float: 1.2345679E8
Converted to double: 1.23456789E8
Converted to short: -13035
Converted to byte: 21
Converted to char: ☺

public static void main(String[] args) {
    long longValue = 123456789L;

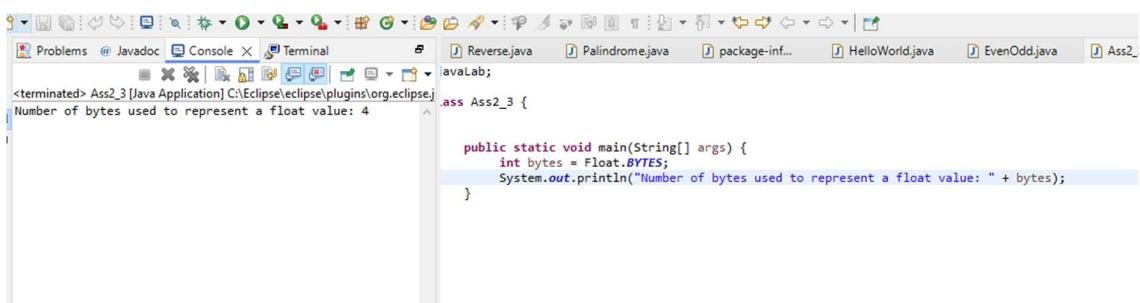
    int intValue = (int) longValue;
    float floatValue = longValue;
    double doubleValue = longValue;
    short shortValue = (short) longValue;
    byte byteValue = (byte) longValue;
    char charValue = (char) longValue;

    System.out.println("long value: " + longValue);
    System.out.println("Converted to int: " + intValue);
    System.out.println("Converted to float: " + floatValue);
    System.out.println("Converted to double: " + doubleValue);
    System.out.println("Converted to short: " + shortValue);
    System.out.println("Converted to byte: " + byteValue);
    System.out.println("Converted to char: " + charValue);
}
```

## 6. Working with java.lang.Float

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

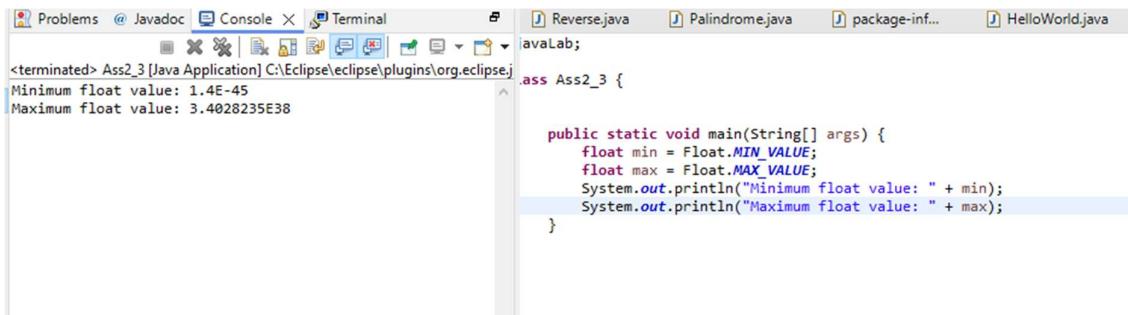
b. Write a program to test how many bytes are used to represent a float value using the `BYTES` field. (Hint: Use `Float.BYTES`).



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\plugins\org.eclipse.jdt.core\src\javaLab
Number of bytes used to represent a float value: 4

public static void main(String[] args) {
    int bytes = Float.BYTES;
    System.out.println("Number of bytes used to represent a float value: " + 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).

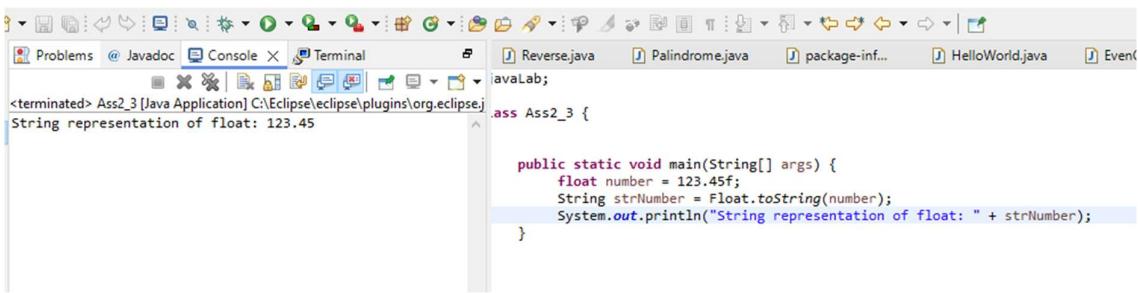


The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The output window displays the results of running the 'Ass2\_3' application, showing the minimum and maximum float values. The code editor shows the main method of the 'Ass2\_3' class, which prints these values using the `System.out.println` statement.

```
javaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\Ass2_3.java
Minimum float value: 1.4E-45
Maximum float value: 3.4028235E38

public static void main(String[] args) {
    float min = Float.MIN_VALUE;
    float max = Float.MAX_VALUE;
    System.out.println("Minimum float value: " + min);
    System.out.println("Maximum float value: " + max);
}
```

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)`).

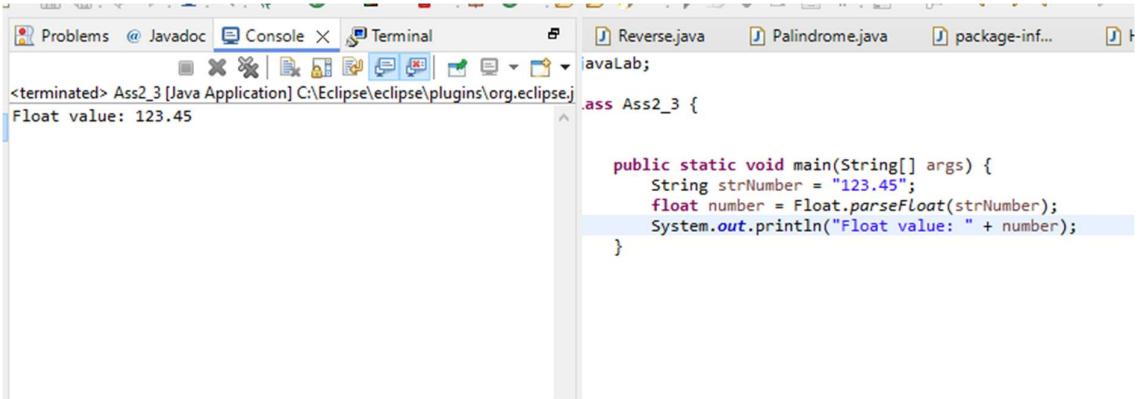


The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The output window displays the result of running the 'Ass2\_3' application, showing the string representation of a float value. The code editor shows the main method of the 'Ass2\_3' class, which uses the `Float.toString` method to convert a float variable to a string and prints it.

```
javaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\Ass2_3.java
String representation of float: 123.45

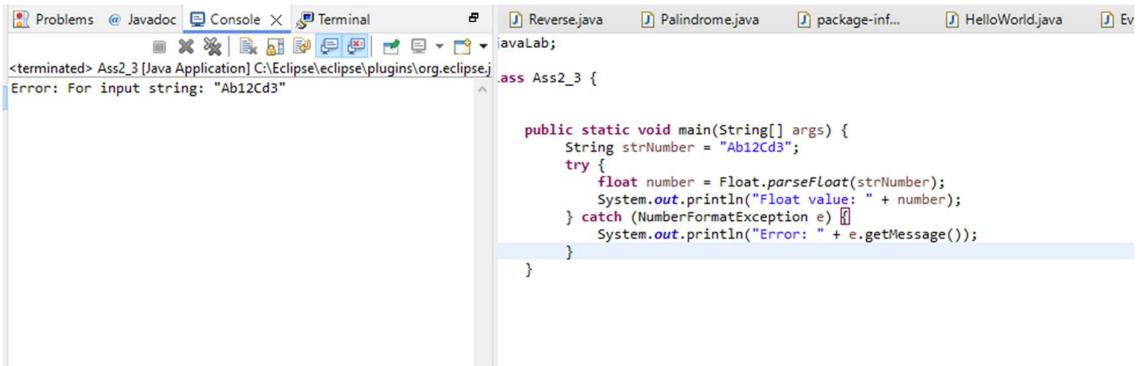
public static void main(String[] args) {
    float number = 123.45f;
    String strNumber = Float.toString(number);
    System.out.println("String representation of float: " + strNumber);
}
```

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)`).



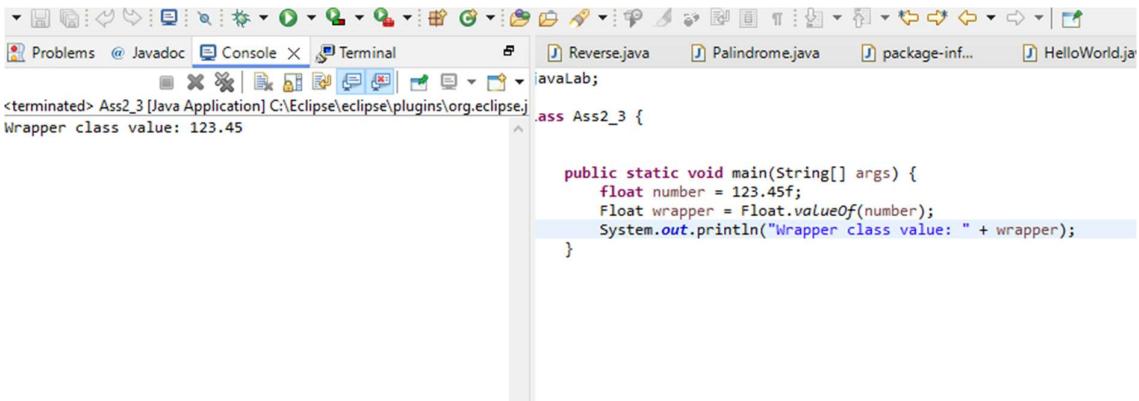
```
Problems @ Javadoc Console X Terminal
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.j
avaLab;
class Ass2_3 {
    public static void main(String[] args) {
        String strNumber = "123.45";
        float number = Float.parseFloat(strNumber);
        System.out.println("Float value: " + number);
    }
}
```

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).



```
Problems @ Javadoc Console X Terminal
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.j
avaLab;
class Ass2_3 {
    public static void main(String[] args) {
        String strNumber = "Ab12Cd3";
        try {
            float number = Float.parseFloat(strNumber);
            System.out.println("Float value: " + number);
        } catch (NumberFormatException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

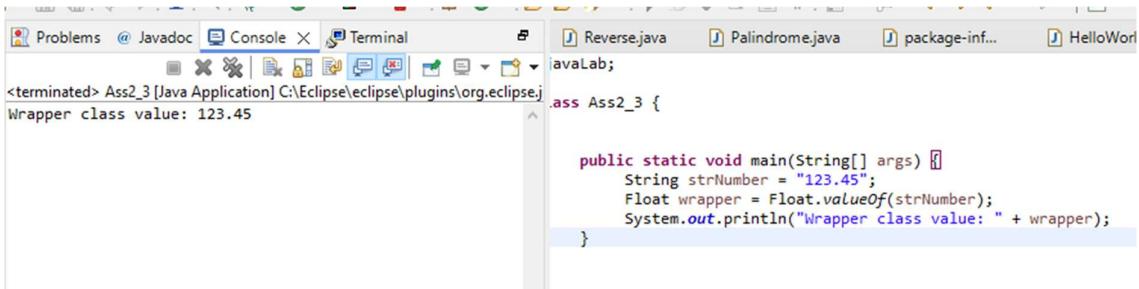
g. Declare a method-local variable number of type float with some value and convert it to the corresponding wrapper class using Float.valueOf(). (Hint: Use Float.valueOf(float)).



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\javaLab;
Wrapper class value: 123.45

.ass Ass2_3 {
    public static void main(String[] args) {
        float number = 123.45f;
        Float wrapper = Float.valueOf(number);
        System.out.println("Wrapper class value: " + wrapper);
    }
}
```

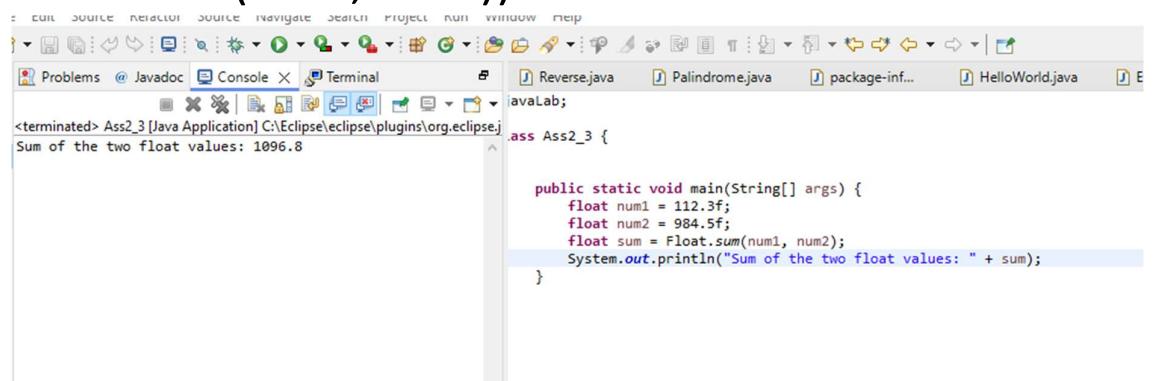
h. Declare a method-local variable strNumber of type String with some float value and convert it to the corresponding wrapper class using `Float.valueOf()`. (Hint: Use `Float.valueOf(String)`).



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\javaLab;
Wrapper class value: 123.45

.ass Ass2_3 {
    public static void main(String[] args) {
        String strNumber = "123.45";
        Float wrapper = Float.valueOf(strNumber);
        System.out.println("Wrapper class value: " + wrapper);
    }
}
```

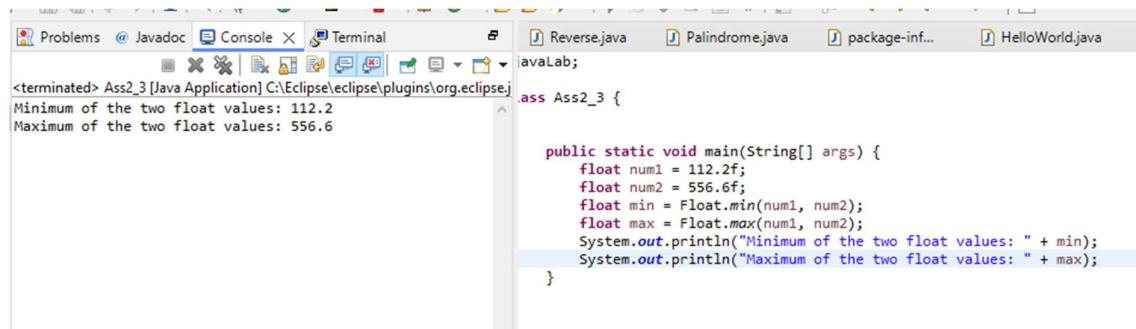
i. Declare two float variables with values 112.3 and 984.5, and add them using a method from the `Float` class. (Hint: Use `Float.sum(float, float)`).



```
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.jdt.core\src\javaLab;
Sum of the two float values: 1096.8

.ass Ass2_3 {
    public static void main(String[] args) {
        float num1 = 112.3f;
        float num2 = 984.5f;
        float sum = Float.sum(num1, num2);
        System.out.println("Sum of the two float values: " + sum);
    }
}
```

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)`).



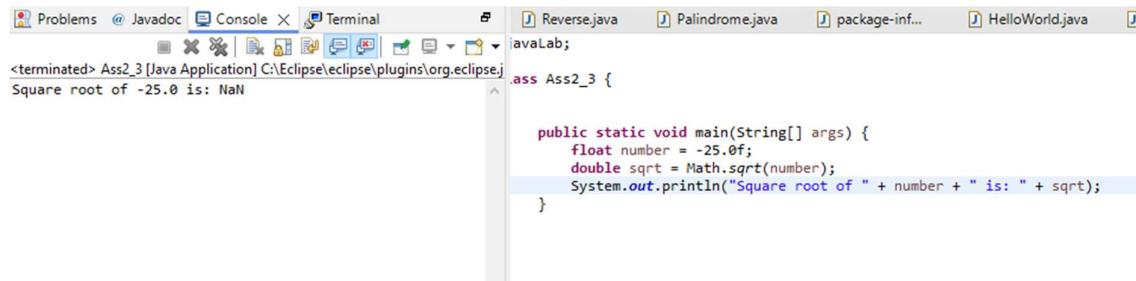
The screenshot shows the Eclipse IDE interface with the Java perspective selected. The top menu bar includes 'File', 'Edit', 'Select', 'Run', 'Debug', 'View', 'Search', 'Help', and 'Properties'. The left sidebar has 'Problems', '@ Javadoc', 'Console X', and 'Terminal'. The right sidebar lists files: Reverse.java, Palindrome.java, package-inf..., and HelloWorld.java. The main editor area contains the following Java code:

```
javaLab;
ass Ass2_3 {
    public static void main(String[] args) {
        float num1 = 112.2f;
        float num2 = 556.6f;
        float min = Float.min(num1, num2);
        float max = Float.max(num1, num2);
        System.out.println("Minimum of the two float values: " + min);
        System.out.println("Maximum of the two float values: " + max);
    }
}
```

The output window shows the results of the program execution:

```
<terminated> Ass2_3 [Java Application] C:\Eclipse\plugins\org.eclipse.jdt.core\src\javaLab\Ass2_3.java
Minimum of the two float values: 112.2
Maximum of the two float values: 556.6
```

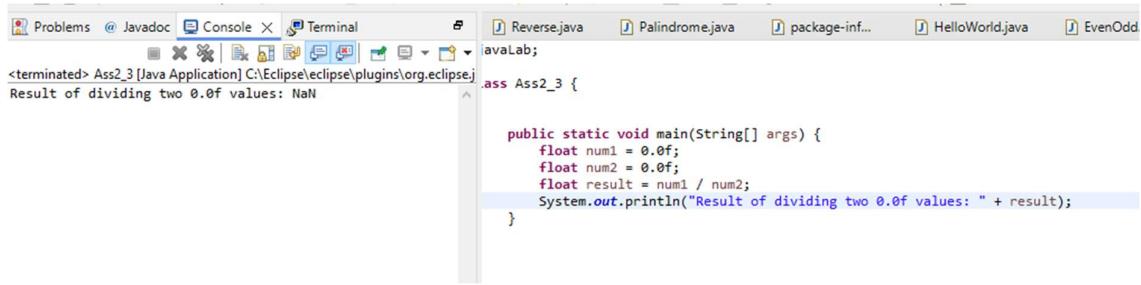
k. Declare a float variable with the value -25.0f. Find the square root of this value. (Hint: Use `Math.sqrt()` method).



The screenshot shows the Eclipse IDE interface with the Java perspective selected. The top menu bar includes 'File', 'Edit', 'Select', 'Run', 'Debug', 'View', 'Search', 'Help', and 'Properties'. The left sidebar has 'Problems', '@ Javadoc', 'Console X', and 'Terminal'. The right sidebar lists files: Reverse.java, Palindrome.java, package-inf..., and HelloWorld.java. The main editor area contains the following Java code:

```
javaLab;
ass Ass2_3 {
    public static void main(String[] args) {
        float number = -25.0f;
        double sqrt = Math.sqrt(number);
        System.out.println("Square root of " + number + " is: " + sqrt);
    }
}
```

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

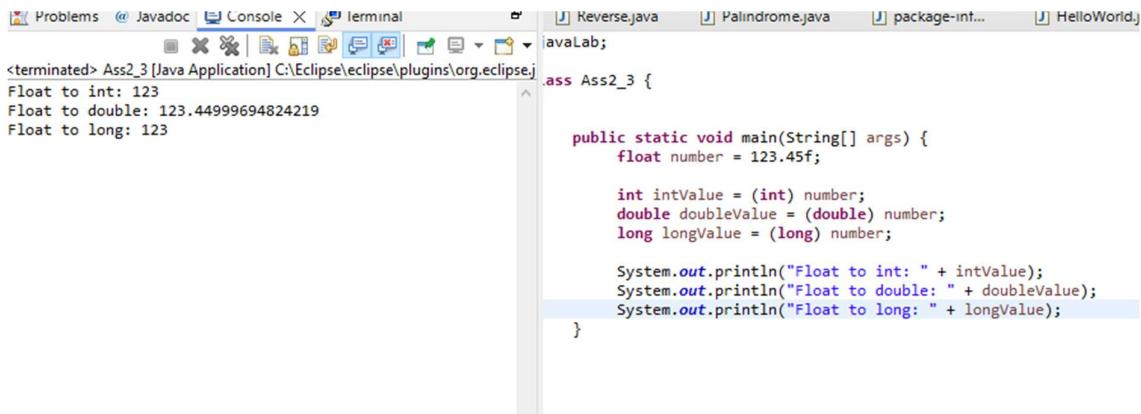


```
Problems @ Javadoc Console X Terminal
avaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.j
Result of dividing two 0.0f values: NaN

.ass Ass2_3 {

    public static void main(String[] args) {
        float num1 = 0.0f;
        float num2 = 0.0f;
        float result = num1 / num2;
        System.out.println("Result of dividing two 0.0f values: " + result);
    }
}
```

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



```
Problems @ Javadoc Console X terminal
avaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.j
Float to int: 123
Float to double: 123.44999694824219
Float to long: 123

.ass Ass2_3 {

    public static void main(String[] args) {
        float number = 123.45f;

        int intValue = (int) number;
        double doubleValue = (double) number;
        long longValue = (long) number;

        System.out.println("Float to int: " + intValue);
        System.out.println("Float to double: " + doubleValue);
        System.out.println("Float to long: " + longValue);
    }
}
```

## 7. Working with java.lang.Double

- Explore the Java API documentation for `java.lang.Double` and observe its modifiers and super types.
- Write a program to test how many bytes are used to represent a double value using the `BYTES` field. (Hint: Use `Double.BYTES`).

A screenshot of the Eclipse IDE interface. The top bar shows tabs for 'Problems', '@ Javadoc', 'Console X', and 'Terminal'. Below the tabs, the console area displays the output of a terminated Java application named 'Ass2\_3'. The output reads: 'Number of bytes used to represent a double value: 8'. The code editor window shows a Java file named 'Ass2\_3.java' with the following content:

```
javaLab;
public class Ass2_3 {
    public static void main(String[] args) {
        int bytes = Double.BYTES;
        System.out.println("Number of bytes used to represent a double value: " + 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).

A screenshot of the Eclipse IDE interface. The top bar shows tabs for 'Problems', '@ Javadoc', 'Console X', and 'Terminal'. Below the tabs, the console area displays the output of a terminated Java application named 'Ass2\_3'. The output reads: 'minimum double value: 4.9E-324' and 'maximum double value: 1.7976931348623157E308'. The code editor window shows a Java file named 'Ass2\_3.java' with the following content:

```
javaLab;
public class Ass2_3 {
    public static void main(String[] args) {
        double min = Double.MIN_VALUE;
        double max = Double.MAX_VALUE;
        System.out.println("Minimum double value: " + min);
        System.out.println("Maximum double value: " + max);
    }
}
```

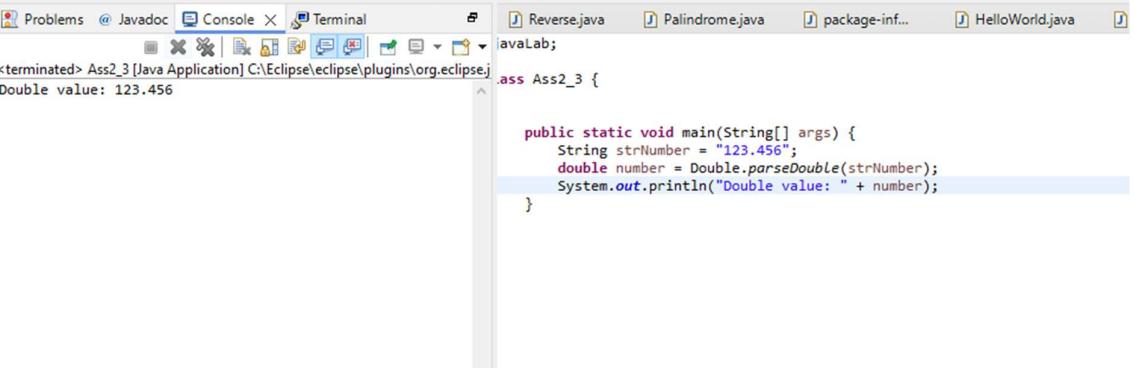
d. Declare a method-local variable number of type double with some value and convert it to a String using the toString method. (Hint: Use Double.toString(double)).

A screenshot of the Eclipse IDE interface. The top bar shows tabs for 'Problems', '@ Javadoc', 'Console X', and 'Terminal'. Below the tabs, the console area displays the output of a terminated Java application named 'Ass2\_3'. The output reads: 'String representation of double: 123.456'. The code editor window shows a Java file named 'Ass2\_3.java' with the following content:

```
javaLab;
public class Ass2_3 {
    public static void main(String[] args) {
        double number = 123.456;
        String strNumber = Double.toString(number);
        System.out.println("String representation of double: " + strNumber);
    }
}
```

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

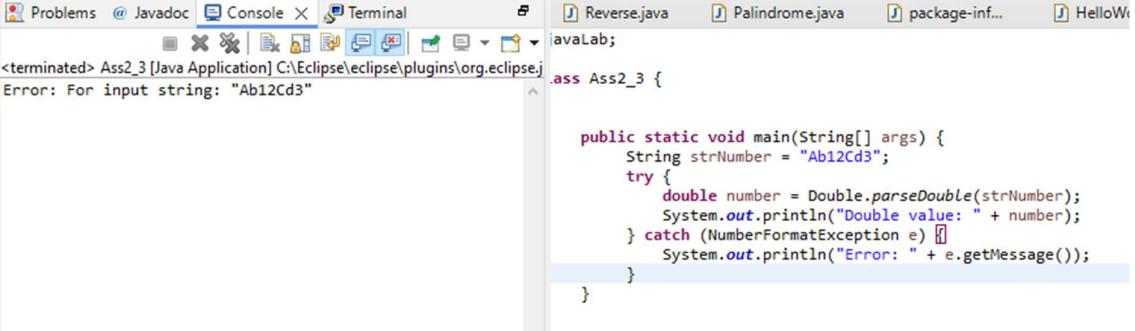
double value using the parseDouble method.  
(Hint: Use Double.parseDouble(String)).



The screenshot shows the Eclipse IDE interface with the 'Console' tab active. The output window displays the message 'Double value: 123.456'. The code editor shows a Java file named 'Ass2\_3.java' with the following content:

```
public static void main(String[] args) {
    String strNumber = "123.456";
    double number = Double.parseDouble(strNumber);
    System.out.println("Double value: " + number);
}
```

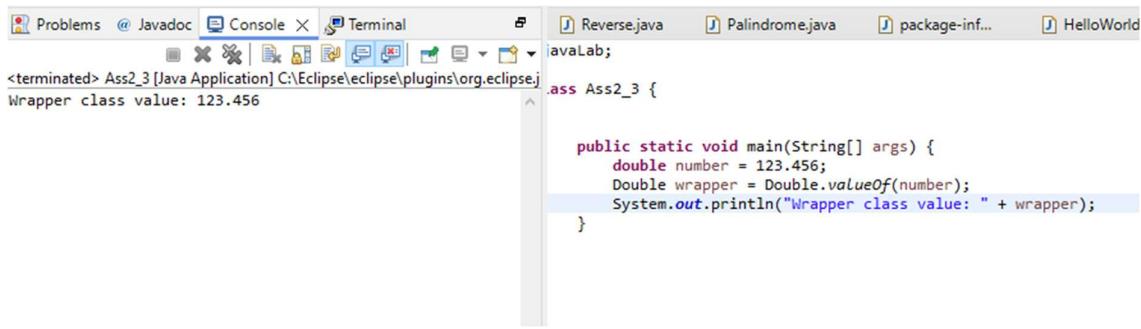
f. Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to a double value. (Hint: parseDouble method will throw a NumberFormatException).



The screenshot shows the Eclipse IDE interface with the 'Console' tab active. The output window displays the error message 'Error: For input string: "Ab12Cd3"'. The code editor shows a Java file named 'Ass2\_3.java' with the following content:

```
public static void main(String[] args) {
    String strNumber = "Ab12Cd3";
    try {
        double number = Double.parseDouble(strNumber);
        System.out.println("Double value: " + number);
    } catch (NumberFormatException e) {
        System.out.println("Error: " + e.getMessage());
    }
}
```

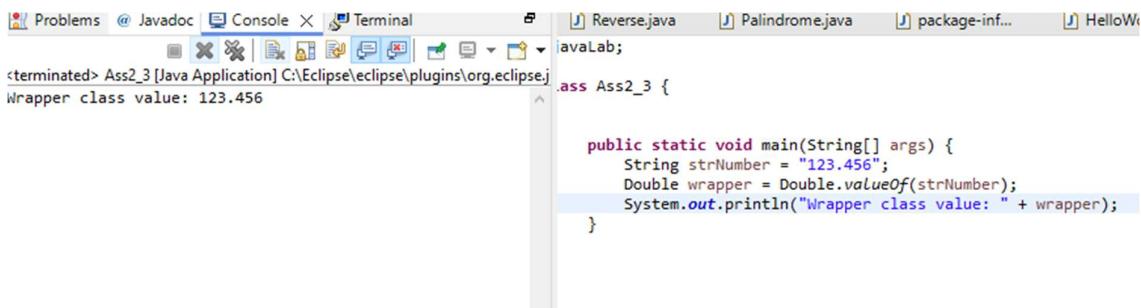
g. Declare a method-local variable number of type double with some value and convert it to the corresponding wrapper class using Double.valueOf(). (Hint: Use Double.valueOf(double)).



```
Problems @ Javadoc Console X Terminal Reverse.java Palindrome.java package-info... HelloWorld.java
avaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.j
Wrapper class value: 123.456
.ass Ass2_3 {

    public static void main(String[] args) {
        double number = 123.456;
        Double wrapper = Double.valueOf(number);
        System.out.println("Wrapper class value: " + wrapper);
    }
}
```

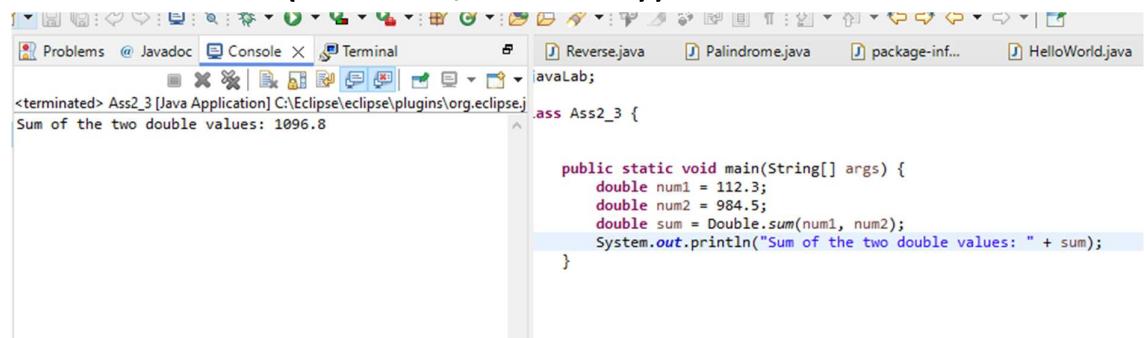
h. Declare a method-local variable strNumber of type String with some double value and convert it to the corresponding wrapper class using Double.valueOf(). (Hint: Use Double.valueOf(String)).



```
Problems @ Javadoc Console X Terminal Reverse.java Palindrome.java package-info... HelloWorld.java
avaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.j
Wrapper class value: 123.456
.ass Ass2_3 {

    public static void main(String[] args) {
        String strNumber = "123.456";
        Double wrapper = Double.valueOf(strNumber);
        System.out.println("Wrapper class value: " + wrapper);
    }
}
```

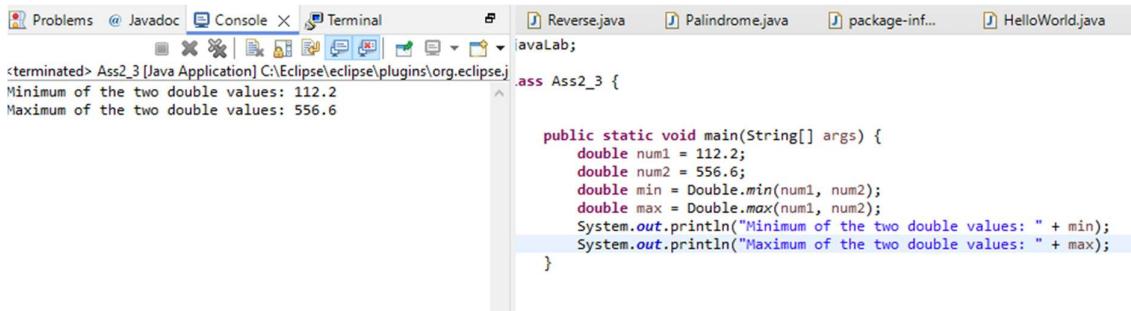
i. Declare two double variables with values 112.3 and 984.5, and add them using a method from the Double class. (Hint: Use Double.sum(double, double)).



```
Problems @ Javadoc Console X Terminal Reverse.java Palindrome.java package-info... HelloWorld.java
avaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.j
Sum of the two double values: 1096.8
.ass Ass2_3 {

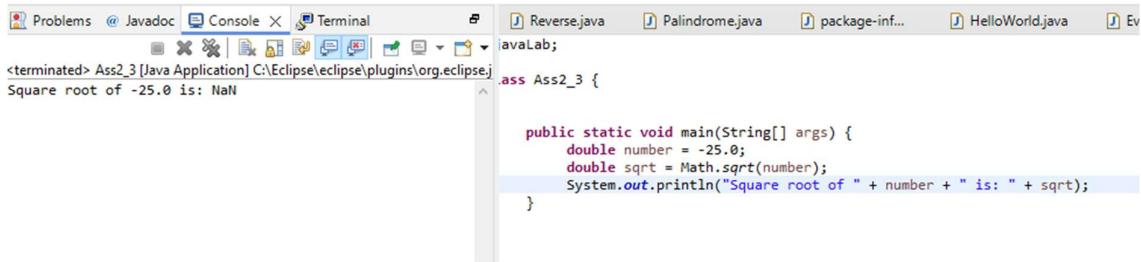
    public static void main(String[] args) {
        double num1 = 112.3;
        double num2 = 984.5;
        double sum = Double.sum(num1, num2);
        System.out.println("Sum of the two double values: " + sum);
    }
}
```

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



```
Problems @ Javadoc Console X Terminal Reverse.java Palindrome.java package-info... HelloWorld.java
javaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.j
ass Ass2_3 {
    public static void main(String[] args) {
        double num1 = 112.2;
        double num2 = 556.6;
        double min = Double.min(num1, num2);
        double max = Double.max(num1, num2);
        System.out.println("Minimum of the two double values: " + min);
        System.out.println("Maximum of the two double values: " + max);
    }
}
```

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

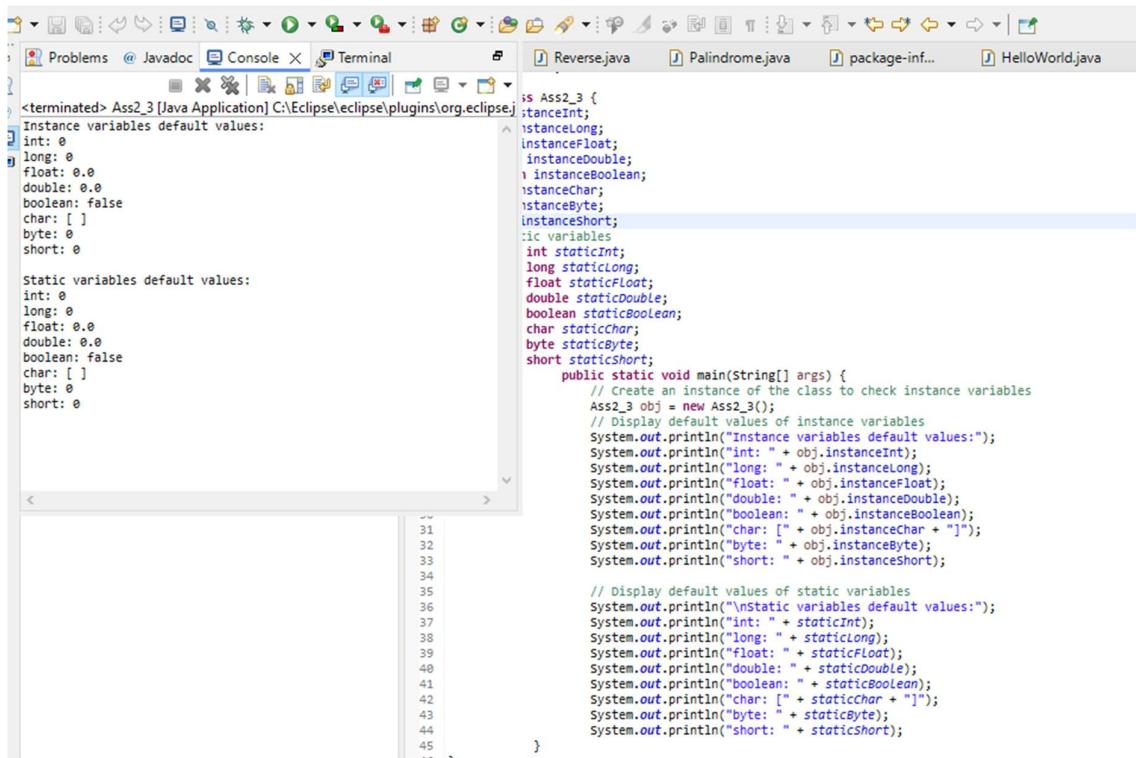


```
Problems @ Javadoc Console X Terminal Reverse.java Palindrome.java package-info... HelloWorld.java
javaLab;
<terminated> Ass2_3 [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.j
ass Ass2_3 {
    public static void main(String[] args) {
        double number = -25.0;
        double sqrt = Math.sqrt(number);
        System.out.println("Square root of " + number + " is: " + sqrt);
    }
}
```

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







The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The code in the editor is as follows:

```
<terminated> Ass2_3 [Java Application] C:\Eclipse\workspace\org.eclipse.jdt.core\src\Ass2_3.java
Instance variables default values:
int: 0
long: 0
float: 0.0
double: 0.0
boolean: false
char: []
byte: 0
short: 0

Static variables default values:
int: 0
long: 0
float: 0.0
double: 0.0
boolean: false
char: []
byte: 0
short: 0

is Ass2_3 {
    instanceInt;
    instanceLong;
    instanceFloat;
    instanceDouble;
    instanceBoolean;
    instanceChar;
    instanceByte;
    instanceShort;
    static variables
    int staticInt;
    long staticLong;
    float staticFloat;
    double staticDouble;
    boolean staticBoolean;
    char staticChar;
    byte staticByte;
    short staticShort;
    public static void main(String[] args) {
        // Create an instance of the class to check instance variables
        Ass2_3 obj = new Ass2_3();
        // Display default values of instance variables
        System.out.println("Instance variables default values:");
        System.out.println("int: " + obj.instanceInt);
        System.out.println("long: " + obj.instanceLong);
        System.out.println("float: " + obj.instanceFloat);
        System.out.println("double: " + obj.instanceDouble);
        System.out.println("boolean: " + obj.instanceBoolean);
        System.out.println("char: [" + obj.instanceChar + "]");
        System.out.println("byte: " + obj.instanceByte);
        System.out.println("short: " + obj.instanceShort);

        // Display default values of static variables
        System.out.println("static variables default values:");
        System.out.println("int: " + staticInt);
        System.out.println("long: " + staticLong);
        System.out.println("float: " + staticFloat);
        System.out.println("double: " + staticDouble);
        System.out.println("boolean: " + staticBoolean);
        System.out.println("char: [" + staticChar + "]");
        System.out.println("byte: " + staticByte);
        System.out.println("short: " + staticShort);
    }
}
```

## 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).

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** Shows the Eclipse logo, workspace name "Ass2\_3 [Java Application]", and various tool icons.
- Left Sidebar:** Lists "Problems", "@ Javadoc", "Console X", and "Terminal".
- Right Sidebar:** Shows tabs for "Reverse.java", "Palindrome.java", "package-inf...", and "HelloWor...".
- Code Editor:** Displays Java code for an arithmetic operations program. The code handles command-line arguments, performs arithmetic operations (addition, subtraction, multiplication, division, modulus), and handles invalid operators and zero division/modulus errors.

```
1 /**
2  * static void main(String[] args) {
3  *     if (args.length != 3) {
4  *         System.out.println("Usage: java ArithmeticOperations <n1> <operator> <n2>");
5  *         return;
6  *     }
7  *     try {
8  *         int n1 = Integer.parseInt(args[0]);
9  *         String operator = args[1];
10 *         int n2 = Integer.parseInt(args[2]);
11 *         int result = 0;
12 *
13 *         switch (operator) {
14 *             case "+":
15 *                 result = n1 + n2;
16 *                 break;
17 *             case "-":
18 *                 result = n1 - n2;
19 *                 break;
20 *             case "*":
21 *                 result = n1 * n2;
22 *                 break;
23 *             case "/":
24 *                 if (n2 != 0) {
25 *                     result = n1 / n2;
26 *                 } else {
27 *                     System.out.println("Error: Division by zero is not allowed.");
28 *                     return;
29 *                 }
30 *                 break;
31 *             case "%":
32 *                 if (n2 != 0) {
33 *                     result = n1 % n2;
34 *                 } else {
35 *                     System.out.println("Error: Modulus by zero is not allowed.");
36 *                     return;
37 *                 }
38 *                 break;
39 *             default:
40 *                 System.out.println("Error: Invalid operator. Use +, -, *, /, or %.");
41 *                 return;
42 *         }
43 *         System.out.println("Result: " + n1 + " " + operator + " " + n2 + " = " + result);
44 *     } catch (NumberFormatException e) {
45 *         System.out.println("Error: Please enter valid integers for n1 and n2.");
46 *     }
47 * }
48 */
49
50
51
52
53
54
55 }
```