

**Note:** Consider the following before starting the assignment:

- A **static field** declared inside a class is called a **class-level variable**. To access this variable, use the class name and the dot operator (e.g., `Integer.MAX_VALUE`).
- A **static method** defined inside a class is called a **class-level method**. To access this method, use the class name and the dot operator (e.g., `Integer.parseInt()`).
- When accessing static members within the same class, you do not need to use the class name.

### 1. Working with `java.lang.Boolean`

a. Explore the [Java API documentation for `java.lang.Boolean`](#) and observe its modifiers and super types.

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)` ).

```
public class bool {  
  
    public static void main(String[] args) {  
  
        boolean status = true ;  
  
        String stringstr = Boolean.toString(status);  
  
        System.out.println(stringstr);  
  
    }  
  
}
```

## ASSIGNMENT NO.2

```
1 package oopj;  
2  
3 public class bool {  
4  
5     public static void main(String[] args) {  
6         boolean status = true ;  
7         String stringstr = Boolean.toString(status);  
8         System.out.println(stringstr);  
9     }  
10 }  
11 }  
12
```

Console ×

<terminated> bool [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (10 Sept 2024)  
true

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)`).

```
public class strtoboo {  
  
    public static void main(String[] args) {  
  
        String strStatus = "true";  
  
        Boolean.parseBoolean(strStatus);  
  
        System.out.println(strStatus);  
  
    }  
  
}
```

## ASSIGNMENT NO.2

```
1 package oopj;
2
3 public class strtoboo {
4     public static void main(String[] args) {
5         String strStatus = "true";
6         Boolean.parseBoolean(strStatus);
7         System.out.println(strStatus);
8     }
9
10 }
11
```

Console ×

<terminated> strtoboo [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (10 Sept  
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"`).

```
public class strtoboo1 {

    public static void main(String[] args) {

        String strStatus = "1";

        boolean bo = Boolean.parseBoolean(strStatus);

        System.out.println(bo);

    }

}
```

## ASSIGNMENT NO.2

```
1 package oopj;  
2  
3 public class strtobool {  
4  
5     public static void main(String[] args) {  
6         String strStatus = "1";  
7         boolean bo =Boolean.parseBoolean(strStatus);  
8         System.out.println(bo);  
9     }  
10 }  
11  
12
```

Console ×

<terminated> strtobool [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (10 Se  
false

Writable | Smart Insert | 10 : 10 : 200

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)`).

```
public static void main(String[] args) {
```

```
String status ="true";
```

```
boolean bo =Boolean.valueOf(status);
```

```
System.out.println(bo);
```

```
}
```

## ASSIGNMENT NO.2

```
1 package oopj;  
2  
3 public class strtoboo2 {  
4     public static void main(String[] args) {  
5         String status ="true";  
6         boolean bo =Boolean.valueOf(status);  
7         System.out.println(bo);  
8     }  
9 }  
10
```

Console X

<terminated> strtoboo2 [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (10 Se  
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)`).

```
public class Strtoboo3{  
  
    public static void main(String[] args) {  
  
        String strStatus ="true";  
  
        boolean bo =Boolean.valueOf(strStatus);  
  
        System.out.println(bo);  
    }  
  
}
```

## ASSIGNMENT NO.2

```
1 package oopj;
2
3 public class strtoboo3 {
4
5     public static void main(String[] args) {
6         String strStatus = "true";
7         boolean bo = Boolean.valueOf(strStatus);
8         System.out.println(bo);
9     }
10 }
11 }
12 }
```

Console X

<terminated> strtoboo3 [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64.22.0.2.v20240802-1626\jre\bin\javaw.exe (10 S  
true

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

## 2. Working with `java.lang.Byte`

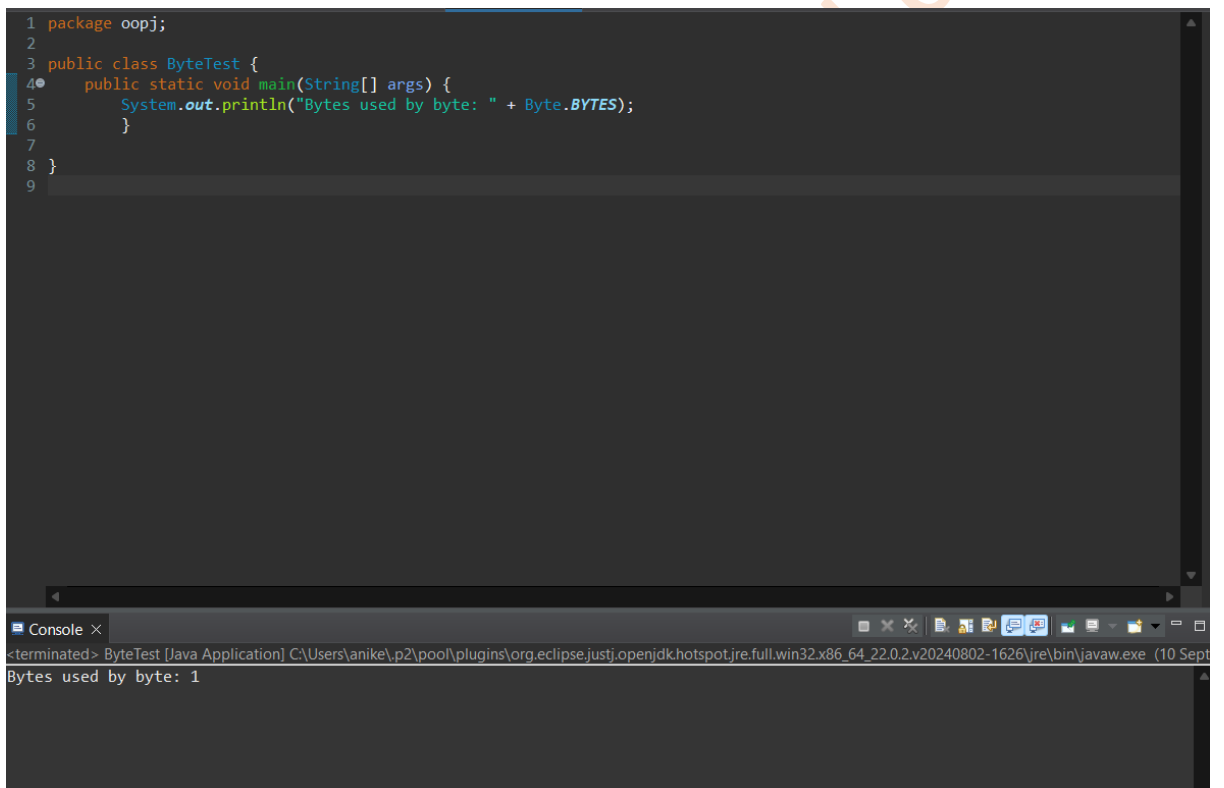
a. Explore the [Java API documentation for `java.lang.Byte`](#) and observe its modifiers and super types.

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

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

```
package oopj;
```

```
public class ByteTest {  
  
    public static void main(String[] args) {  
  
        System.out.println("Bytes used by byte: " + Byte.BYTES);  
  
    }  
  
}
```



The screenshot shows an IDE with a Java file named ByteTest.java. The code is as follows:

```
1 package oopj;  
2  
3 public class ByteTest {  
4     public static void main(String[] args) {  
5         System.out.println("Bytes used by byte: " + Byte.BYTES);  
6     }  
7  
8 }  
9
```

Below the code editor, the console window is open, showing the output: "Bytes used by byte: 1". The console title bar indicates the application is "ByteTest [Java Application]" and the path is "C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (10 Sept)".

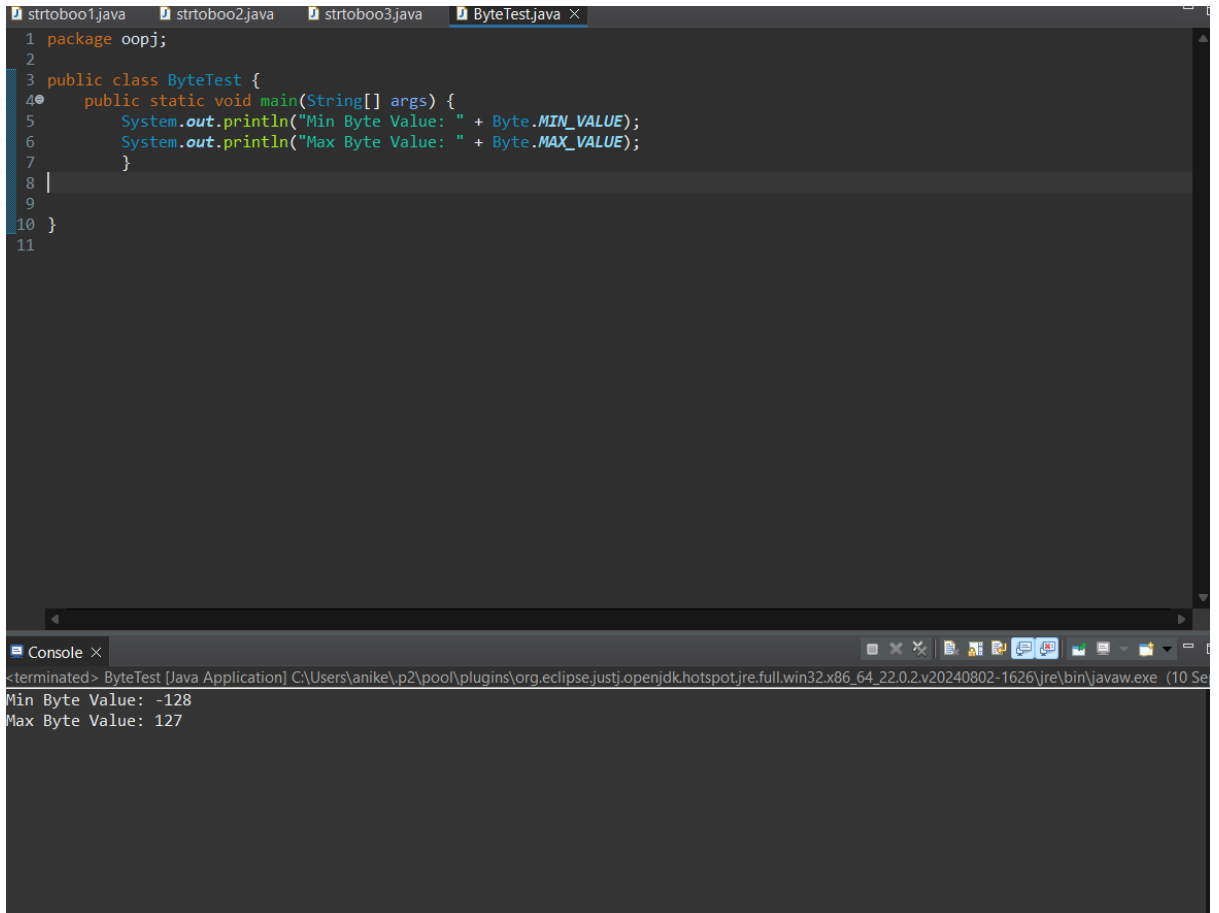
c. Write a program to find the minimum and maximum values of `byte` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Byte.MIN_VALUE` and `Byte.MAX_VALUE`).

```
public class ByteTest {  
  
    public static void main(String[] args) {  
  
        System.out.println("Min Byte Value: " + Byte.MIN_VALUE);  
  
        System.out.println("Max Byte Value: " + Byte.MAX_VALUE);  
  
    }  
  
}
```

## ASSIGNMENT NO.2

```
}
```

```
}
```



```
1 package oopj;
2
3 public class ByteTest {
4     public static void main(String[] args) {
5         System.out.println("Min Byte Value: " + Byte.MIN_VALUE);
6         System.out.println("Max Byte Value: " + Byte.MAX_VALUE);
7     }
8
9
10 }
11
```

```
<terminated> ByteTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\bin\javaw.exe (10 Se
Min Byte Value: -128
Max Byte Value: 127
```

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)` ).

```
public class ByteTest {
```

```
    public static void main(String[] args) {
```

```
        byte number = 10;
```

```
        String byteAsString = Byte.toString(number);
```

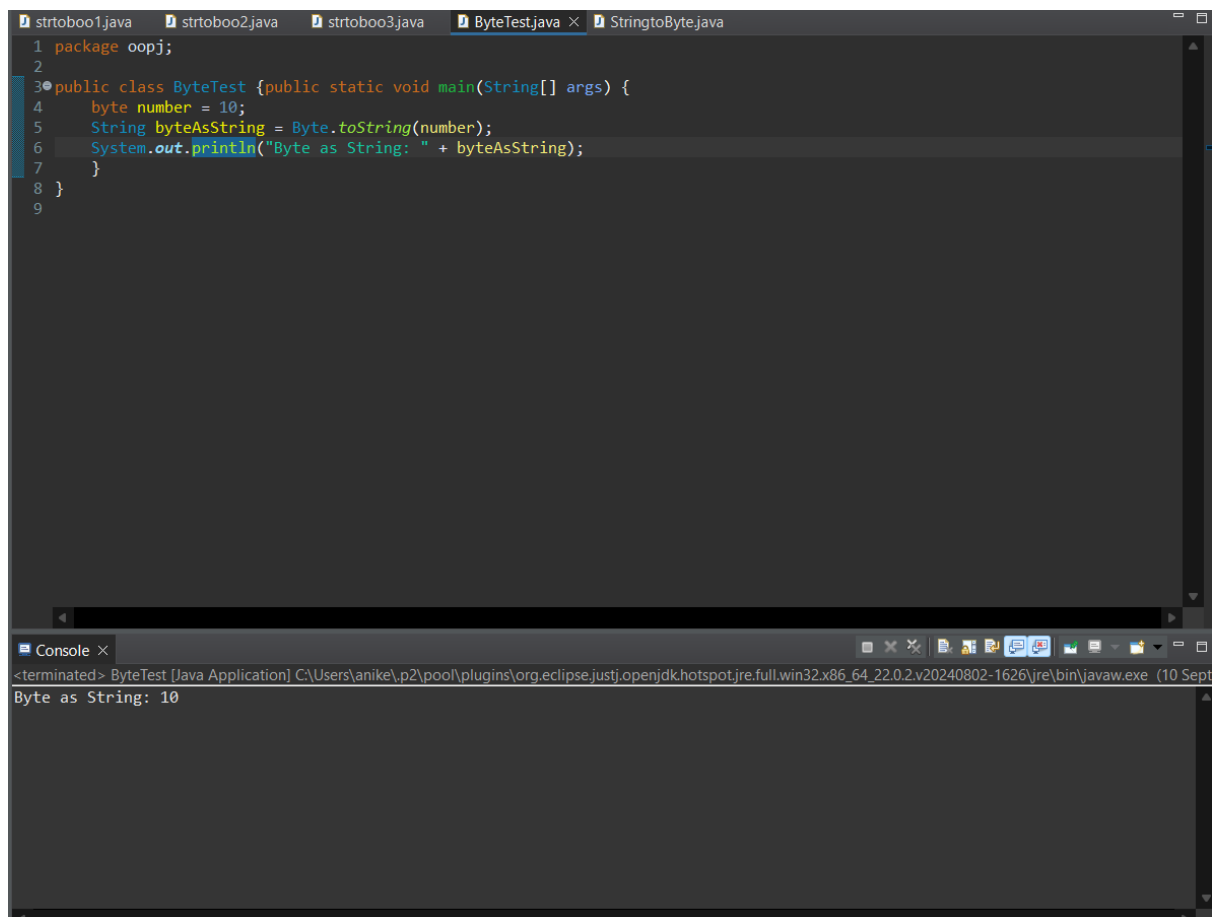
```
        System.out.println("Byte as String: " + byteAsString);
```

```
    }
```

```
}
```



## ASSIGNMENT NO.2



```
1 package oopj;
2
3 public class ByteTest {public static void main(String[] args) {
4     byte number = 10;
5     String byteAsString = Byte.toString(number);
6     System.out.println("Byte as String: " + byteAsString);
7 }
8 }
9
```

Console

<terminated> ByteTest [Java Application] C:\Users\anike\.p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (10 Sept  
Byte as String: 10

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)`).

```
public class StringtoByte {

    public static void main(String[] args) {

        String strNumber = "12";

        byte byteValue = Byte.parseByte(strNumber);

        System.out.println("String to Byte: " + byteValue);

    }

}
```

## ASSIGNMENT NO.2

```
1 package oopj;
2
3 public class StringtoByte {
4     public static void main(String[] args) {
5         String strNumber = "12";
6         byte byteValue = Byte.parseByte(strNumber);
7         System.out.println("String to Byte: " + byteValue);
8     }
9 }
10
11
```

Console ×

<terminated> StringtoByte [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (String to Byte: 12

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`).

```
public class InvalidByteConversion {

    public static void main(String[] args) {

        String strNumber = "Ab12Cd3";

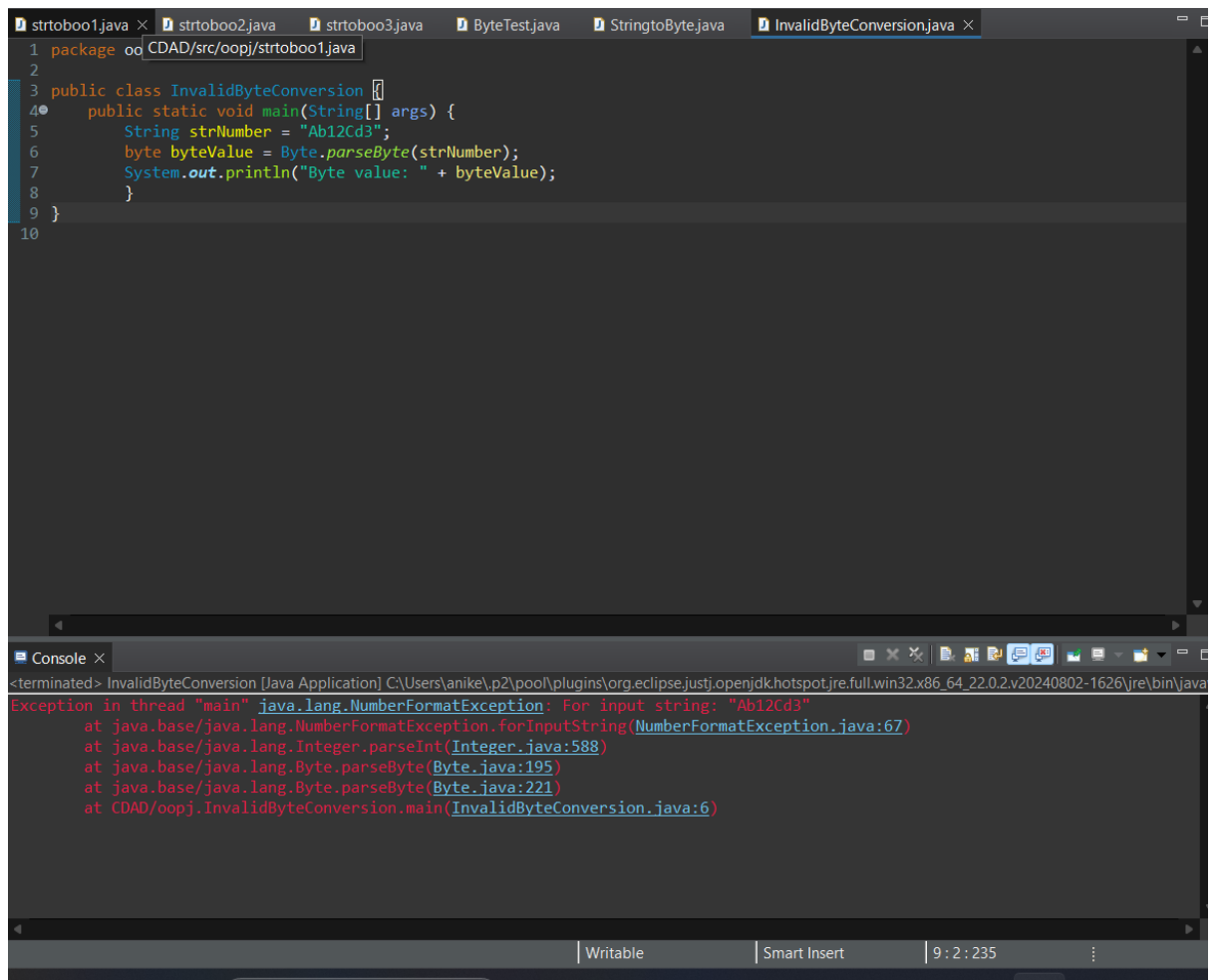
        byte byteValue = Byte.parseByte(strNumber);

        System.out.println("Byte value: " + byteValue);

    }

}
```

## ASSIGNMENT NO.2



The screenshot shows the Eclipse IDE with several tabs open. The active tab is 'InvalidByteConversion.java'. The code in this file is as follows:

```
1 package oo.CDAD/src/ooj/strtoboo1.java;
2
3 public class InvalidByteConversion {
4     public static void main(String[] args) {
5         String strNumber = "Ab12Cd3";
6         byte byteValue = Byte.parseByte(strNumber);
7         System.out.println("Byte value: " + byteValue);
8     }
9 }
10
```

Below the code editor, the 'Console' window shows the following error message:

```
<terminated> InvalidByteConversion [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\bin\java
Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
    at java.base/java.lang.Integer.parseInt(Integer.java:588)
    at java.base/java.lang.Byte.parseByte(Byte.java:195)
    at java.base/java.lang.Byte.parseByte(Byte.java:221)
    at CDAD/ooj.InvalidByteConversion.main(InvalidByteConversion.java:6)
```

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)`).

```
public class ByteTest {

    public static void main(String[] args) {

        byte number = 10;

        Byte byteWrapper = Byte.valueOf(number);

        System.out.println("Byte as Wrapper: " + byteWrapper);

    }

}
```

## ASSIGNMENT NO.2

```
1 package oopj;
2
3 public class ByteTest {
4     public static void main(String[] args) {
5         byte number = 10;
6         Byte byteWrapper = Byte.valueOf(number);
7         System.out.println("Byte as Wrapper: " + byteWrapper);
8     }
9 }
10
```

Console ×

<terminated> ByteTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (10 Sep 2024, 10:00 AM)

Byte as Wrapper: 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)`).

```
public class StringtoByteValueOf {

    public static void main(String[] args) {

        String strNumber = "20";

        Byte byteWrapper = Byte.valueOf(strNumber);

        System.out.println("String to Byte Wrapper: " + byteWrapper);

    }

}
```

## ASSIGNMENT NO.2

```
1 package oopj;
2
3 public class StringtoByteValueOf {
4     public static void main(String[] args) {
5         String strNumber = "20";
6         Byte byteWrapper = Byte.valueOf(strNumber);
7         System.out.println("String to Byte Wrapper: " + byteWrapper);
8     }
9 }
10
11
```

Console ×

<terminated> StringtoByteValueOf [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.  
String to Byte Wrapper: 20

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

```
public class ByteConversion {

    public static void main(String[] args) {

        byte number = 50;

        int intValue = number;

        double doubleValue = number;

        System.out.println("Byte to Int: " + intValue);

        System.out.println("Byte to Double: " + doubleValue);

    }

}
```

## ASSIGNMENT NO.2

```
1 package oopj;
2
3 public class ByteConversion {
4     public static void main(String[] args) {
5         byte number = 50;
6         int intValue = number;
7         double doubleValue = number;
8         System.out.println("Byte to Int: " + intValue);
9         System.out.println("Byte to Double: " + doubleValue);
10    }
11 }
12 }
13 }
```

Console X

```
<terminated> ByteConversion [Java Application] C:\Users\anike\p2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\bin\javaw.exe (1)
Byte to Int: 50
Byte to Double: 50.0
```

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

- a. Explore the [Java API documentation for `java.lang.Short`](#) and observe its modifiers and super types.

Modifier and Type	Field and Description
static int	<b>BYTES</b> The number of bytes used to represent a short value in two's complement binary form.
static short	<b>MAX_VALUE</b> A constant holding the maximum value a short can have, $2^{15}-1$ .
static short	<b>MIN_VALUE</b> A constant holding the minimum value a short can have, $-2^{15}$ .
static int	<b>SIZE</b> The number of bits used to represent a short value in two's complement binary form.
static Class<Short>	<b>TYPE</b> The Class instance representing the primitive type short.

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

```
public class ShortTest {

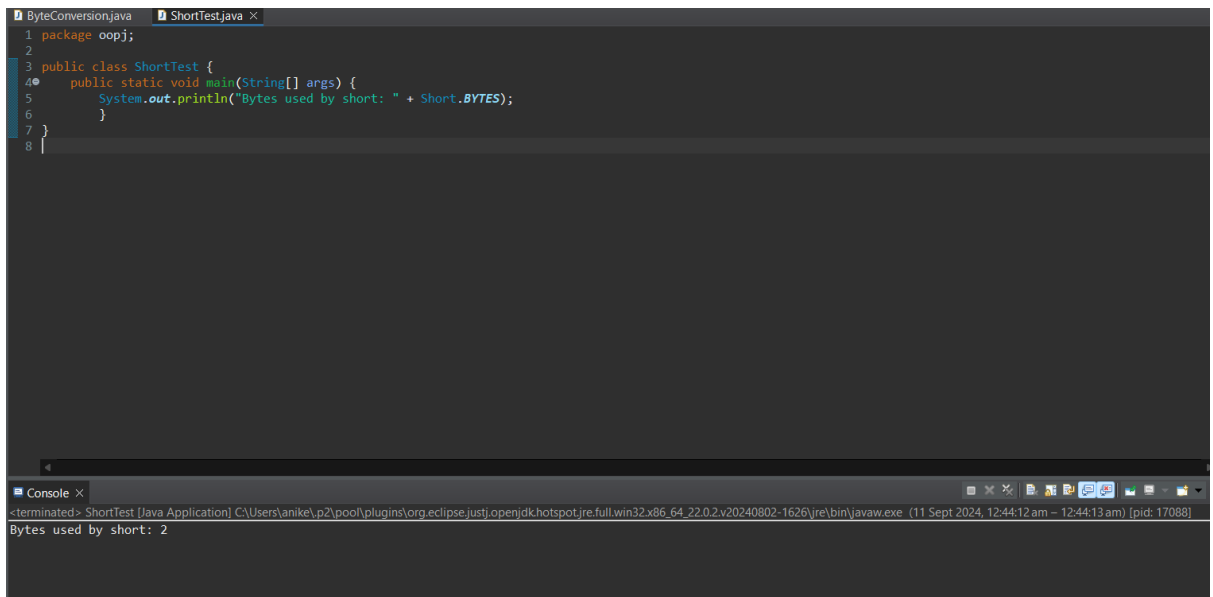
    public static void main(String[] args) {

        System.out.println("Bytes used by short: " + Short.BYTES);

    }

}
```

## ASSIGNMENT NO.2



```
ByteConversion.java ShortTest.java X
1 package oopj;
2
3 public class ShortTest {
4     public static void main(String[] args) {
5         System.out.println("Bytes used by short: " + Short.BYTES);
6     }
7 }
8

Console X
<terminated> ShortTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.22.0.2.v20240802-1626\jre\bin\javaw.exe (11 Sept 2024, 12:44:12 am - 12:44:13 am) [pid: 17088]
Bytes used by short: 2
```

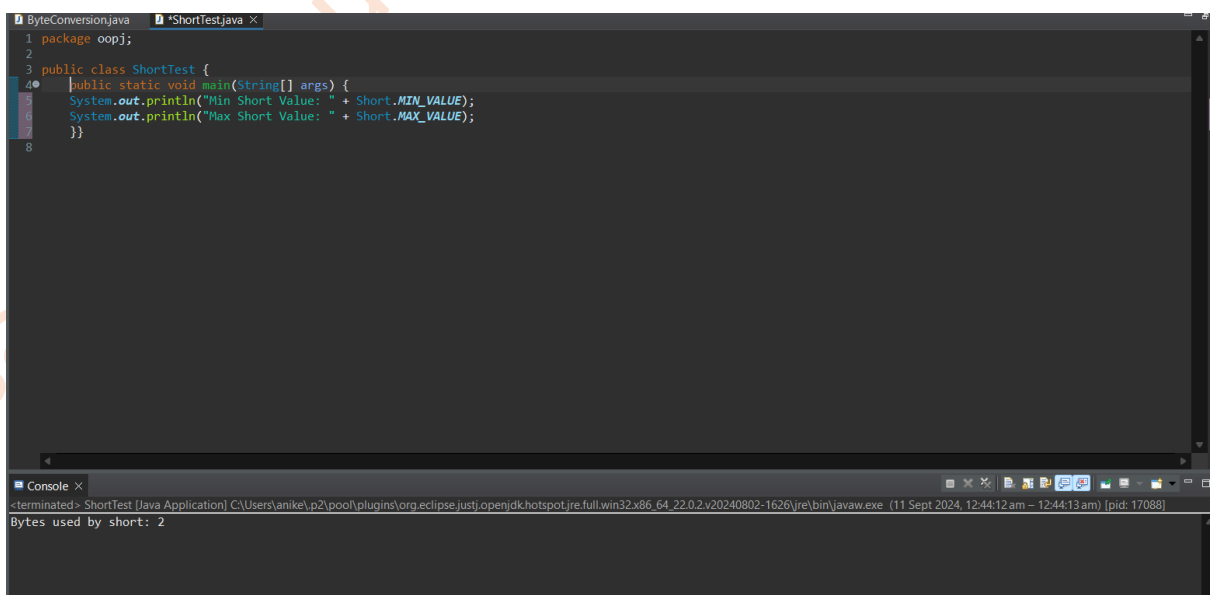
c. Write a program to find the minimum and maximum values of `short` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Short.MIN_VALUE` and `Short.MAX_VALUE`).

```
public static void main(String[] args) {

    System.out.println("Min Short Value: " + Short.MIN_VALUE);

    System.out.println("Max Short Value: " + Short.MAX_VALUE);

}
```



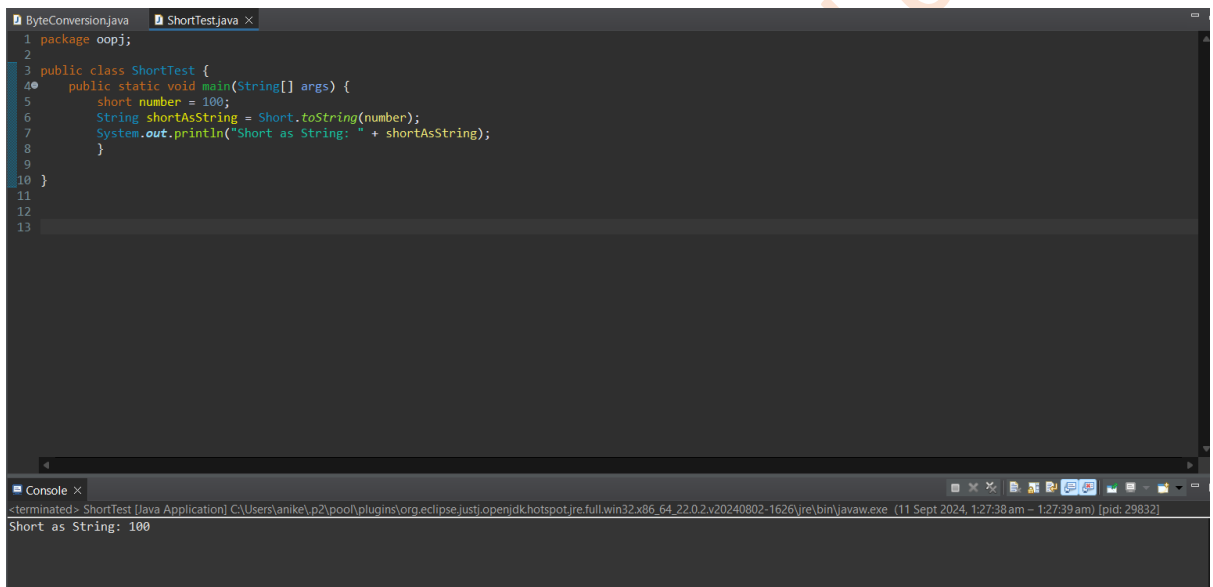
```
ByteConversion.java ShortTest.java X
1 package oopj;
2
3 public class ShortTest {
4     public static void main(String[] args) {
5         System.out.println("Min Short Value: " + Short.MIN_VALUE);
6         System.out.println("Max Short Value: " + Short.MAX_VALUE);
7     }
8 }

Console X
<terminated> ShortTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.22.0.2.v20240802-1626\jre\bin\javaw.exe (11 Sept 2024, 12:44:12 am - 12:44:13 am) [pid: 17088]
Bytes used by short: 2
```

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)`).

## ASSIGNMENT NO.2

```
public class ShortTest {  
  
    public static void main(String[] args) {  
  
        short number = 100;  
  
        String shortAsString = Short.toString(number);  
  
        System.out.println("Short as String: " + shortAsString);  
  
    }  
  
}
```



The screenshot shows an IDE with two tabs: 'ByteConversion.java' and 'ShortTest.java'. The 'ShortTest.java' tab is active, displaying the following code:

```
1 package oopj;  
2  
3 public class ShortTest {  
4     public static void main(String[] args) {  
5         short number = 100;  
6         String shortAsString = Short.toString(number);  
7         System.out.println("Short as String: " + shortAsString);  
8     }  
9  
10 }  
11  
12  
13
```

Below the code editor, the 'Console' tab is open, showing the output of the program:

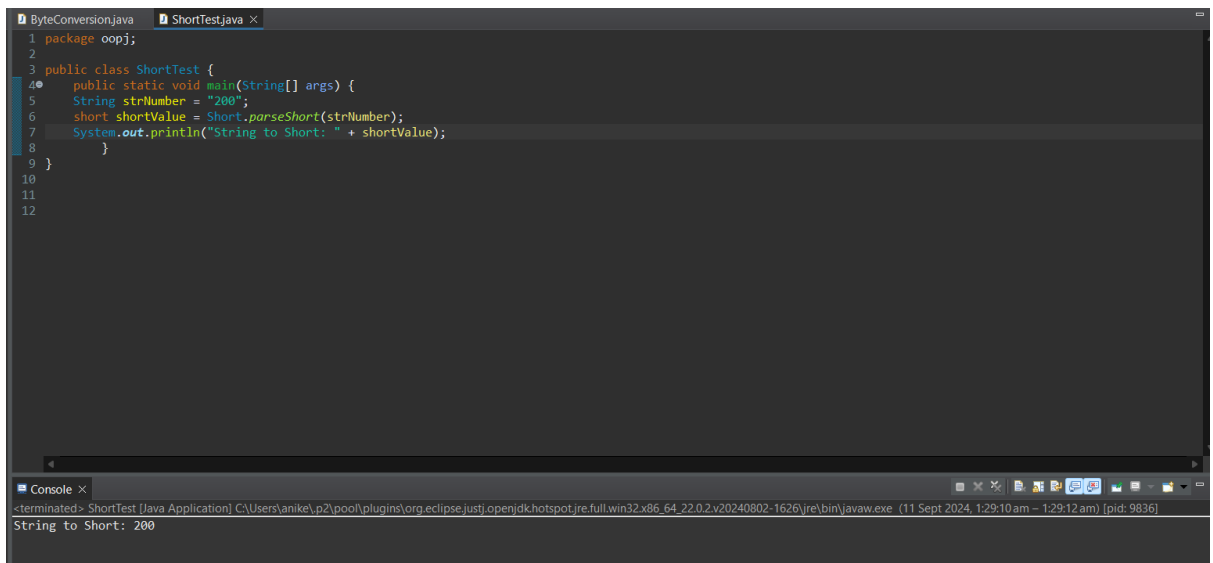
```
<terminated> ShortTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\bin\javaw.exe (11 Sept 2024, 1:27:38 am - 1:27:39 am) [pid: 29832]  
Short as String: 100
```

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)`).

```
public class ShortTest {  
  
    public static void main(String[] args) {  
  
        String strNumber = "200";  
  
        short shortValue = Short.parseShort(strNumber);  
  
        System.out.println("String to Short: " + shortValue);  
  
    }  
  
}
```



## ASSIGNMENT NO.2



```
1 package oopj;
2
3 public class ShortTest {
4     public static void main(String[] args) {
5         String strNumber = "200";
6         short shortValue = Short.parseShort(strNumber);
7         System.out.println("String to Short: " + shortValue);
8     }
9 }
10
11
12
```

Console x

<terminated> ShortTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (11 Sept 2024, 1:29:10 am - 1:29:12 am) [pid: 9836]

String to Short: 200

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`).

```
public class ShortTest {

    public static void main(String[] args) {

        String strNumber = "Ab12Cd3";

        short shortValue = Short.parseShort(strNumber);

        System.out.println("Short value: " + shortValue);

    }

}
```

## ASSIGNMENT NO.2

```
1 package oopj;
2
3 public class ShortTest {
4     public static void main(String[] args) {
5         String strNumber = "Ab12cd3";
6         short shortValue = Short.parseShort(strNumber);
7         System.out.println("Short value: " + shortValue);
8     }
9 }
10
11
12
```

Console ×

<terminated> ShortTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64.22.0.2.v20240802-1626\jre\bin\javaw.exe (11 Sept 2024, 1:30:17 am – 1:30:19 am) [pid: 26968]

Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12cd3"  
at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)  
at java.base/java.lang.Integer.parseInt(Integer.java:588)  
at java.base/java.lang.Short.parseShort(Short.java:138)  
at java.base/java.lang.Short.parseShort(Short.java:164)  
at CDAD\oopj.ShortTest.main(ShortTest.java:6)

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)`).

```
public static void main(String[] args) {  
  
    short number = 100;  
  
    Short wrappedNumber = Short.valueOf(number);  
  
    System.out.println("Wrapped short value: " + wrappedNumber);  
  
}
```

```
1 package oopj;
2
3 public class ShortWrapperExample {
4     public static void main(String[] args) {
5         short number = 100;
6         Short wrappedNumber = Short.valueOf(number);
7         System.out.println("Wrapped short value: " + wrappedNumber);
8     }
9 }
10
```

Console ×

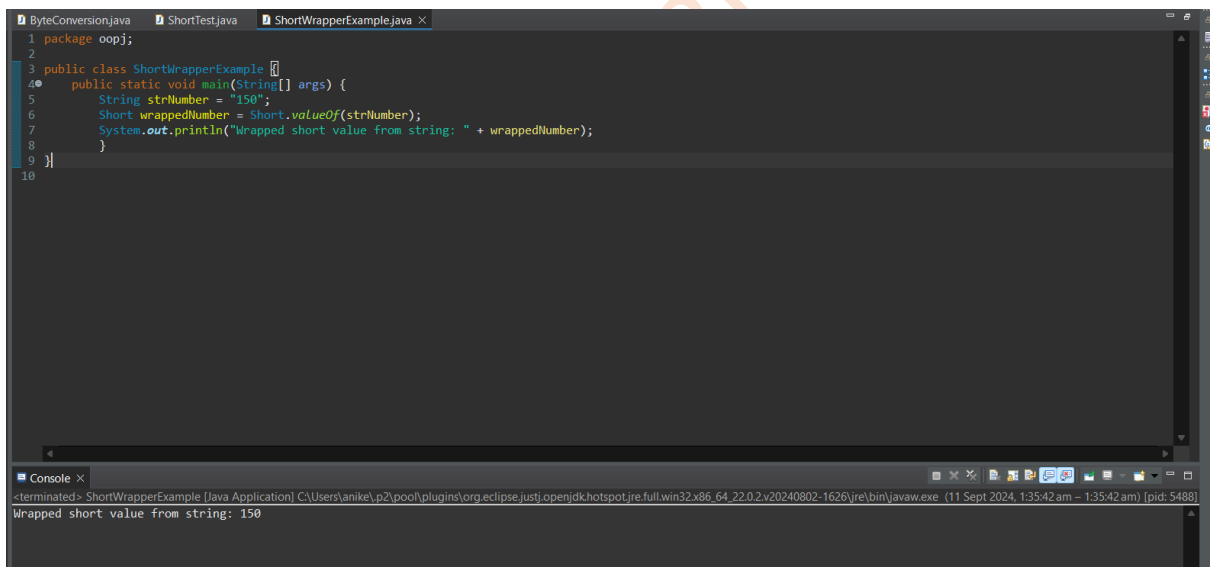
<terminated> ShortWrapperExample [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64.22.0.2.v20240802-1626\jre\bin\javaw.exe (11 Sept 2024, 1:31:29 am – 1:31:29 am) [pid: 7336]

Wrapped short value: 100

## ASSIGNMENT NO.2

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)`).

```
public class ShortWrapperExample {  
  
    public static void main(String[] args) {  
  
        String strNumber = "150";  
  
        Short wrappedNumber = Short.valueOf(strNumber);  
  
        System.out.println("Wrapped short value from string: " +  
wrappedNumber);  
  
    }  
  
}
```



```
1 package oopj;  
2  
3 public class ShortWrapperExample {  
4     public static void main(String[] args) {  
5         String strNumber = "150";  
6         Short wrappedNumber = Short.valueOf(strNumber);  
7         System.out.println("Wrapped short value from string: " + wrappedNumber);  
8     }  
9 }  
10
```

Console X  
-terminated> ShortWrapperExample [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64.22.0.2.v20240802-1626\jre\bin\javaw.exe (11 Sept 2024, 1:35:42 am - 1:35:42 am) [pid: 5488]  
Wrapped short value from string: 150

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

```
public class ShortTest {  
  
    public static void main(String[] args) {  
  
        short number = 120;  
  
        int intValue = number;  
  
        float floatValue = number;
```

```
double doubleValue = number;

long longValue = number;

System.out.println("Short to int: " + intValue);

System.out.println("Short to float: " + floatValue);

System.out.println("Short to double: " + doubleValue);

System.out.println("Short to long: " + longValue);

int intToShort = 200;

short convertedShort = (short) intToShort;

System.out.println("Int to short: " + convertedShort);

}}
```

## ASSIGNMENT NO.2

```
ByteConversion.java  ShortTest.java ×  ShortWrapperExample.java
1 package oopj;
2
3 public class ShortTest {
4     public static void main(String[] args) {
5         short number = 120;
6         int intValue = number;
7         float floatValue = number;
8         double doubleValue = number;
9         long longValue = number;
10        System.out.println("Short to int: " + intValue);
11        System.out.println("Short to float: " + floatValue);
12        System.out.println("Short to double: " + doubleValue);
13        System.out.println("Short to long: " + longValue);
14        int intToShort = 200;
15        short convertedShort = (short) intToShort;
16        System.out.println("Int to short: " + convertedShort);
17    }
18
19
20
```

```
Console ×
<terminated> ShortTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64
Short to int: 120
Short to float: 120.0
Short to double: 120.0
Short to long: 120
Int to short: 200
```

### 4. Working with `java.lang.Integer`

a. Explore the [Java API documentation for `java.lang.Integer`](#) and observe its modifiers and super types.

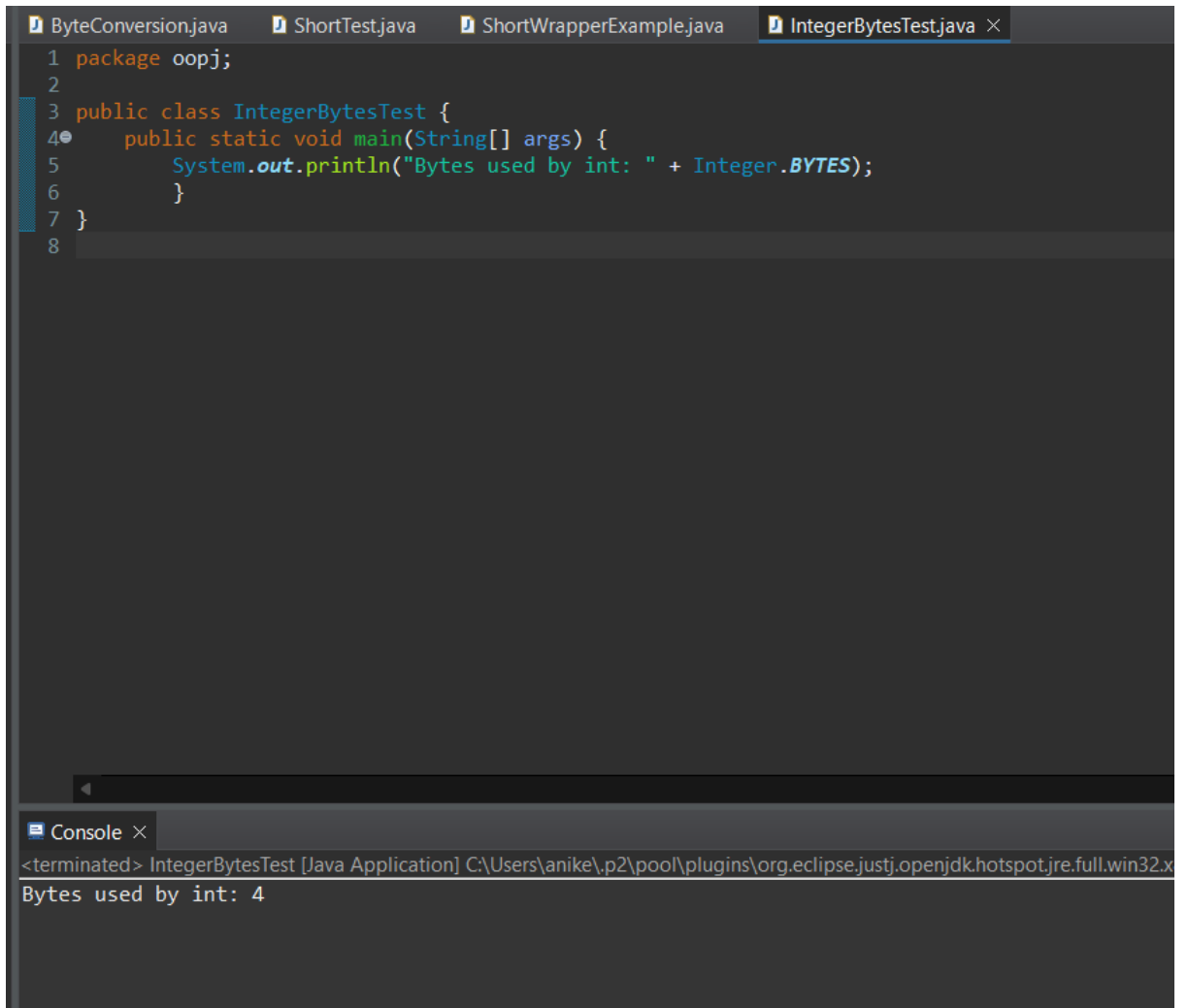
Modifier and Type	Field and Description
static int	BYTES The number of bytes used to represent an int value in two's complement binary form.
static int	MAX_VALUE A constant holding the maximum value an int can have, $2^{31}-1$ .
static int	MIN_VALUE A constant holding the minimum value an int can have, $-2^{31}$ .
static int	SIZE The number of bits used to represent an int value in two's complement binary form.
static Class<Integer>	TYPE The Class instance representing the primitive type int.

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

```
public class IntegerBytesTest {
```

## ASSIGNMENT NO.2

```
public static void main(String[] args) {  
  
    System.out.println("Bytes used by int: " + Integer.BYTES);  
  
}  
  
}
```



The screenshot shows an IDE with four tabs: ByteConversion.java, ShortTest.java, ShortWrapperExample.java, and IntegerBytesTest.java. The IntegerBytesTest.java tab is active, showing the following code:

```
1 package oopj;  
2  
3 public class IntegerBytesTest {  
4     public static void main(String[] args) {  
5         System.out.println("Bytes used by int: " + Integer.BYTES);  
6     }  
7 }  
8
```

Below the code editor is a console window with the following output:

```
<terminated> IntegerBytesTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x  
Bytes used by int: 4
```

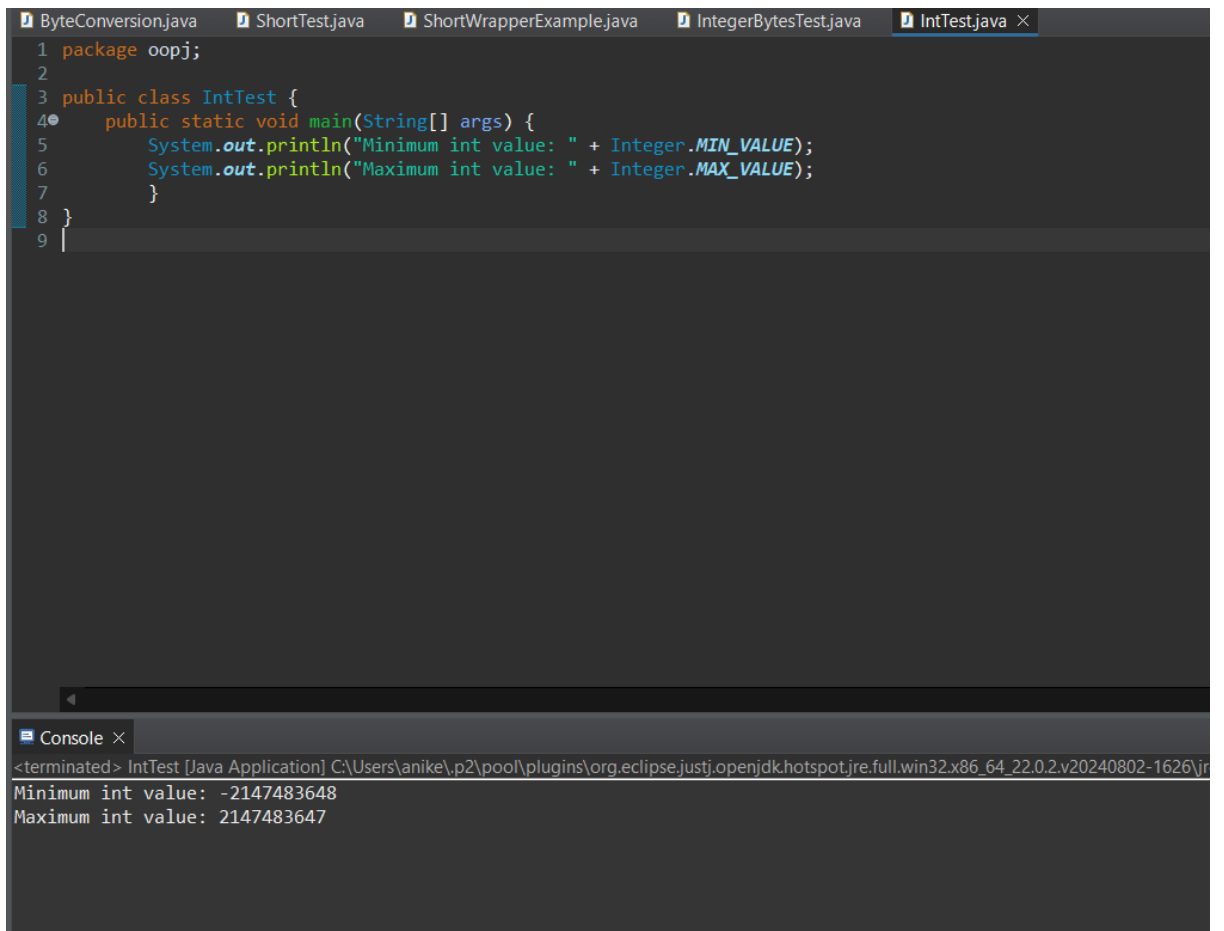
c. Write a program to find the minimum and maximum values of `int` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Integer.MIN_VALUE` and `Integer.MAX_VALUE`).

```
public class IntTest {  
  
    public static void main(String[] args) {  
  
        System.out.println("Minimum int value: " + Integer.MIN_VALUE);  
  
        System.out.println("Maximum int value: " + Integer.MAX_VALUE);  
  
    }  
}
```

## ASSIGNMENT NO.2

```
}
```

```
}
```



The screenshot shows an Eclipse IDE with several open files: ByteConversion.java, ShortTest.java, ShortWrapperExample.java, IntegerBytesTest.java, and IntTest.java. The IntTest.java file is active and shows the following code:

```
1 package oopj;  
2  
3 public class IntTest {  
4     public static void main(String[] args) {  
5         System.out.println("Minimum int value: " + Integer.MIN_VALUE);  
6         System.out.println("Maximum int value: " + Integer.MAX_VALUE);  
7     }  
8 }  
9
```

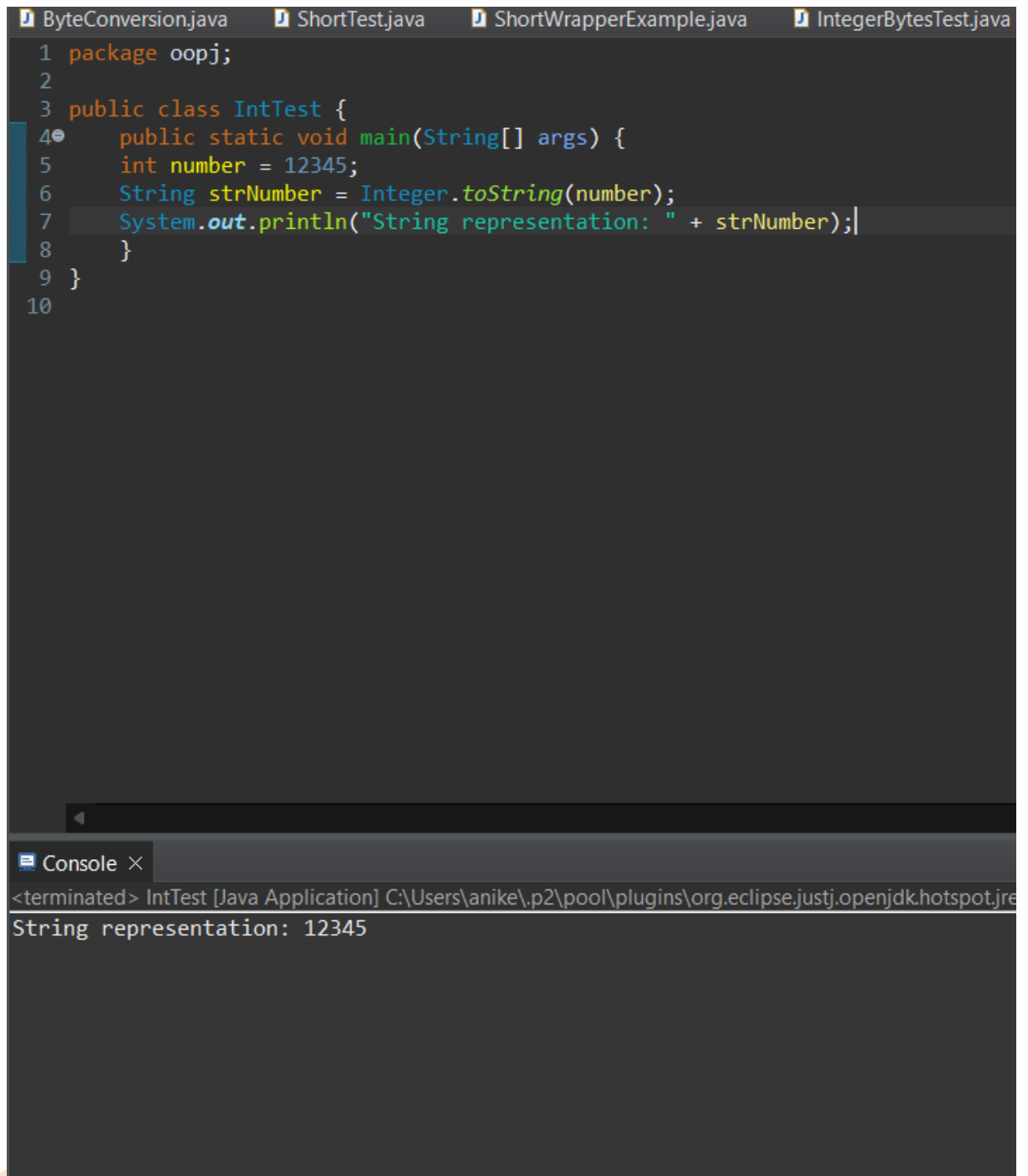
Below the code editor, the Console window is open, showing the output of the program:

```
<terminated> IntTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre  
Minimum int value: -2147483648  
Maximum int value: 2147483647
```

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)`).

```
public class IntTest {  
  
    public static void main(String[] args) {  
  
        int number = 12345;  
  
        String strNumber = Integer.toString(number);  
  
        System.out.println("String representation: " + strNumber);  
  
    }  
}
```

## ASSIGNMENT NO.2



The screenshot shows the Eclipse IDE with four tabs: ByteConversion.java, ShortTest.java, ShortWrapperExample.java, and IntegerBytesTest.java. The active tab is ShortTest.java, which contains the following code:

```
1 package oopj;
2
3 public class IntTest {
4     public static void main(String[] args) {
5         int number = 12345;
6         String strNumber = Integer.toString(number);
7         System.out.println("String representation: " + strNumber);
8     }
9 }
10
```

Below the code editor is the Console window, which shows the output of the program:

```
<terminated> IntTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre
String representation: 12345
```

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)`).

```
public class IntTest {

    public static void main(String[] args) {
```



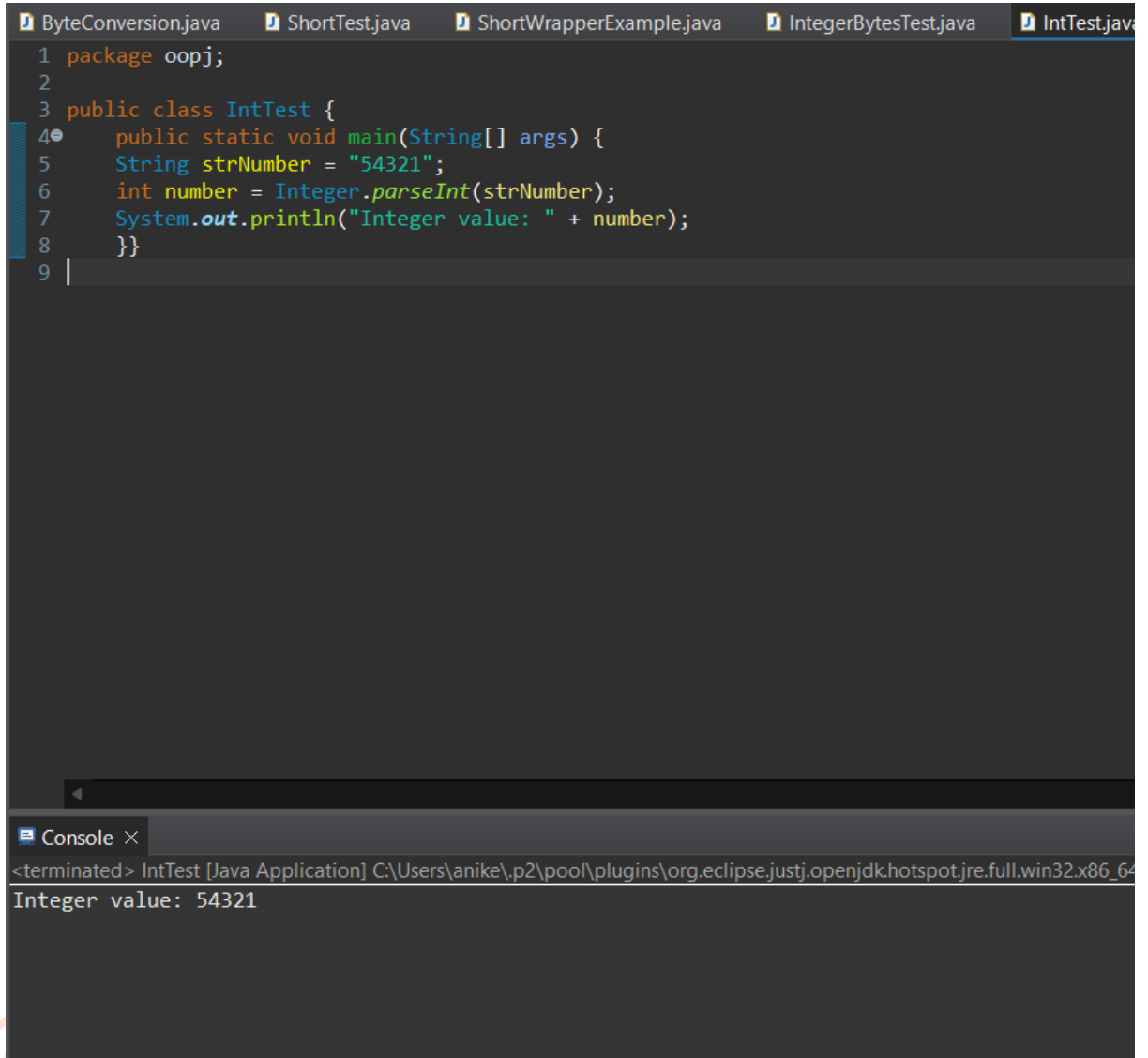
## ASSIGNMENT NO.2

```
String strNumber = "54321";

int number = Integer.parseInt(strNumber);

System.out.println("Integer value: " + number);

}}
```



The screenshot shows the Eclipse IDE with several Java files open in the top editor: ByteConversion.java, ShortTest.java, ShortWrapperExample.java, IntegerBytesTest.java, and IntTest.java. The IntTest.java file is selected and shows the following code:

```
1 package oopj;
2
3 public class IntTest {
4     public static void main(String[] args) {
5         String strNumber = "54321";
6         int number = Integer.parseInt(strNumber);
7         System.out.println("Integer value: " + number);
8     }
9 }
```

Below the code editor is the Console window, which displays the output of the program:

```
<terminated> IntTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64
Integer value: 54321
```

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`).

```
public class IntTest {

    public static void main(String[] args) {
```

## ASSIGNMENT NO.2

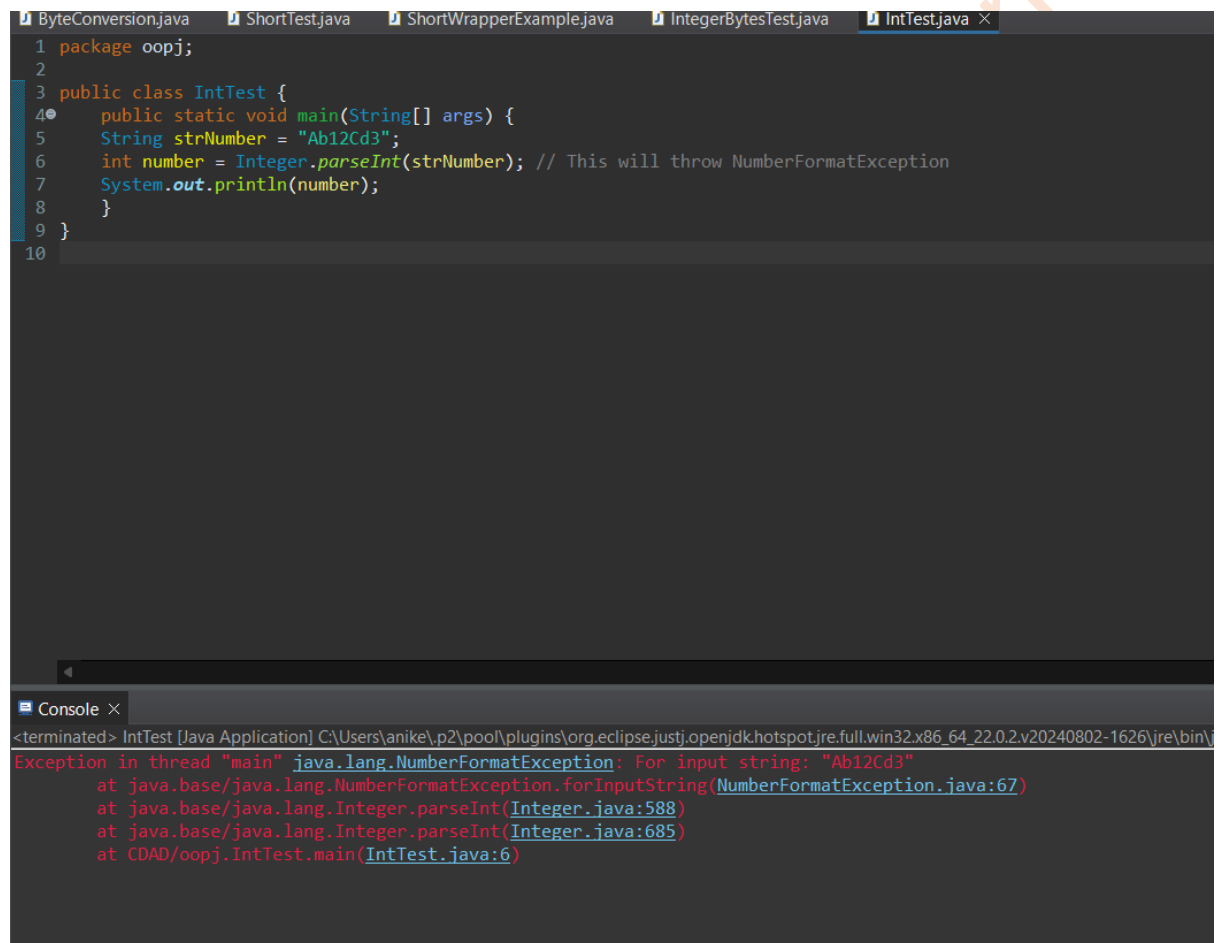
```
String strNumber = "Ab12Cd3";

int number = Integer.parseInt(strNumber); // This will throw
NumberFormatException

System.out.println(number);

}

}
```



The screenshot shows an IDE with several tabs: ByteConversion.java, ShortTest.java, ShortWrapperExample.java, IntegerBytesTest.java, and IntTest.java. The IntTest.java tab is active, showing the following code:

```
1 package oopj;
2
3 public class IntTest {
4     public static void main(String[] args) {
5         String strNumber = "Ab12Cd3";
6         int number = Integer.parseInt(strNumber); // This will throw NumberFormatException
7         System.out.println(number);
8     }
9 }
10
```

Below the code editor is a console window showing the following error message:

```
<terminated> IntTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\bin\
Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
    at java.base/java.lang.Integer.parseInt(Integer.java:588)
    at java.base/java.lang.Integer.parseInt(Integer.java:685)
    at CDAD/ooj.IntTest.main(IntTest.java:6)
```

**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)`).

```
public class IntTest {

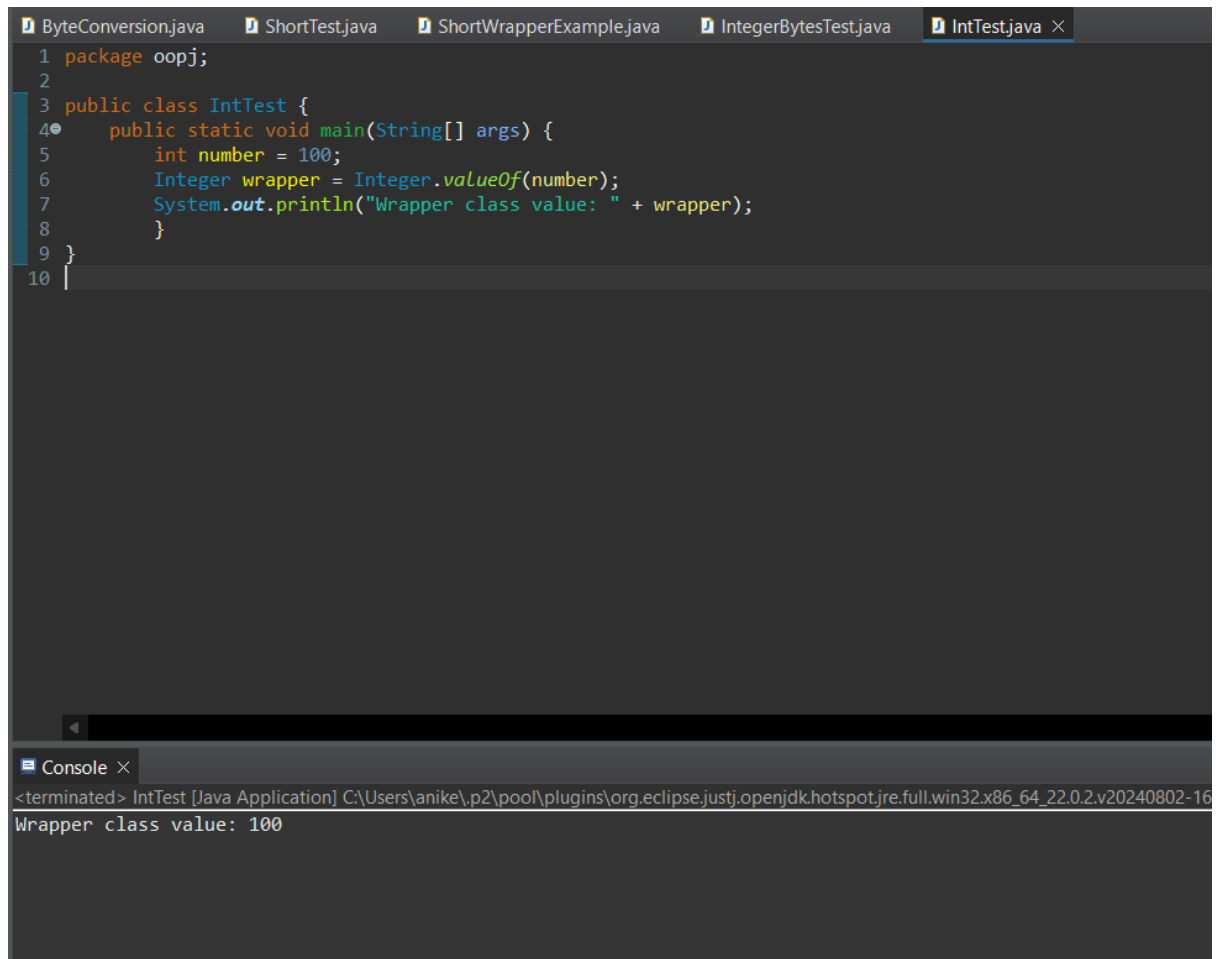
    public static void main(String[] args) {

        int number = 100;

        Integer wrapper = Integer.valueOf(number);
```

## ASSIGNMENT NO.2

```
        System.out.println("Wrapper class value: " + wrapper);  
    }  
}
```



The screenshot shows an IDE with several tabs: ByteConversion.java, ShortTest.java, ShortWrapperExample.java, IntegerBytesTest.java, and IntTest.java. The IntTest.java tab is active, showing the following code:

```
1 package oopj;  
2  
3 public class IntTest {  
4     public static void main(String[] args) {  
5         int number = 100;  
6         Integer wrapper = Integer.valueOf(number);  
7         System.out.println("Wrapper class value: " + wrapper);  
8     }  
9 }  
10
```

Below the code editor is a console window titled "Console ×" showing the output of the program:

```
<terminated> IntTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-16  
Wrapper class value: 100
```

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)`).

```
public class IntTest {  
  
    public static void main(String[] args) {  
  
        String strNumber = "200";  
  
        Integer wrapper = Integer.valueOf(strNumber);  
  
        System.out.println("Wrapper class value: " + wrapper);  
  
    }  
}
```

}

```

1 package oopj;
2
3 public class IntTest {
4     public static void main(String[] args) {
5         String strNumber = "200";
6         Integer wrapper = Integer.valueOf(strNumber);
7         System.out.println("Wrapper class value: " + wrapper);
8     }
9 }
10

```

Console ×

<terminated> IntTest [Java Application] C:\Users\anike\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1  
 Wrapper class value: 200

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)`).

```

public class IntTest {

    public static void main(String[] args) {

        int num1 = 10;

        int num2 = 20;

        int sum = Integer.sum(num1, num2);

        System.out.println("Sum: " + sum);

    }
}

```

```
1 package oopj;  
2  
3 public class IntTest {  
4     public static void main(String[] args) {  
5         int num1 = 10;  
6         int num2 = 20;  
7         int sum = Integer.sum(num1, num2);  
8         System.out.println("Sum: " + sum);  
9     }  
10 }
```

Console ×

<terminated> IntTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.  
Sum: 30

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)`).

```
public class IntTest {  
    public static void main(String[] args) {  
        int num1 = 10;  
        int num2 = 20;  
        System.out.println("Minimum: " + Integer.min(num1, num2));  
        System.out.println("Maximum: " + Integer.max(num1, num2));  
    }  
}
```

## ASSIGNMENT NO.2

```
ByteConversion.java  ShortTest.java  ShortWrapperExample.java  IntegerByteTest.java  IntTest.java
1 package oopj;
2
3 public class IntTest {
4     public static void main(String[] args) {
5         int num1 = 10;
6         int num2 = 20;
7         System.out.println("Minimum: " + Integer.min(num1, num2));
8         System.out.println("Maximum: " + Integer.max(num1, num2));
9     }
10 }
11
```

Console ×

<terminated> IntTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802

Minimum: 10  
Maximum: 20

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.toString(int)`, `Integer.toOctalString(int)`, and `Integer.toHexString(int)`).

```
public class IntTest {

    public static void main(String[] args) {

        int number = 7;

        System.out.println("Binary: " +
Integer.toString(number));

        System.out.println("Octal: " + Integer.toOctalString(number));

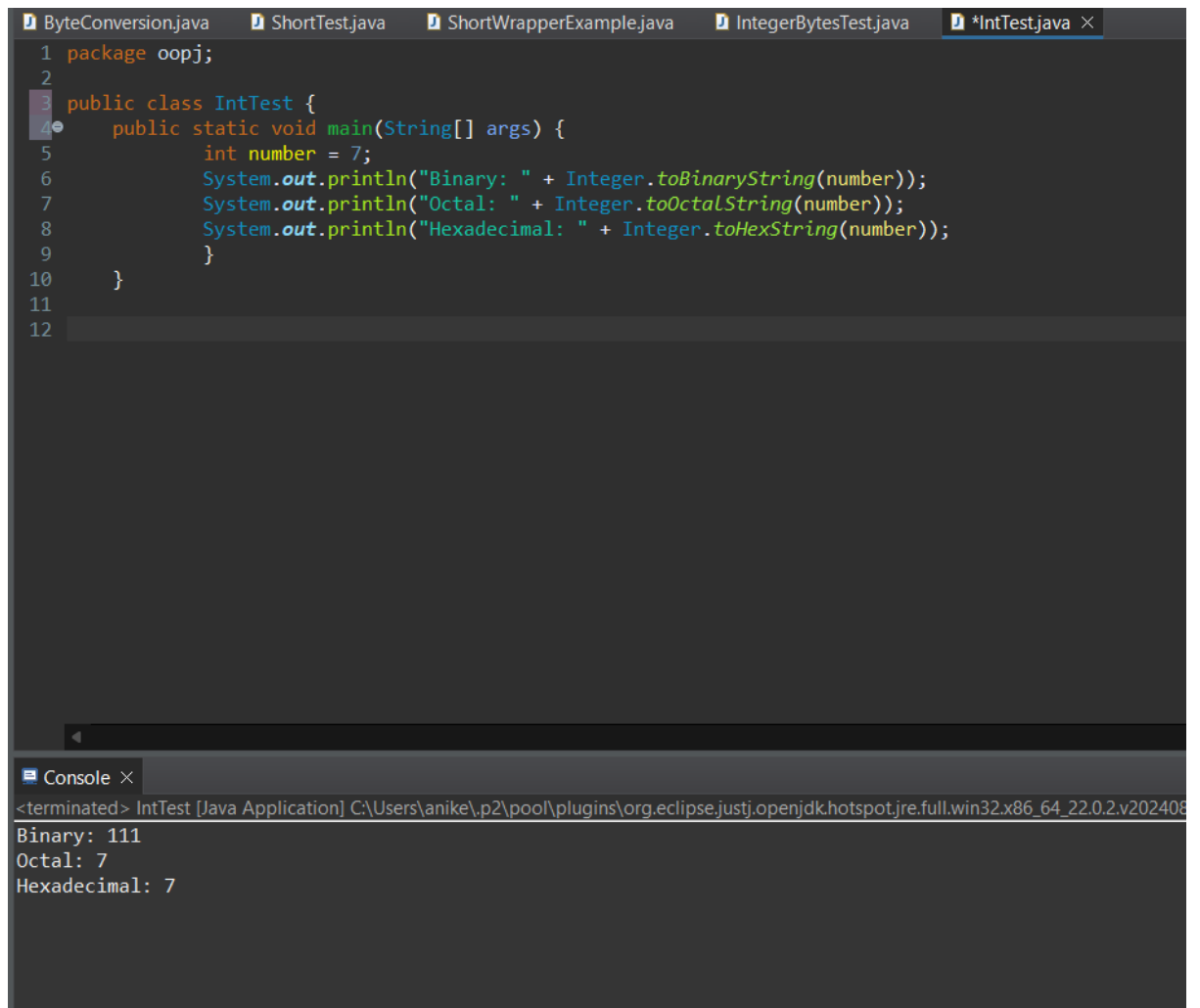
        System.out.println("Hexadecimal: " +
Integer.toHexString(number));

    }

}
```

## ASSIGNMENT NO.2

}



The screenshot shows an IDE with several tabs: ByteConversion.java, ShortTest.java, ShortWrapperExample.java, IntegerBytesTest.java, and \*IntTest.java. The active tab is \*IntTest.java, which contains the following code:

```
1 package oopj;
2
3 public class IntTest {
4     public static void main(String[] args) {
5         int number = 7;
6         System.out.println("Binary: " + Integer.toBinaryString(number));
7         System.out.println("Octal: " + Integer.toOctalString(number));
8         System.out.println("Hexadecimal: " + Integer.toHexString(number));
9     }
10 }
11
12
```

Below the code editor is a console window titled 'Console' showing the output of the program:

```
<terminated> IntTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v202408
Binary: 111
Octal: 7
Hexadecimal: 7
```

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

```
public class IntTest {

    public static void main(String[] args) {

        int number = 100;

        // Convert int to long

        long longValue = (long) number;

        System.out.println("Long value: " + longValue);

        // Convert int to double
```

## ASSIGNMENT NO.2

```
double doubleValue = (double) number;

System.out.println("Double value: " + doubleValue);

// Convert int to float

float floatValue = (float) number;

System.out.println("Float value: " + floatValue);

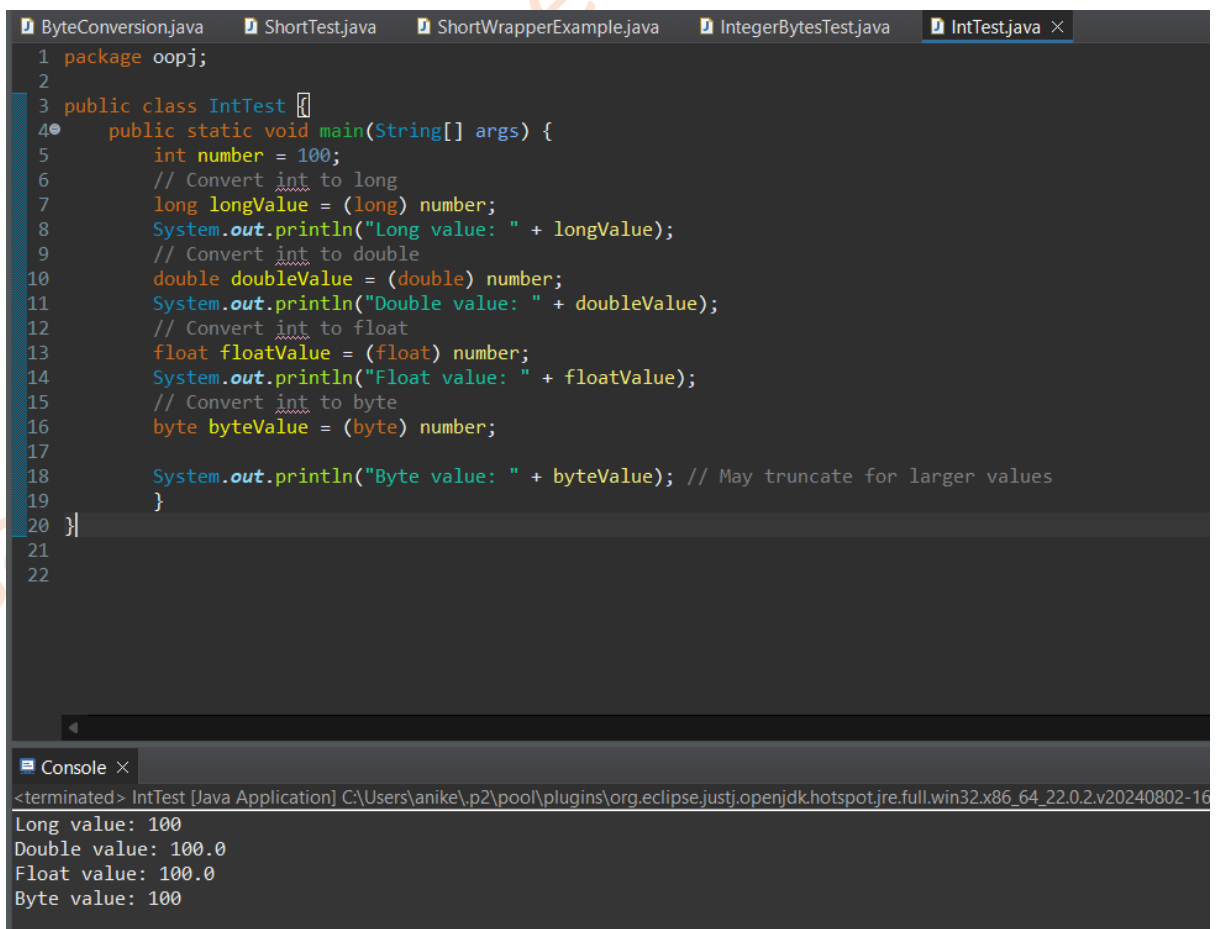
// Convert int to byte

byte byteValue = (byte) number;

System.out.println("Byte value: " + byteValue); // May truncate for
larger values

}

}
```



The screenshot shows an IDE with several tabs: ByteConversion.java, ShortTest.java, ShortWrapperExample.java, IntegerBytesTest.java, and IntTest.java. The IntTest.java tab is active, displaying the following code:

```
1 package oopj;
2
3 public class IntTest {
4     public static void main(String[] args) {
5         int number = 100;
6         // Convert int to long
7         long longValue = (long) number;
8         System.out.println("Long value: " + longValue);
9         // Convert int to double
10        double doubleValue = (double) number;
11        System.out.println("Double value: " + doubleValue);
12        // Convert int to float
13        float floatValue = (float) number;
14        System.out.println("Float value: " + floatValue);
15        // Convert int to byte
16        byte byteValue = (byte) number;
17
18        System.out.println("Byte value: " + byteValue); // May truncate for larger values
19    }
20 }
21
22
```

Below the code editor is a console window titled "Console" showing the output of the program:

```
<terminated> IntTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-16
Long value: 100
Double value: 100.0
Float value: 100.0
Byte value: 100
```



## 5. Working with `java.lang.Long`

a. Explore the [Java API documentation for `java.lang.Long`](#) and observe its modifiers and super types.

Modifier and Type	Field and Description
static int	<b>BYTES</b> The number of bytes used to represent a <code>float</code> value.
static int	<b>MAX_EXPONENT</b> Maximum exponent a finite <code>float</code> variable may have.
static float	<b>MAX_VALUE</b> A constant holding the largest positive finite value of type <code>float</code> , $(2 \cdot 2^{23}) \cdot 2^{127}$ .
static int	<b>MIN_EXPONENT</b> Minimum exponent a normalized <code>float</code> variable may have.
static float	<b>MIN_NORMAL</b> A constant holding the smallest positive normal value of type <code>float</code> , $2^{126}$ .
static float	<b>MIN_VALUE</b> A constant holding the smallest positive nonzero value of type <code>float</code> , $2^{149}$ .
static float	<b>NaN</b> A constant holding a Not-a-Number (NaN) value of type <code>float</code> .
static float	<b>NEGATIVE_INFINITY</b> A constant holding the negative infinity of type <code>float</code> .
static float	<b>POSITIVE_INFINITY</b> A constant holding the positive infinity of type <code>float</code> .
static int	<b>SIZE</b> The number of bits used to represent a <code>float</code> value.
static Class<Float>	<b>TYPE</b> The Class instance representing the primitive type <code>float</code> .

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.

```
public class LongOperationTest {

    public static void main(String[] args) {

        // b. Test how many bytes are used to represent a long value

        System.out.println("Bytes used to represent a long: " + Long.BYTES);

        // c. Find the minimum and maximum values of long

        System.out.println("Minimum long value: " + Long.MIN_VALUE);

        System.out.println("Maximum long value: " + Long.MAX_VALUE);

        // d. Convert a long value to a String using Long.toString

        long number = 123456789L;

        String strNumber = Long.toString(number);

        System.out.println("String representation of long: " + strNumber);

        // e. Convert a String to a long value using Long.parseLong

        String validStrNumber = "987654321";

        long parsedLong = Long.parseLong(validStrNumber);

        System.out.println("Parsed long value from string: " + parsedLong);

        // f. Attempt to convert an invalid String to a long value (will throw
        NumberFormatException)

        String invalidStrNumber = "Ab12Cd3";
```

```
try {  
  
    long invalidParsedLong = Long.parseLong(invalidStrNumber); // This  
will throw an exception  
  
    System.out.println(invalidParsedLong);  
  
} catch (NumberFormatException e) {  
  
    System.out.println("Exception: " + e.getMessage());  
  
}  
  
// g. Convert a long value to the corresponding wrapper class using  
Long.valueOf(long)  
  
Long longWrapper = Long.valueOf(number);  
  
System.out.println("Wrapper class value from long: " + longWrapper);  
  
// h. Convert a String to the corresponding wrapper class using  
Long.valueOf(String)  
  
Long wrapperFromString = Long.valueOf(validStrNumber);  
  
System.out.println("Wrapper class value from string: " +  
wrapperFromString);  
  
// i. Add two long values using Long.sum  
  
long num1 = 1123L;  
long num2 = 9845L;  
  
long sum = Long.sum(num1, num2);  
  
System.out.println("Sum of two longs: " + sum);  
  
// j. Find the minimum and maximum of two long values using  
Long.min and Long.max  
  
long num3 = 1122L;  
  
long num4 = 5566L;  
  
System.out.println("Minimum of two longs: " + Long.min(num3,  
num4));
```

```

        System.out.println("Maximum of two longs: " + Long.max(num3,
num4));

        // k. Convert a long value to binary, octal, and hexadecimal strings

        long value = 7L;

        System.out.println("Binary representation: " +
Long.toBinaryString(value));

        System.out.println("Octal representation: " +
Long.toOctalString(value));

        System.out.println("Hexadecimal representation: " +
Long.toHexString(value));

        // l. Experiment with converting a long value into other primitive
types

        // Convert long to int

        int intValue = (int) number;

        System.out.println("Converted to int: " + intValue);

        // Convert long to double

        double doubleValue = (double) number;

        System.out.println("Converted to double: " + doubleValue);

        // Convert long to float

        float floatValue = (float) number;

        System.out.println("Converted to float: " + floatValue);

        // Convert long to byte (might truncate if too large)

        byte byteValue = (byte) number;

        System.out.println("Converted to byte: " + byteValue);

    }

}

```

```

ByteConversion.java ShortTest.java ShortWrapperExample.java IntegerBytesTest.java IntTest.java LongOperationTest.java
3 public class LongOperationTest {
4     public static void main(String[] args) {
5         // b. Test how many bytes are used to represent a long value
6         System.out.println("Bytes used to represent a long: " + Long.BYTES);
7         // c. Find the minimum and maximum values of long
8         System.out.println("Minimum long value: " + Long.MIN_VALUE);
9         System.out.println("Maximum long value: " + Long.MAX_VALUE);
10        // d. Convert a long value to a String using Long.toString
11        long number = 123456789L;
12        String strNumber = Long.toString(number);
13        System.out.println("String representation of long: " + strNumber);
14        // e. Convert a String to a long value using Long.parseLong
15        String validStrNumber = "987654321";
16        long parsedLong = Long.parseLong(validStrNumber);
17        System.out.println("Parsed long value from string: " + parsedLong);
18
19        // f. Attempt to convert an invalid String to a long value (will throw NumberFormatException)
20        String invalidStrNumber = "Ab12Cd3";
21        try {
22            long invalidParsedLong = Long.parseLong(invalidStrNumber); // This will throw an exception
23            System.out.println(invalidParsedLong);
24        } catch (NumberFormatException e) {
25            System.out.println("Exception: " + e.getMessage());
26        }
27        // g. Convert a long value to the corresponding wrapper class using Long.valueOf(long)
28        Long longWrapper = Long.valueOf(number);
29        System.out.println("Wrapper class value from long: " + longWrapper);
    }
}

```

```

<terminated> LongOperationTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802
Sum of two longs: 10968
Minimum of two longs: 1122
Maximum of two longs: 5566
Binary representation: 111
Octal representation: 7
Hexadecimal representation: 7
Converted to int: 123456789
Converted to double: 1.23456789E8
Converted to float: 1.2345679E8
Converted to byte: 21
    
```

## 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.

```
public class FloatTest {

    public static void main(String[] args) {

        // b. Test how many bytes are used to represent a float value

        System.out.println("Bytes used to represent a float: " + Float.BYTES);

        // c. Find the minimum and maximum values of float

        System.out.println("Minimum float value: " + Float.MIN_VALUE);

        System.out.println("Maximum float value: " + Float.MAX_VALUE);

        // d. Convert a float value to a String using Float.toString

        float number = 123.45f;

        String strNumber = Float.toString(number);
```

```

System.out.println("String representation of float: " + strNumber);

// e. Convert a String to a float value using Float.parseFloat

String validStrNumber = "987.65";

float parsedFloat = Float.parseFloat(validStrNumber);

System.out.println("Parsed float value from string: " + parsedFloat);

// f. Attempt to convert an invalid String to a float value (will throw
NumberFormatException)

String invalidStrNumber = "Ab12Cd3";

try {

float invalidParsedFloat = Float.parseFloat(invalidStrNumber);

System.out.println(invalidParsedFloat);

} catch (NumberFormatException e) {

System.out.println("Exception: " + e.getMessage());

}

// g. Convert a float value to the corresponding wrapper class using
Float.valueOf(float)

Float floatWrapper = Float.valueOf(number);

System.out.println("Wrapper class value from float: " +
floatWrapper);

// h. Convert a String to the corresponding wrapper class using
Float.valueOf(String)

Float wrapperFromString = Float.valueOf(validStrNumber);

System.out.println("Wrapper class value from string: " +
wrapperFromString);

// i. Add two float values using Float.sum

float num1 = 112.3f;

float num2 = 984.5f;

```

## ASSIGNMENT NO.2

```
float sum = Float.sum(num1, num2);

System.out.println("Sum of two floats: " + sum);

// j. Find the minimum and maximum of two float values using
Float.min and Float.max

float num3 = 112.2f;

float num4 = 556.6f;

System.out.println("Minimum of two floats: " + Float.min(num3,
num4));

System.out.println("Maximum of two floats: " + Float.max(num3,
num4));

// k. Find the square root of a negative float value

float negativeValue = -25.0f;

double sqrtValue = Math.sqrt(negativeValue); // Math.sqrt returns a
double

System.out.println("Square root of " + negativeValue + ": " +
sqrtValue);

// l. Divide two float variables with the same value of 0.0f and observe
the result

float zero1 = 0.0f;

float zero2 = 0.0f;

float divisionResult = zero1 / zero2;

System.out.println("Result of dividing 0.0f by 0.0f: " + divisionResult);

// m. Experiment with converting a float value into other primitive
types

// Convert float to int

int intValue = (int) number;

System.out.println("Converted to int: " + intValue);

// Convert float to long
```



## ASSIGNMENT NO.2

```

        long longValue = (long) number;

        System.out.println("Converted to long: " + longValue);

        // Convert float to double

        double doubleValue = (double) number;

        System.out.println("Converted to double: " + doubleValue);

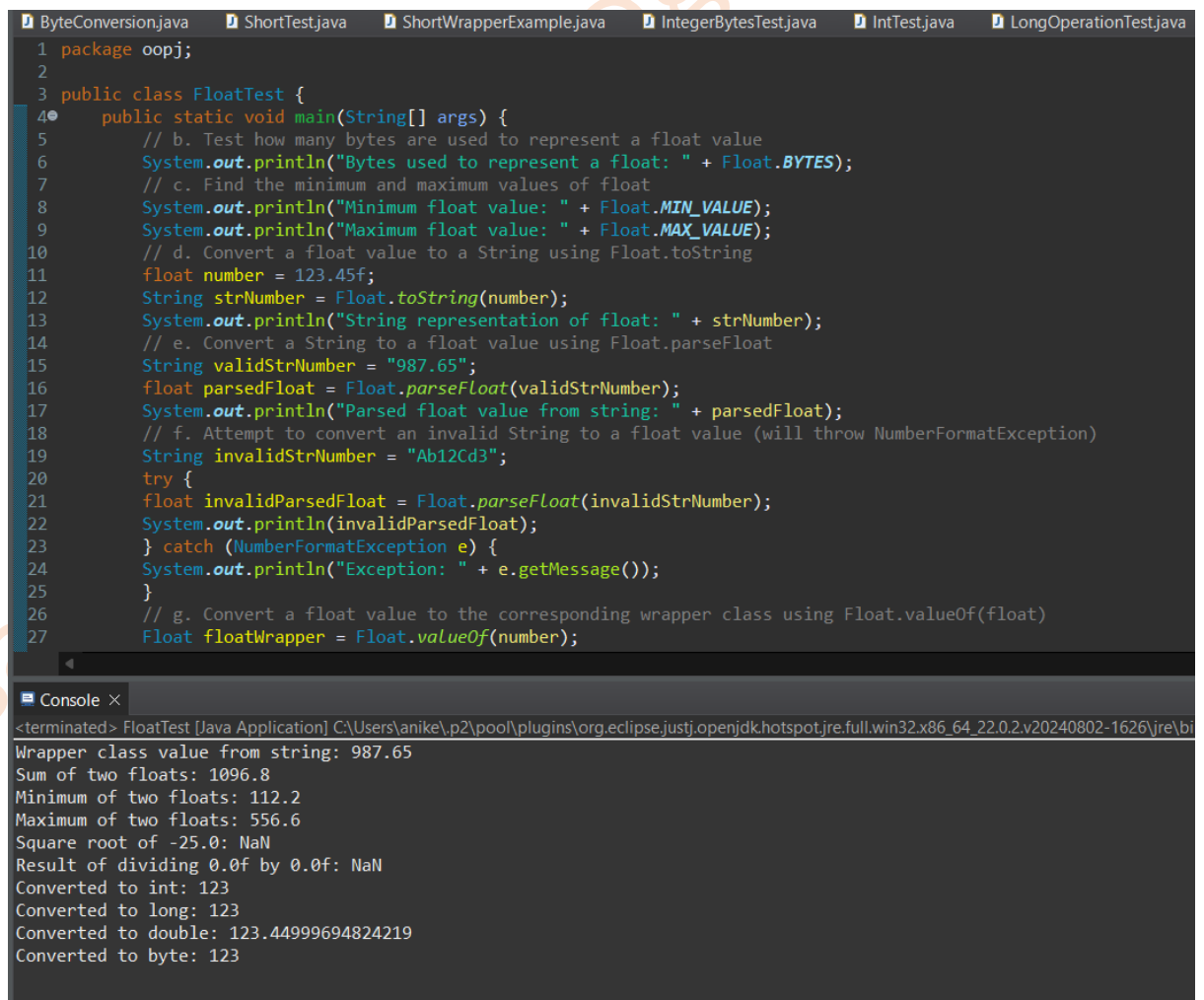
        // Convert float to byte (may truncate)

        byte byteValue = (byte) number;

        System.out.println("Converted to byte: " + byteValue);

    }
}

```



The screenshot shows an IDE with a Java project. The editor displays the `FloatTest` class in the `oopj` package. The code performs various float operations: printing byte representation, finding min/max values, converting float to string, parsing string to float, handling exceptions for invalid strings, and converting float to wrapper classes. The console output shows the results of these operations.

```

ByteConversion.java ShortTest.java ShortWrapperExample.java IntegerBytesTest.java IntTest.java LongOperationTest.java
1 package oopj;
2
3 public class FloatTest {
4     public static void main(String[] args) {
5         // b. Test how many bytes are used to represent a float value
6         System.out.println("Bytes used to represent a float: " + Float.BYTES);
7         // c. Find the minimum and maximum values of float
8         System.out.println("Minimum float value: " + Float.MIN_VALUE);
9         System.out.println("Maximum float value: " + Float.MAX_VALUE);
10        // d. Convert a float value to a String using Float.toString
11        float number = 123.45f;
12        String strNumber = Float.toString(number);
13        System.out.println("String representation of float: " + strNumber);
14        // e. Convert a String to a float value using Float.parseFloat
15        String validStrNumber = "987.65";
16        float parsedFloat = Float.parseFloat(validStrNumber);
17        System.out.println("Parsed float value from string: " + parsedFloat);
18        // f. Attempt to convert an invalid String to a float value (will throw NumberFormatException)
19        String invalidStrNumber = "Ab12Cd3";
20        try {
21            float invalidParsedFloat = Float.parseFloat(invalidStrNumber);
22            System.out.println(invalidParsedFloat);
23        } catch (NumberFormatException e) {
24            System.out.println("Exception: " + e.getMessage());
25        }
26        // g. Convert a float value to the corresponding wrapper class using Float.valueOf(float)
27        Float floatWrapper = Float.valueOf(number);
28    }
29 }

```

Console Output:

```

<terminated> FloatTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\bin
Wrapper class value from string: 987.65
Sum of two floats: 1096.8
Minimum of two floats: 112.2
Maximum of two floats: 556.6
Square root of -25.0: NaN
Result of dividing 0.0f by 0.0f: NaN
Converted to int: 123
Converted to long: 123
Converted to double: 123.44999694824219
Converted to byte: 123

```

## 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 112.2 and 556.6, and find the minimum and maximum values using the `Double` class. (Hint: Use `Double.min(double, double)` and `Double.max(double, double)`).
- k. Declare a `double` variable with the value -25.0. Find the square root of this value. (Hint: Use `Math.sqrt()` method).
- 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 `double` value into other primitive types or vice versa and observe the results.

```
public class DoubleOperationTest {  
  
    public static void main(String[] args) {  
  
        // b. Bytes used to represent a double value  
  
        System.out.println("Bytes used to represent a double value: " +  
Double.BYTES);  
  
        // c. Minimum and Maximum values of double  
  
        System.out.println("Minimum value of double: " +  
Double.MIN_VALUE);  
  
        System.out.println("Maximum value of double: " +  
Double.MAX_VALUE);  
  
        // d. Convert double to String  
  
        double number = 123.456;  
  
        String strNumber = Double.toString(number);  
  
        System.out.println("String representation: " + strNumber);  
  
        // e. Convert String to double  
  
        String strDouble = "456.789";  
  
        double parsedNumber = Double.parseDouble(strDouble);  
  
        System.out.println("Parsed double value: " + parsedNumber);  
  
        // f. Attempt to convert invalid String to double (will cause a  
        NumberFormatException, but not caught here)  
  
        String invalidStrNumber = "Ab12Cd3";  
  
        System.out.println("Attempt to parse invalid string to double: " +  
Double.parseDouble(invalidStrNumber)); // May throw exception  
  
        // g. Convert double to Wrapper class  
  
        Double wrapper = Double.valueOf(number);  
  
        System.out.println("Wrapper class value: " + wrapper);  
  
        // h. Convert String to Wrapper class
```

```
String strWrapper = "234.567";

Double wrapperFromString = Double.valueOf(strWrapper);

System.out.println("Wrapper class value from string: " +
wrapperFromString);

// i. Add two doubles using Double.sum

double num1 = 112.3;

double num2 = 984.5;

double sum = Double.sum(num1, num2);

System.out.println("Sum: " + sum);

// j. Find minimum and maximum of two doubles

double num3 = 112.2;

double num4 = 556.6;

double min = Double.min(num3, num4);

double max = Double.max(num3, num4);

System.out.println("Minimum: " + min);

System.out.println("Maximum: " + max);

// k. Find square root of a double value

double negativeNumber = -25.0;

double sqrtResult = Math.sqrt(negativeNumber);

System.out.println("Square root of negative number: " + sqrtResult);
// Will be NaN

// l. Divide two doubles with value 0.0

double zero1 = 0.0;

double zero2 = 0.0;

double divisionResult = zero1 / zero2;
```

```

System.out.println("Result of division 0.0 / 0.0: " + divisionResult); //
Will be NaN

// m. Convert double to other primitive types and vice versa

float floatValue = (float) number;

long longValue = (long) number;

int intValue = (int) number;

short shortValue = (short) number;

byte byteValue = (byte) number;

System.out.println("Double to float: " + floatValue);

System.out.println("Double to long: " + longValue);

System.out.println("Double to int: " + intValue);

System.out.println("Double to short: " + shortValue);

System.out.println("Double to byte: " + byteValue);

double backToDoubleFromFloat = floatValue;

double backToDoubleFromLong = longValue;

double backToDoubleFromInt = intValue;

double backToDoubleFromShort = shortValue;

double backToDoubleFromByte = byteValue;

System.out.println("Back to double from float: " +
backToDoubleFromFloat);

System.out.println("Back to double from long: " +
backToDoubleFromLong);

System.out.println("Back to double from int: " +
backToDoubleFromInt);

System.out.println("Back to double from short: " +
backToDoubleFromShort);

```

```

        System.out.println("Back to double from byte: " +
backToDoubleFromByte);

    }

}

```

```

IntegerBytesTest.java  IntTest.java  LongOperationTest.java  FloatTest.java  DoubleOperationTest.java x
46     double zero2 = 0.0;
47     double divisionResult = zero1 / zero2;
48     System.out.println("Result of division 0.0 / 0.0: " + divisionResult); // Will be NaN
49     // m. Convert double to other primitive types and vice versa
50     float floatValue = (float) number;
51     long longValue = (long) number;
52     int intValue = (int) number;
53     short shortValue = (short) number;
54     byte byteValue = (byte) number;
55     System.out.println("Double to float: " + floatValue);
56     System.out.println("Double to long: " + longValue);
57     System.out.println("Double to int: " + intValue);
58     System.out.println("Double to short: " + shortValue);
59     System.out.println("Double to byte: " + byteValue);
60     double backToDoubleFromFloat = floatValue;
61     double backToDoubleFromLong = longValue;
62     double backToDoubleFromInt = intValue;
63     double backToDoubleFromShort = shortValue;
64     double backToDoubleFromByte = byteValue;
65     System.out.println("Back to double from float: " + backToDoubleFromFloat);
66     System.out.println("Back to double from long: " + backToDoubleFromLong);
67     System.out.println("Back to double from int: " + backToDoubleFromInt);
68     System.out.println("Back to double from short: " + backToDoubleFromShort);
69     System.out.println("Back to double from byte: " + backToDoubleFromByte);
70 }
71 }
72 }

Console x
<terminated> DoubleOperationTest [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\bin\javaw.exe (11 S
Bytes used to represent a double value: 8
Minimum value of double: 4.9E-324
Maximum value of double: 1.7976931348623157E308
String representation: 123.456
Parsed double value: 456.789
Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3"
    at java.base/jdk.internal.math.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:2054)
    at java.base/jdk.internal.math.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
    at java.base/java.lang.Double.parseDouble(Double.java:938)
    at CDAD/oojp.DoubleOperationTest.main(DoubleOperationTest.java:20)

```

## 8. Conversion between Primitive Types and Strings

Initialize a variable of each primitive type with a user-defined value and convert it into String:

- First, use the `toString` method of the corresponding wrapper class. (e.g., `Integer.toString()`).
- Then, use the `valueOf` method of the `String` class. (e.g., `String.valueOf()`).

```
public class Conversion {
```

```
    public static void main(String[] args) {
```

```
        // Primitive types and their user-defined values
```

```
        int intValue = 42;
```

```
double doubleValue = 3.14159;

char charValue = 'A';

boolean booleanValue = true;

long longValue = 123456789L;

float floatValue = 9.81f;

short shortValue = 12345;

byte byteValue = 100;

// Using wrapper class toString method

System.out.println("Using wrapper class toString method:");

System.out.println("int to String: " +
Integer.toString(intValue));

System.out.println("double to String: " +
Double.toString(doubleValue));

System.out.println("char to String: " +
Character.toString(charValue));

System.out.println("boolean to String: " +
Boolean.toString(booleanValue));

System.out.println("long to String: " +
Long.toString(longValue));

System.out.println("float to String: " +
Float.toString(floatValue));

System.out.println("short to String: " +
Short.toString(shortValue));

System.out.println("byte to String: " +
Byte.toString(byteValue));

// Using String.valueOf method

System.out.println("\nUsing String.valueOf method:");

System.out.println("int to String: " + String.valueOf(intValue));
```

```
        System.out.println("double to String: " +  
String.valueOf(doubleValue));  
  
        System.out.println("char to String: " +  
String.valueOf(charValue));  
  
        System.out.println("boolean to String: " +  
String.valueOf(booleanValue));  
  
        System.out.println("long to String: " +  
String.valueOf(longValue));  
  
        System.out.println("float to String: " +  
String.valueOf(floatValue));  
  
        System.out.println("short to String: " +  
String.valueOf(shortValue));  
  
        System.out.println("byte to String: " +  
String.valueOf(byteValue));  
  
    }  
  
}
```



```

Conversion.java ×
10     long longValue = 123456789L;
11     float floatValue = 9.81f;
12     short shortValue = 12345;
13     byte byteValue = 100;
14     // Using wrapper class toString method
15     System.out.println("Using wrapper class toString method:");
16     System.out.println("int to String: " + Integer.toString(intValue));
17     System.out.println("double to String: " + Double.toString(doubleValue));
18     System.out.println("char to String: " + Character.toString(charValue));
19     System.out.println("boolean to String: " + Boolean.toString(booleanValue));
20     System.out.println("long to String: " + Long.toString(longValue));
21     System.out.println("float to String: " + Float.toString(floatValue));
22     System.out.println("short to String: " + Short.toString(shortValue));
23     System.out.println("byte to String: " + Byte.toString(byteValue));
24     // Using String.valueOf method
25     System.out.println("\nUsing String.valueOf method:");
26     System.out.println("int to String: " + String.valueOf(intValue));
27     System.out.println("double to String: " + String.valueOf(doubleValue));
28     System.out.println("char to String: " + String.valueOf(charValue));
29     System.out.println("boolean to String: " + String.valueOf(booleanValue));
30     System.out.println("long to String: " + String.valueOf(longValue));
31     System.out.println("float to String: " + String.valueOf(floatValue));
32     System.out.println("short to String: " + String.valueOf(shortValue));
33     System.out.println("byte to String: " + String.valueOf(byteValue));
34     }
35 }
36

Console ×
<terminated> Conversion [Java Application] C:\Users\anike\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.

Using String.valueOf method:
int to String: 42
double to String: 3.14159
char to String: A
boolean to String: true
long to String: 123456789
float to String: 9.81
short to String: 12345
byte to String: 100

```

## 9. Default Values of Primitive Types

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).

```

public class TestConversion {

    // Instance variables (fields)

    int instanceInt;

    double instanceDouble;

    char instanceChar;

    boolean instanceBoolean;

```

```
long instanceLong;

float instanceFloat;

short instanceShort;

byte instanceByte;

// Static variables

static int staticInt;

static double staticDouble;

static char staticChar;

static boolean staticBoolean;

static long staticLong;

static float staticFloat;

static short staticShort;

static byte staticByte;

public static void main(String[] args) {

    // Create an instance of DefaultValuesTest

    TestConversion test = new TestConversion();

    // Print default values of instance variables

    System.out.println("Default values of instance variables:");

    System.out.println("int: " + test.instanceInt);

    System.out.println("double: " + test.instanceDouble);

    System.out.println("char: [" + test.instanceChar + "]");

    System.out.println("boolean: " + test.instanceBoolean);

    System.out.println("long: " + test.instanceLong);

    System.out.println("float: " + test.instanceFloat);
```

```
System.out.println("short: " + test.instanceShort);

System.out.println("byte: " + test.instanceByte);

// Print default values of static variables

System.out.println("\nDefault values of static variables:");


System.out.println("int: " + staticInt);

System.out.println("double: " + staticDouble);

System.out.println("char: [" + staticChar + "]");

System.out.println("boolean: " + staticBoolean);

System.out.println("long: " + staticLong);

System.out.println("float: " + staticFloat);

System.out.println("short: " + staticShort);

System.out.println("byte: " + staticByte);

}

}
```

```
<terminated> TestConversion [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre...
Default values of instance variables:
int: 0
double: 0.0
char: [ ]
boolean: false
long: 0
float: 0.0
short: 0
byte: 0

Default values of static variables:
int: 0
double: 0.0
char: [ ]
boolean: false
long: 0
float: 0.0
short: 0
byte: 0
```

## 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).

```
public class ArithmeticSwitch {

    public static void main(String[] args) {

        // Check if the number of arguments is exactly 3

        if (args.length != 3) {

            System.out.println("Usage: java ArithmeticOperationsCLI <num1>
<num2> <operator>");

            System.out.println("Example: java ArithmeticOperationsCLI 10 5 +");
```

```
return;

}

// Parse the command-line arguments

int num1 = Integer.parseInt(args[0]);

int num2 = Integer.parseInt(args[1]);

String operator = args[2];

// Variable to store the result of the operation

double result;

// Perform the arithmetic operation based on the operator

switch (operator) {

case "+":

    result = num1 + num2;

    break;

case "-":

    result = num1 - num2;

    break;

case "*":

    result = num1 * num2;

    break;

case "/":

    // Handle division by zero

    if (num2 == 0) {

        System.out.println("Error: Division by zero is not allowed.");
```

```
        return;
    }

    result = (double) num1 / num2;

    break;

default:

    System.out.println("Error: Invalid operator. Use +, -, *, or /.");

    return;

}

// Print the result

System.out.println("Result: " + result);

}

}
```

## ASSIGNMENT NO.2

```
Conversion.java  TestConversion.java  ArithmeticSwitch.java ×
1 package oopj;
2
3 public class ArithmeticSwitch {
4     public static void main(String[] args) {
5         // Check if the number of arguments is exactly 3
6         if (args.length != 3) {
7             System.out.println("Usage: java ArithmeticOperationsCLI <num1> <num2> <operator>");
8             System.out.println("Example: java ArithmeticOperationsCLI 10 5 +");
9             return;
10        }
11        // Parse the command-line arguments
12
13        int num1 = Integer.parseInt(args[0]);
14        int num2 = Integer.parseInt(args[1]);
15        String operator = args[2];
16        // Variable to store the result of the operation
17        double result;
18        // Perform the arithmetic operation based on the operator
19        switch (operator) {
20            case "+":
21                result = num1 + num2;
22                break;
23            case "-":
24                result = num1 - num2;
25                break;
26            case "*":
27                result = num1 * num2;
28                break;
29        }
30    }
31}
```

Console ×

<terminated> ArithmeticSwitch [Java Application] C:\Users\anike\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64

Usage: java ArithmeticOperationsCLI <num1> <num2> <operator>

Example: java ArithmeticOperationsCLI 10 5 +