

ASSIGNMENT NO.2

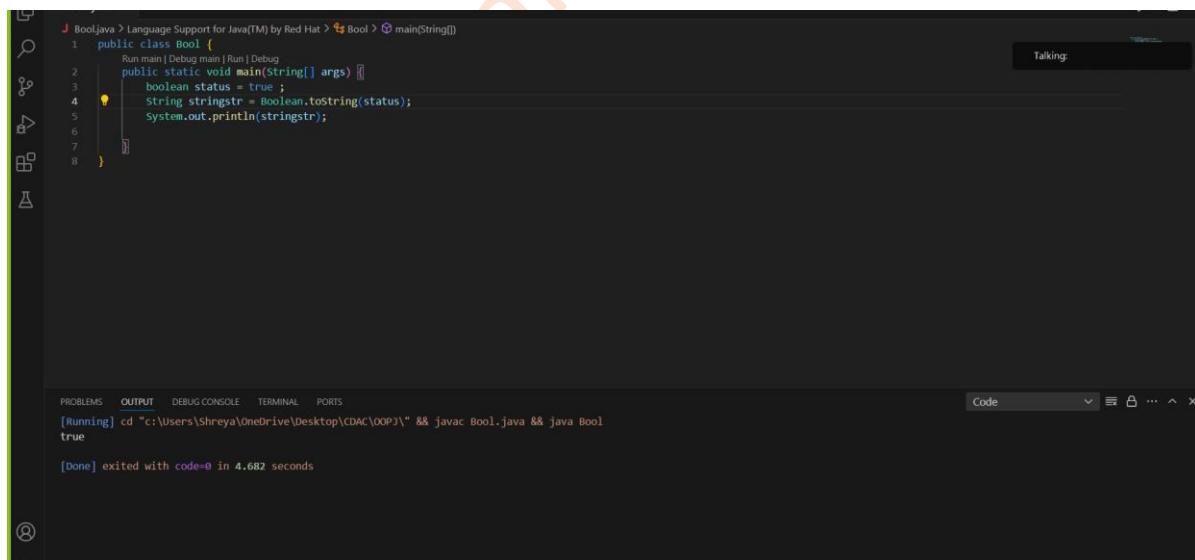
Note: Consider the following before starting the assignment:

- 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`

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



The screenshot shows a Java IDE interface with the following details:

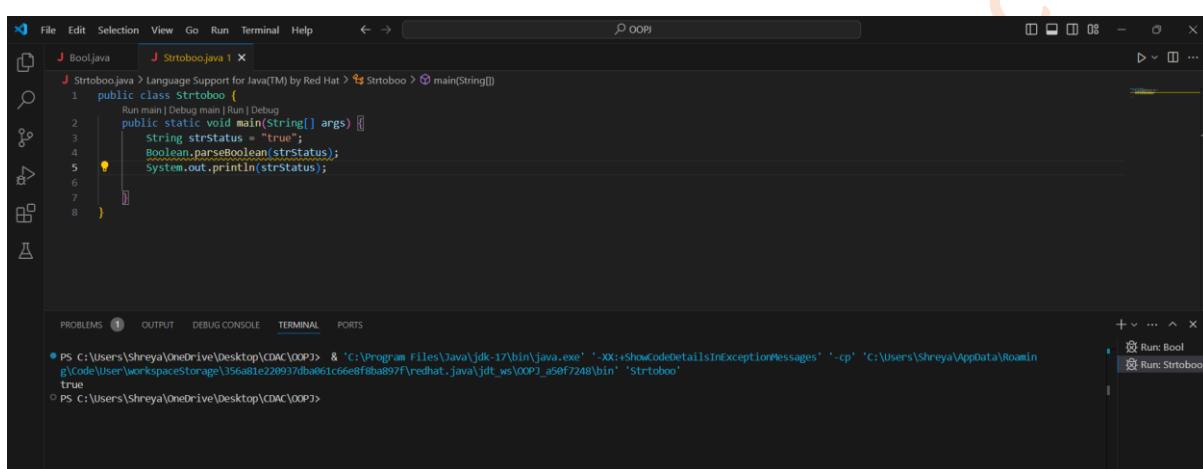
- Code Editor:** Displays the Java code for the `Bool` class.
- Terminal:** Shows the command-line output of running the code. It includes:
 - The command: `[Running] cd "c:\Users\Shreya\OneDrive\Desktop\CDAC\OOPJ\" && javac Bool.java && java Bool`
 - The output: `true`
 - The completion message: `[Done] exited with code=0 in 4.682 seconds`

- 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 {
```

ASSIGNMENT NO.2

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



- 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

```

J Bool.java J Strtoboo.java J Strtoboo1.java
1 public class Strtoboo {
2     public static void main(String[] args) {
3         String strStatus = "1";
4         boolean bo = Boolean.parseBoolean(strStatus);
5         System.out.println(bo);
6     }
7 }
8

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Shreya\OneDrive\Desktop\CDAC\OOP> & "C:\Program Files\Java\jdk-17\bin\java.exe" "-XX:+ShowCodeDetailsInExceptionMessages" "-cp" "C:\Users\Shreya\AppData\Roaming\Code\User\workspaceStorage\356a81e220937db061c66ef8ba897f\redhat.java\jdt_ws\OOP_a5ef7248\bin" 'Strtoboo'
false
PS C:\Users\Shreya\OneDrive\Desktop\CDAC\OOP>

```

+ v ... ^ x

- Run: Bool
- Run: Strtoboo
- powershell
- Run: Strtoboo...

- 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 class Strtoboo2{

    public static void main(String[] args) {

        String status ="true";

        boolean bo =Boolean.valueOf(status);

        System.out.println(bo);

    }

}

```

```

J Bool.java J Strtoboo.java J Strtoboo1.java J Strtoboo2.java
1 public class Strtoboo2 {
2     public static void main(String[] args) {
3         String status = "true";
4         boolean bo = Boolean.valueOf(status);
5         System.out.println(bo);
6     }
7 }
8

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Shreya\OneDrive\Desktop\CDAC\OOP> & "C:\Program Files\Java\jdk-17\bin\java.exe" "-XX:+ShowCodeDetailsInExceptionMessages" "-cp" "C:\Users\Shreya\AppData\Roaming\Code\User\workspaceStorage\356a81e220937db061c66ef8ba897f\redhat.java\jdt_ws\OOP_a5ef7248\bin" 'Strtoboo2'
true
PS C:\Users\Shreya\OneDrive\Desktop\CDAC\OOP>

```

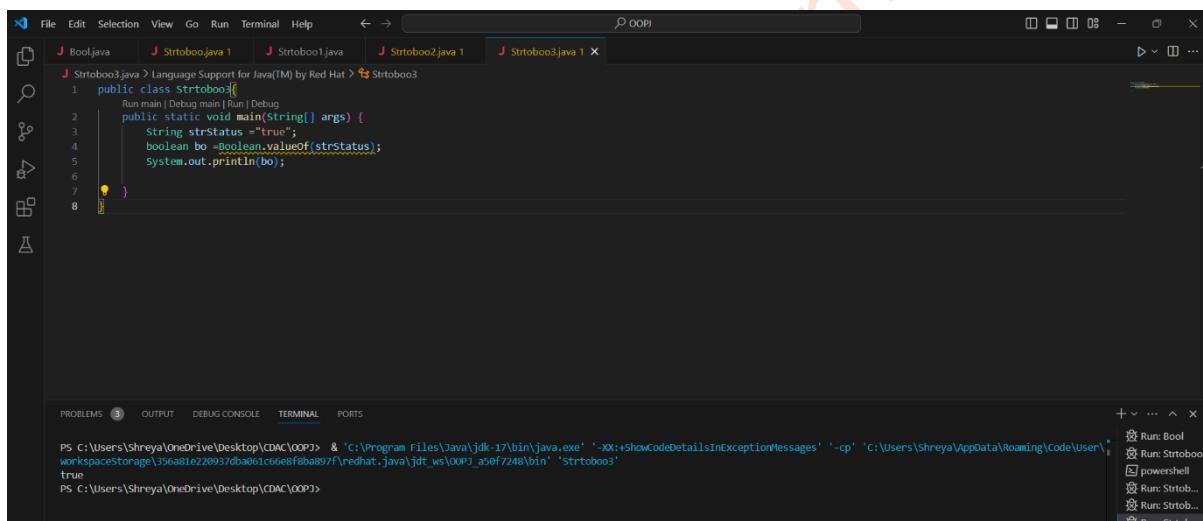
+ v ... ^ x

- Run: Bool
- Run: Strtoboo
- powershell
- Run: Strtoboo...

ASSIGNMENT NO.2

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);  
  
    }  
  
}
```



The screenshot shows an IDE interface with several tabs at the top: File, Edi, Selection, View, Go, Run, Terminal, Help. Below the tabs, there are five tabs for different Java files: Bool.java, Strtoboo.java 1, Strtoboo1.java, Strtoboo2.java 1, and Strtoboo3.java. The Strtoboo3.java tab is active, displaying the provided Java code. On the left, there's a sidebar with icons for file operations like Open, Save, and Find. The main workspace is a large text editor area. At the bottom, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing command-line output:

```
PS C:\Users\Shreya\OneDrive\Desktop\DAC\OOP3> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '--XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Shreya\AppData\Roaming\Code\User\workspaceStorage\356a81e220937dbad61c6ee8faba97f\redhat\java\jdt_ws\OOP_250f7248\bin' 'strtoboo'  
true  
PS C:\Users\Shreya\OneDrive\Desktop\DAC\OOP3>
```

To the right of the terminal, there's a sidebar with several run configurations:

- Run: Bool
- Run: Strtoboo
- powershell
- Run: Strtob...
- Run: Strtob...
- Run: Strtob...

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

ASSIGNMENT NO.2

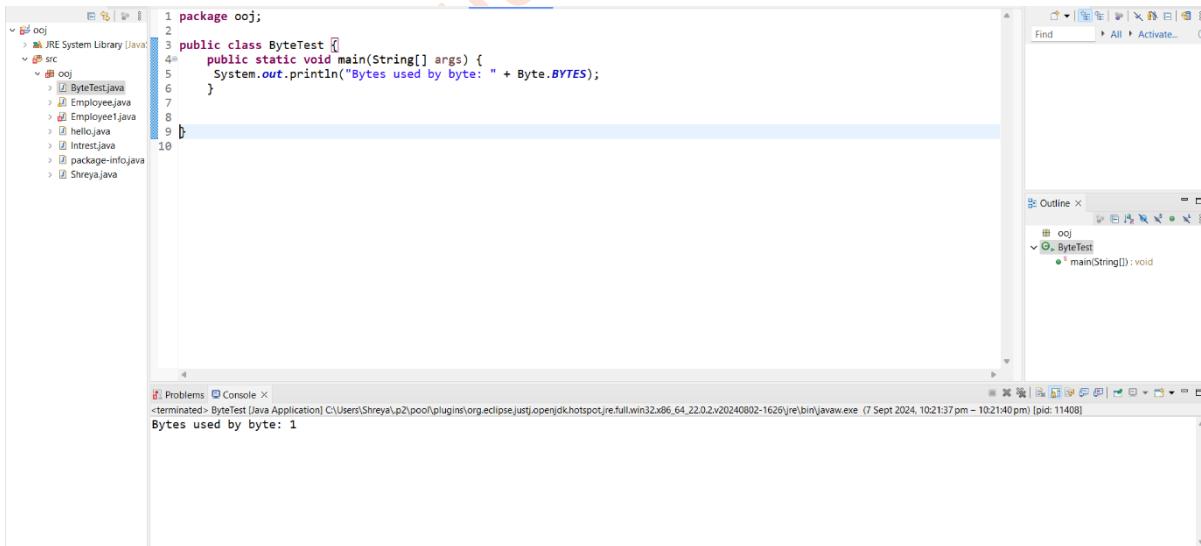
2. Working with `java.lang.Byte`

- 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	BYTES The number of bytes used to represent a <code>byte</code> value in two's complement binary form.
static byte	MAX_VALUE A constant holding the maximum value a <code>byte</code> can have, $2^7 - 1$.
static byte	MIN_VALUE A constant holding the minimum value a <code>byte</code> can have, -2^7 .
static int	SIZE The number of bits used to represent a <code>byte</code> value in two's complement binary form.
static Class<Byte>	TYPE 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 ooj;
public class ByteTest {
    public static void main(String[] args) {
        System.out.println("Bytes used by byte: " + Byte.BYTES);
    }
}
```



ASSIGNMENT NO.2

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);
    }
}
```



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

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

- Project Explorer:** Shows a Java project named "ByteTest" with files like ByteTest.java, Employee.java, Employee1.java, hello.java, Intrest.java, package-info.java, and Shreya.java.
- Code Editor:** Displays the content of ByteTest.java:

```
3  public class ByteTest {  
4      public static void main(String[] args) {  
5          /*  
6             System.out.println("Bytes used by byte: " + Byte.BYTES);  
7         }  
8     */  
9     /*  
10        System.out.println("Min Byte Value: " + Byte.MIN_VALUE);  
11        System.out.println("Max Byte Value: " + Byte.MAX_VALUE);  
12    }  
13 }  
14 */  
15  
16 byte number = 10;  
17 String byteAsString = Byte.toString(number);  
18 System.out.println("Byte as String: " + byteAsString);  
19  
20 }  
21 }  
22 }
```
- Console:** Shows the output: "Byte as String: 10".
- Outline View:** Shows the class structure: ooj, ByteTest, and main(String[]);void.
- Status Bar:** Shows the date: 07 September 2024.

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);  
    }  
}
```

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

```

7 }
8 */
9 /*
10  * System.out.println("Min Byte Value: " + Byte.MIN_VALUE);
11  * System.out.println("Max Byte Value: " + Byte.MAX_VALUE);
12  */
13 }
14 */
15 */
16 */
17 */
18 byte number = 10;
19 String byteAsString = Byte.toString(number);
20 System.out.println("Byte as String: " + byteAsString);
21 */
22 */
23 */
24 */
25 */
26 String strNumber = "Ab12Cd3";
27 byte byteValue = Byte.parseByte(strNumber); // This will throw a NumberFormatException
28 System.out.println("Byte value: " + byteValue); // This line will not be reached
29 */
30 */
31 */

```

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 ooj.ByteTest.main(ByteTest.java:27)

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);
    }
}

```

```

15 */
16 */
17 */
18 byte number = 10;
19 String byteAsString = Byte.toString(number);
20 System.out.println("Byte as String: " + byteAsString);
21 */
22 */
23 */
24 */
25 */
26 String strNumber = "Ab12Cd3";
27 byte byteValue = Byte.parseByte(strNumber);
28 System.out.println("Byte value: " + byteValue);
29 */
30 */
31 */
32 */
33 */
34 byte number = 10;
35 Byte byteWrapper = Byte.valueOf(number);
36 System.out.println("Byte as Wrapper: " + byteWrapper);
37 */
38 */
39 */

```

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) {

```

ASSIGNMENT NO.2

```

String strNumber = "20";
Byte byteWrapper = Byte.valueOf(strNumber);
System.out.println("String to Byte Wrapper: " + byteWrapper);
}

}

23 */
24
25 /*
26     String strNumber = "Ab12Cd3";
27     byte byteValue = Byte.parseByte(strNumber);
28     System.out.println("Byte value: " + byteValue);
29 }
30 */
31 */
32 */
33 */
34     byte number = 10;
35     Byte byteWrapper = Byte.valueOf(number);
36     System.out.println("Byte as Wrapper: " + byteWrapper);
37 }
38 */
39 */
40 String strNumber = "20";
41 Byte byteWrapper = Byte.valueOf(strNumber);
42 System.out.println("String to Byte Wrapper: " + byteWrapper);
43 }
44 */
45 */
46 */
47 */

```

The screenshot shows the Eclipse IDE interface with the ByteTest.java file open in the editor. The code demonstrates how to convert a string to a Byte wrapper and vice versa. The output window at the bottom shows the result of the program's execution.

- 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);
    }
}

```

```

23 */
24
25 /*
26     String strNumber = "Ab12Cd3";
27     byte byteValue = Byte.parseByte(strNumber);
28     System.out.println("Byte value: " + byteValue);
29 }
30 */
31 */
32 */
33 */
34     byte number = 10;
35     Byte byteWrapper = Byte.valueOf(number);
36     System.out.println("Byte as Wrapper: " + byteWrapper);
37 */
38 */
39 */
40 String strNumber = "20";
41 Byte byteWrapper = Byte.valueOf(strNumber);
42 System.out.println("String to Byte Wrapper: " + byteWrapper);
43 */
44 */
45 */
46 */
47 */

```

The screenshot shows the Eclipse IDE interface with the ByteConversion.java file open in the editor. The code performs various primitive type conversions. The output window at the bottom shows the results of the program's execution.

ASSIGNMENT NO.2

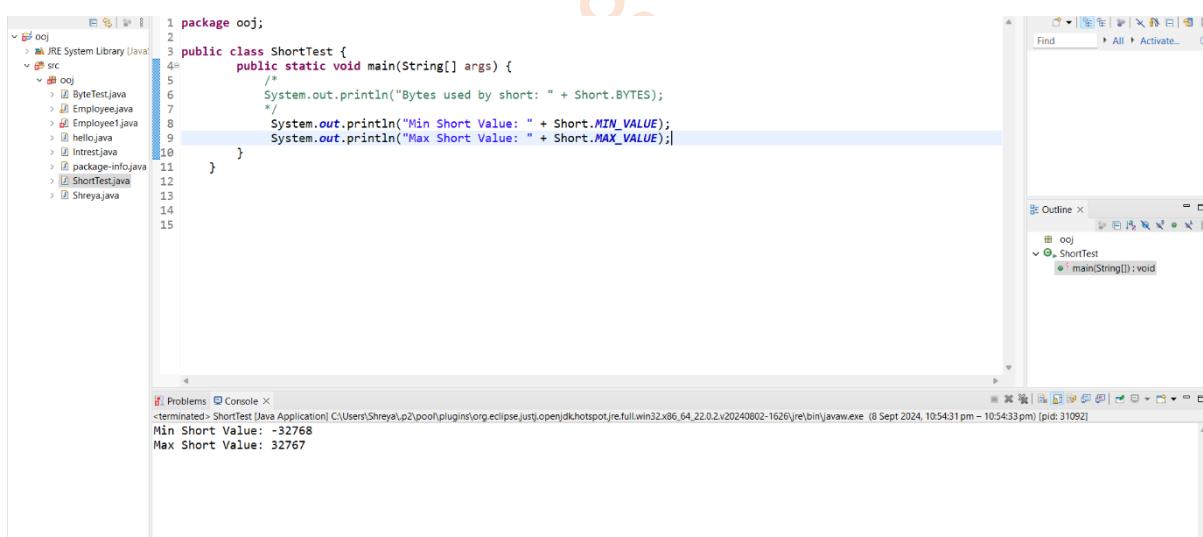
3. Working with `java.lang.Short`

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

- 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);
    }
}
```



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

ASSIGNMENT NO.2

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

- Project Explorer (left):** Shows a project named "ooj" containing several Java files: ByteTest.java, Employee.java, Employee1.java, hello.java, Interest.java, package-info.java, ShortTest.java, and Shreyajava.
- Code Editor (center):** Displays the content of ShortTest.java:

```

1 package ooj;
2
3 public class ShortTest {
4     public static void main(String[] args) {
5         /*
6             System.out.println("Bytes used by short: " + Short.BYTES);
7             System.out.println("Min Short Value: " + Short.MIN_VALUE);
8             System.out.println("Max Short Value: " + Short.MAX_VALUE);
9         }
10    }
11
12
13
14
15

```
- Console (bottom):** Shows the output of the program execution:

```

Min Short Value: -32768
Max Short Value: 32767

```
- Outline View (right):** Shows the outline of the current file, listing the package declaration and the main method.

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)`).

```

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 the Eclipse IDE interface with the following details:

- Project Explorer (left):** Shows the same project structure as the previous screenshot.
- Code Editor (center):** Displays the modified content of ShortTest.java:

```

1 package ooj;
2
3 public class ShortTest {
4     public static void main(String[] args) {
5         /*
6             System.out.println("Bytes used by short: " + Short.BYTES);
7             System.out.println("Min Short Value: " + Short.MIN_VALUE);
8             System.out.println("Max Short Value: " + Short.MAX_VALUE);
9             */
10            short number = 100;
11            String shortAsString = Short.toString(number);
12            System.out.println("Short as String: " + shortAsString);
13        }
14
15
16
17
18
19

```
- Console (bottom):** Shows the output of the program execution:

```

Short as String: 100

```
- Outline View (right):** Shows the outline of the current file, listing the package declaration and the main method.

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

```
}
```

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

- Project Explorer (left):** Shows a project named "ooj" containing packages "src" and "ooj", and files "ByteTest.java", "Employee.java", "Employee1.java", "Hello.java", "Interest.java", "package-info.java", "ShortTest.java", and "Shreya.java".
- Code Editor (center):** Displays the content of `ShortTest.java`. The code prints various short values and demonstrates the conversion from a string to a short value.
- Console (bottom):** Shows the output of the program execution. It includes the command run, the output of `System.out.println("String as String: " + shortAsString);` (String as String: 100), and the output of `System.out.println("String to Short: " + shortValue);` (String to Short: 200).
- Outline (right):** Shows the class structure with `ShortTest` and its method `main(String[] args)`.

- 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

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

- Project Explorer:** Shows a project named "ooj" containing several Java files: ByteTest.java, Employee.java, Employee1.java, hello.java, Intrest.java, package-info.java, ShortTest.java, and Shreya.java.
- Code Editor:** Displays a Java code snippet. The code attempts to parse a string "Ab12Cd3" into a short value, which results in a NumberFormatException. It also prints the string "String to Short: " + shortValue;.
- Console:** Shows the output of the execution, including the exception stack trace:

```

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 ooj.ShortTest.main(ShortTest.java:25)

```

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

```

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

- Project Explorer:** Shows a project named "ooj" containing several Java files: ByteTest.java, Employee.java, Employee1.java, hello.java, Intrest.java, package-info.java, ShortTest.java, and Shreya.java.
- Code Editor:** Displays a Java code snippet. It uses `Short.valueOf(number)` to wrap the short value 100 into a `Short` object. It also prints the string "String to Short: " + shortValue;.
- Console:** Shows the output of the execution, including the printed value:

```

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)).

The screenshot shows the Eclipse IDE interface. On the left is the Project Explorer with a tree view of Java files under 'ooj'. The main area is the code editor with the following content:

```
String strNumber = "200";
short shortValue = Short.parseShort(strNumber);
System.out.println("String to Short: " + shortValue);

/*
String strNumber = "Ab12Cd3";
short shortValue = Short.parseShort(strNumber); // This will throw a NumberFormatException
System.out.println("Short value: " + shortValue); // This line will not be executed

/*
short number = 100;
Short wrappedNumber = Short.valueOf(number);
System.out.println("Wrapped short value: " + wrappedNumber);

*/
String strNumber = "150";
Short wrappedNumber = Short.valueOf(strNumber);
System.out.println("Wrapped short value from string: " + wrappedNumber);
```

The 'Console' tab is selected, showing the output:

```
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

The screenshot shows the Eclipse IDE interface with a Java code editor containing a file named `ShortTest.java`. The code performs various conversions between primitive types. It includes imports for `java.util.Scanner`, `java.lang.*`, and `java.io.*`. The code uses `System.out.println` statements to output the results of these conversions. The output window shows the results of running the program, which include:

```

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

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

ASSIGNMENT NO.2

The screenshot shows the Eclipse IDE interface. On the left is the Project Explorer with a package named 'ooj' containing several Java files like ByteTest.java, Employee.java, etc. The main editor window displays the following Java code:

```

1 package ooj;
2
3 public class IntTest {
4     public static void main(String[] args) {
5         System.out.println("Bytes used by int: " + Integer.BYTES);
6     }
7 }
8
9

```

The Console tab at the bottom shows the output of the program:

```

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);
    }
}

```

The screenshot shows the Eclipse IDE interface. On the left is the Project Explorer with a package named 'ooj' containing several Java files like ByteTest.java, Employee.java, etc. The main editor window displays the following Java code:

```

1 package ooj;
2
3 public class IntTest {
4     public static void main(String[] args) {
5         // System.out.println("Bytes used by int: " + Integer.BYTES);
6
7         System.out.println("Min Int Value: " + Integer.MIN_VALUE);
8         System.out.println("Max Int Value: " + Integer.MAX_VALUE);
9     }
10
11 }
12

```

The Console tab at the bottom shows the output of the program:

```

Min Int Value: -2147483648
Max 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;
    }
}

```

ASSIGNMENT NO.2

```

String strNumber = Integer.toString(number);
System.out.println("String representation: " + strNumber);
}
}
}

```

The screenshot shows the Eclipse IDE interface with the Java code for assignment e. The code is as follows:

```

1 package ooj;
2
3 public class IntTest {
4     public static void main(String[] args) {
5         // System.out.println("Bytes used by int: " + Integer.BYTES);
6
7         /*
8             System.out.println("Min Int Value: " + Integer.MIN_VALUE);
9             System.out.println("Max Int Value: " + Integer.MAX_VALUE);
10        */
11        int number = 12345;
12        String strNumber = Integer.toString(number);
13        System.out.println("String representation: " + strNumber);
14    }
15
16 }
17
18

```

The code prints the string representation of the integer value 12345. The output in the Console view is:

```

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) {
        String strNumber = "54321";
        int number = Integer.parseInt(strNumber);
        System.out.println("Integer value: " + number);
    }
}

```

The screenshot shows the Eclipse IDE interface with the Java code for assignment f. The code is as follows:

```

1 package ooj;
2
3 public class IntTest {
4     public static void main(String[] args) {
5         // System.out.println("Bytes used by int: " + Integer.BYTES);
6
7         /*
8             System.out.println("Min Int Value: " + Integer.MIN_VALUE);
9             System.out.println("Max Int Value: " + Integer.MAX_VALUE);
10        */
11
12        /*
13            int number = 12345;
14            String strNumber = Integer.toString(number);
15            System.out.println("String representation: " + strNumber);
16        */
17
18        String strNumber = "54321";
19        int number = Integer.parseInt(strNumber);
20        System.out.println("Integer value: " + number);
21
22    }
23
24 }
25

```

The code attempts to convert the string "54321" to an integer using `Integer.parseInt(strNumber)`. The output in the Console view is:

```

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`).

ASSIGNMENT NO.2

```
public class IntTest {
    public static void main(String[] args) {
        String strNumber = "Ab12Cd3";
        int number = Integer.parseInt(strNumber); // This will throw
        NumberFormatException
        System.out.println(number);
    }
}
```

The screenshot shows the Eclipse IDE interface. On the left is the Project Explorer view, which lists several Java files under the 'obj' folder. The 'IntTest.java' file is currently selected. The code editor window displays the provided Java code. Below the code editor is the 'Console' tab, which shows the following error message:

```
<terminated> IntTest [Java Application] C:\Users\Shreya\p2\pool\plugins\org.eclipse.jdt\openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\bin\javaw.exe (9 Sept 2024, 12:34:46 pm - 12:34:49 pm) [pid: 33696]
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 ooj.IntTest.main(IntTest.java:26)
```

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

The screenshot shows the Eclipse IDE interface. The 'IntTest.java' file is selected in the Project Explorer. The code editor contains the provided Java code. The 'Console' tab shows the output:

```
<terminated> IntTest [Java Application] C:\Users\Shreya\p2\pool\plugins\org.eclipse.jdt\openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\bin\javaw.exe (9 Sept 2024, 12:37:09 pm - 12:37:11 pm) [pid: 30060]
Wrapper class value: 100
```

ASSIGNMENT NO.2

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);  
    }  
}
```



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);  
    }  
}
```

ASSIGNMENT NO.2

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

- Project Explorer (left):** Shows a project named "ooj" containing several Java files: ByteTest.java, Employee.java, Employee1.java, hello.java, IntTest.java, package-info.java, ShortTest.java, and Shreya.java.
- Code Editor (center):** Displays the following Java code in the `IntTest.java` file:

```

    ...
    System.out.println("Wrapper class value: " + wrapper);
    ...
    String strNumber = "200";
    Integer wrapper = Integer.valueOf(strNumber);
    System.out.println("Wrapper class value: " + wrapper);
    ...

    int num1 = 10;
    int num2 = 20;
    /*
    int sum = Integer.sum(num1, num2);
    System.out.println("Sum: " + sum);
    */
    System.out.println("Minimum: " + Integer.min(num1, num2));
    System.out.println("Maximum: " + Integer.max(num1, num2));
}
}

```

- Console (bottom-left):** Shows the output of the program: "Minimum: 10" and "Maximum: 20".
- Outline View (right):** Shows the outline of the current file, including the class definition and the main method.

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));
    }
}

```

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

- Project Explorer (left):** Shows a project named "ooj" containing several Java files: ByteTest.java, Employee.java, Employee1.java, hello.java, IntTest.java, package-info.java, ShortTest.java, and Shreya.java.
- Code Editor (center):** Displays the same Java code as the previous screenshot, located in the `IntTest.java` file.
- Console (bottom-left):** Shows the output of the program: "Minimum: 10" and "Maximum: 20".
- Outline View (right):** Shows the outline of the current file, including the class definition and the main method.

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)`).

```

public class IntTest {
    public static void main(String[] args) {

```

ASSIGNMENT NO.2

```
int number = 7;

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

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

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

}

}
```

The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer displays a Java project named 'ooj' containing several source files: ByteTest.java, Employee.java, Employee1.java, hello.java, Intrest.java, IntTest.java, package-info.java, ShortTest.java, and Shreya.java. The IntTest.java file is open in the editor, showing the provided Java code. The code defines an integer variable 'number' and prints its binary, octal, and hexadecimal representations. On the right, the Outline view shows the class 'IntTest' and its main method. Below the editor, the Java Console window shows the execution results: 'Binary: 111', 'Octal: 7', and 'Hexadecimal: f'. The status bar at the bottom indicates the application is running in a terminal window.

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
        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;
    }
}
```

ASSIGNMENT NO.2

```

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

```

The screenshot shows the Eclipse IDE interface with a Java project named 'ooj'. The code in the editor is as follows:

```

56     // System.out.println("Maximum: " + Integer.max(num1, num2));
57
58     /*
59      * int number = 7;
60      * System.out.println("Binary: " + Integer.toBinaryString(number));
61      * System.out.println("Octal: " + Integer.toOctalString(number));
62      * System.out.println("Hexadecimal: " + Integer.toHexString(number));
63     */
64     int number = 100;
65
66     // Convert int to long
67     long longValue = (long) number;
68     System.out.println("Long value: " + longValue);
69
70     // Convert int to double
71     double doubleValue = (double) number;
72     System.out.println("Double value: " + doubleValue);
73
74     // Convert int to float
75     float floatValue = (float) number;
76     System.out.println("Float value: " + floatValue);
77
78     // Convert int to byte
79     byte byteValue = (byte) number;
80     System.out.println("Byte value: " + byteValue); // May truncate for larger values

```

The output window shows the results of the println statements:

```

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	BYTES The number of bytes used to represent a <code>float</code> value.
static int	MAX_EXPONENT Maximum exponent a finite <code>float</code> variable may have.
static float	MAX_VALUE A constant holding the largest positive finite value of type <code>float</code> , $(2 \cdot 2^{31}) \cdot 2^{127}$.
static int	MIN_EXPONENT Minimum exponent a normalized <code>float</code> variable may have.
static float	MIN_NORMAL A constant holding the smallest positive normal value of type <code>float</code> , 2^{-126} .
static float	MIN_VALUE A constant holding the smallest positive nonzero value of type <code>float</code> , 2^{-149} .
static float	NaN A constant holding a Not-a-Number (NaN) value of type <code>float</code> .
static float	NEGATIVE_INFINITY A constant holding the negative infinity of type <code>float</code> .
static float	POSITIVE_INFINITY A constant holding the positive infinity of type <code>float</code> .
static int	SIZE The number of bits used to represent a <code>float</code> value.
static Class<Float>	TYPE The <code>Class</code> 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)`).

ASSIGNMENT NO.2

- 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 LongOperationsTest {  
    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);
}
}

```

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

- Packaging Explorer:** Shows the project structure with files like hello.java, Employee.java, IntTest.java, Shrey.java, ByteTest.java, ShortTest.java, and LongTest.java.
- Code Editor:** Displays the Java code for converting long values to other primitive types (int, double, float, byte).
- Console Output:** Shows the execution results of the code, including:
 - Bytes used to represent a long: 8
 - Minimum long value: -9223372036854775808
 - Maximum long value: 9223372036854775807
 - String representation of long: 123456789
 - Parsed long value from string: 987654321
 - Exception: For input string: "Ab12Cd3"
 - Wrapper class value from long: 123456789
 - Wrapper class value from string: 987654321
 - 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`

- Explore the [Java API documentation for `java.lang.Float`](#) and observe its modifiers and super types.
- Write a program to test how many bytes are used to represent a `float` value using the `BYTES` field. (Hint: Use `Float.BYTES`).
- 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`).
- 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)`).
- 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)`).
- 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;
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
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 the Eclipse IDE interface with the Java perspective. On the left, the Project Explorer shows a package named 'oobj' containing several source files: ByteTest.java, Employee.java, EmployeeTest.java, Hello.java, IntTest.java, LongTest.java, package-info.java, ShortTest.java, and Shreya.java. The 'FloatTest.java' file is open in the editor, displaying the provided Java code. To the right of the editor is the 'Outline' view, which shows the class 'FloatTest' and its method 'main(String[])'. Below the editor is the 'Console' view, which displays the execution output of the program. The output shows various floating-point operations and conversions:

```

1 package oobj;
2 public class FloatTest {
3     public static void main(String[] args) {
4         // a. Test how many bytes are used to represent a float value
5         System.out.println("Bytes used to represent a float: " + Float.BYTES);
6         // b. Find the minimum and maximum values of float
7         System.out.println("Minimum float value: " + Float.MIN_VALUE);
8         System.out.println("Maximum float value: " + Float.MAX_VALUE);
9         // c. Convert a float value to a String using Float.toString
10        float number = 123.45f;
11        String strNumber = Float.toString(number);
12        System.out.println("String representation of float: " + strNumber);
13
14        // d. Convert a float value to a String using Double.toString
15        double parsedNumber = Double.parseDouble(strNumber);
16        System.out.println("Parsed float value from string: " + parsedNumber);
17
18        // e. Divide two float values
19        float result = 987.65f / 123.45f;
20        System.out.println("Result of dividing 987.65f by 123.45f: " + result);
21
22        // f. Calculate square root of a float value
23        float squareRoot = Math.sqrt(987.65f);
24        System.out.println("Square root of 987.65f: " + squareRoot);
25
26        // g. Convert a float value to a long value
27        long convertedLong = (long) 987.65f;
28        System.out.println("Converted to long: " + convertedLong);
29
30        // h. Convert a float value to a double value
31        double convertedDouble = 987.65f;
32        System.out.println("Converted to double: " + convertedDouble);
33
34        // i. Convert a float value to a byte value
35        byte convertedByte = (byte) 987.65f;
36        System.out.println("Converted to byte: " + convertedByte);
37
38    }
39 }

```

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`).
- 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`).
- 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)`).
- 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)`).

ASSIGNMENT NO.2

- 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 DoubleOperationsTest {  
    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;
```

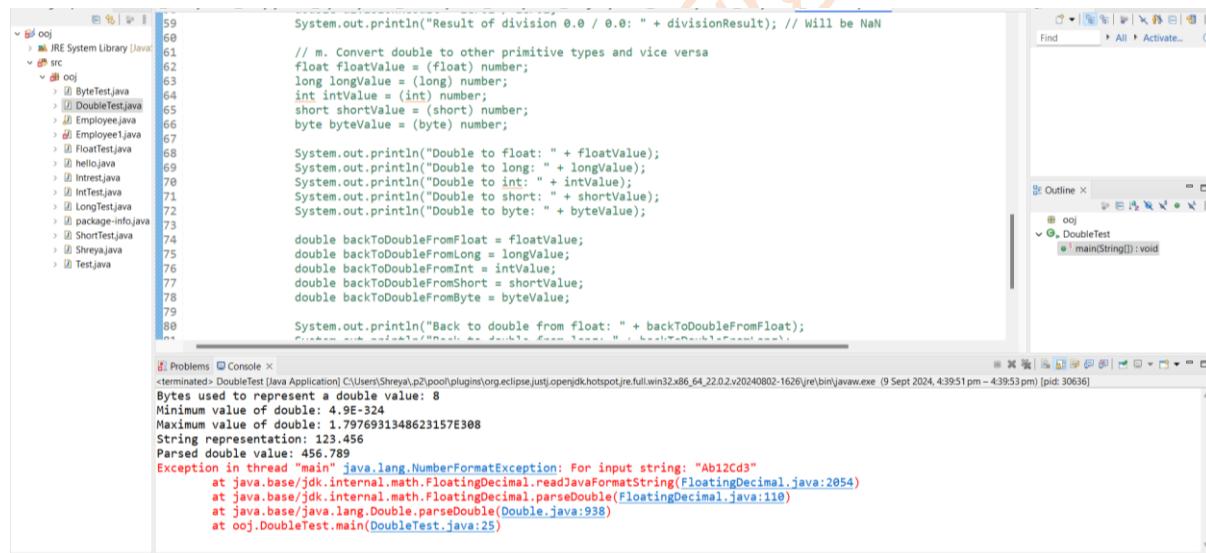
ASSIGNMENT NO.2

```
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);

}
```



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
```

ASSIGNMENT NO.2

```
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));
}

}
```

The screenshot shows an IDE interface with two panes. The left pane displays a Java file named `Conversion.java` containing the assignment code. The right pane shows the terminal window where the code has been run, displaying the output of both the `toString` and `valueOf` methods for all primitive types.

```
// 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));
```

```
Monday - Today
Find All Activate...
Problems Console Terminal
<terminated> Conversion [Java Application] C:\Users\Shreya\p2\pool\pl
Using wrapper class toString 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

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).

```
package oo;

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
        DefaultValuesTest test = new DefaultValuesTest();

        // 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);
}
}

```

```

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
    DefaultValuesTest test = new DefaultValuesTest();

    // 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);
}

```

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).

```

package ooj;
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);
}
```