ASSIGNMENT NO.2

Note: Consider the following before starting the assignment:

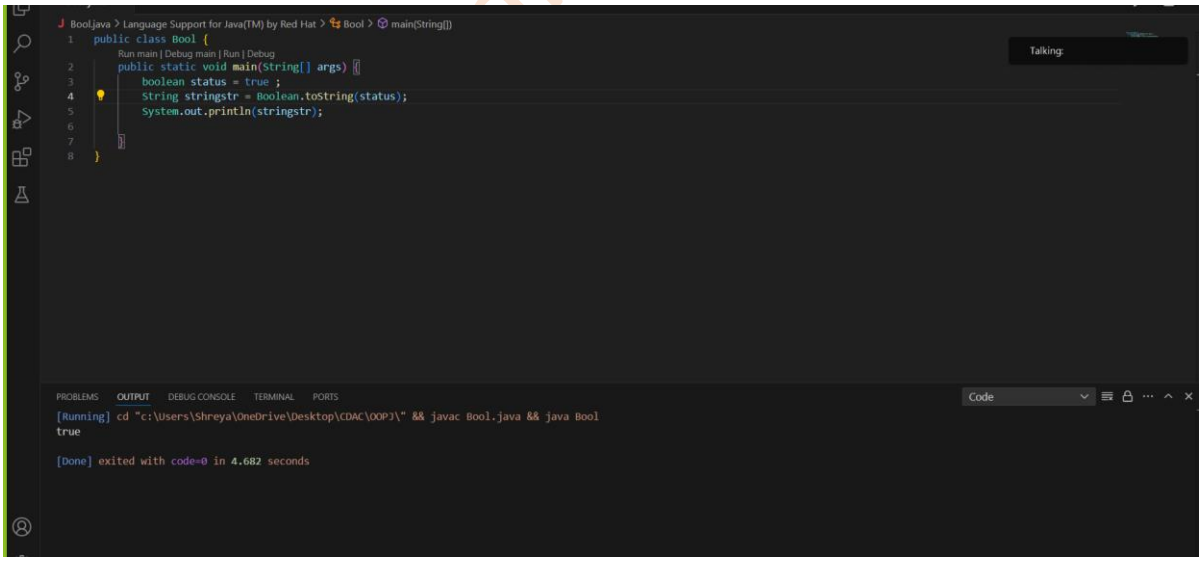
- A **static field** declared inside a class is called a **class-level variable**. To access this variable, use the class name and the dot operator (e.g., `Integer.MAX_VALUE`).
- A **static method** defined inside a class is called a **class-level method**. To access this method, use the class name and 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`

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)`).

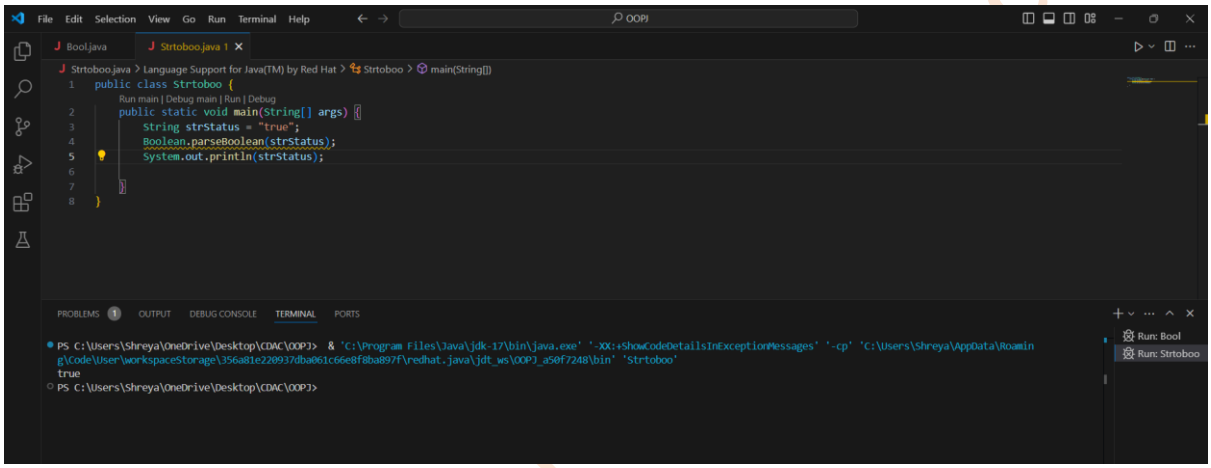
```
public class bool {
   public static void main(String[] args) {
      boolean status = true ;
      String stringstr = Boolean.toString(status);
      System.out.println(stringstr);

   }
}
```



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 {
```

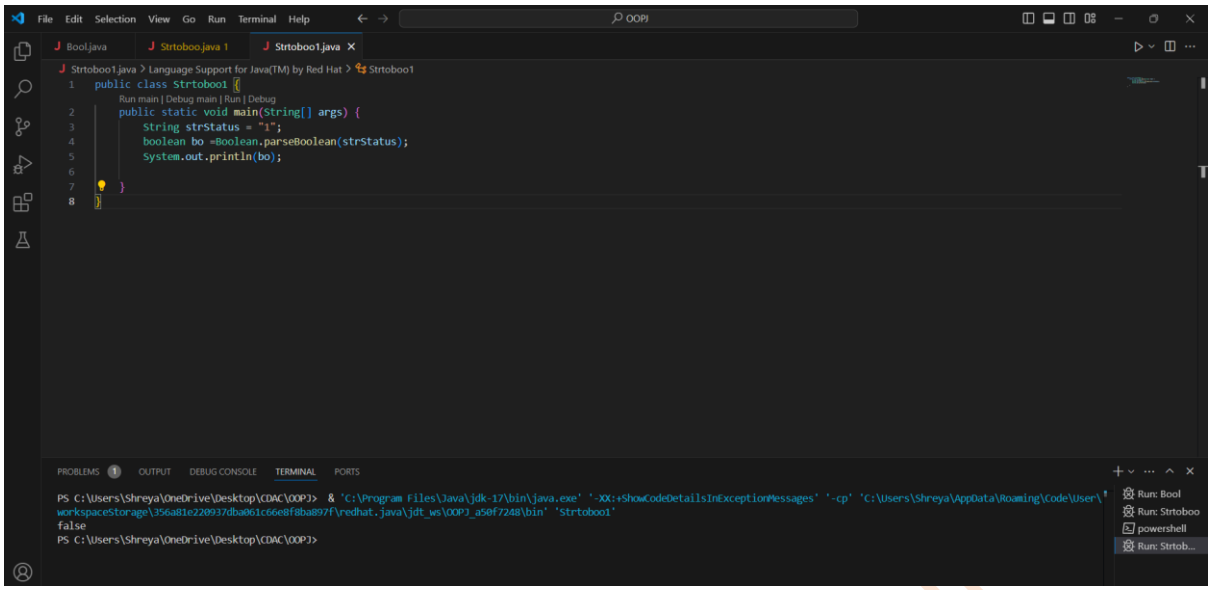
```
    public static void main(String[] args) {
        String strStatus = "true";
        Boolean.parseBoolean(strStatus);
        System.out.println(strStatus);

    }
}
```



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"`).

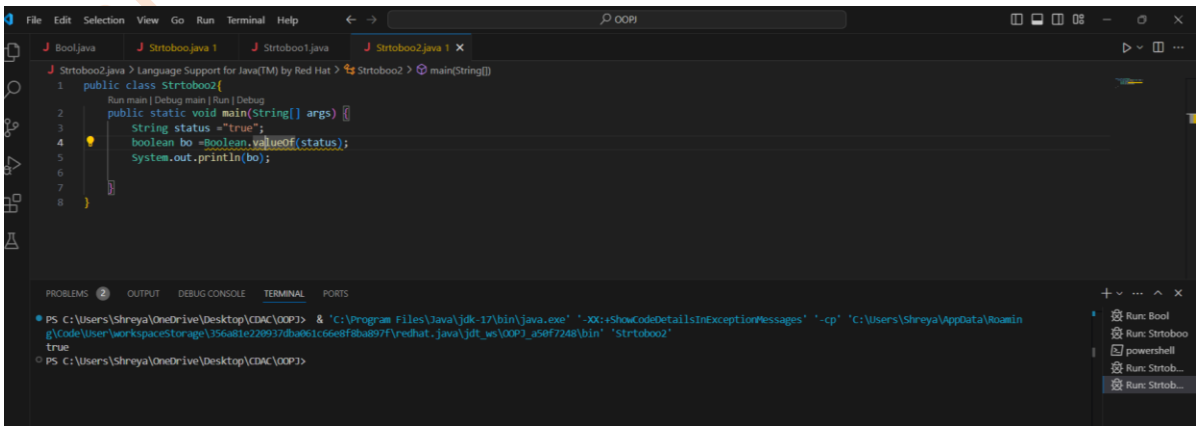
```
public class Strtoboo1 {
    public static void main(String[] args) {
        String strStatus = "1";
        boolean bo =Boolean.parseBoolean(strStatus);
        System.out.println(bo);
    }
}
```

ASSIGNMENT NO.2



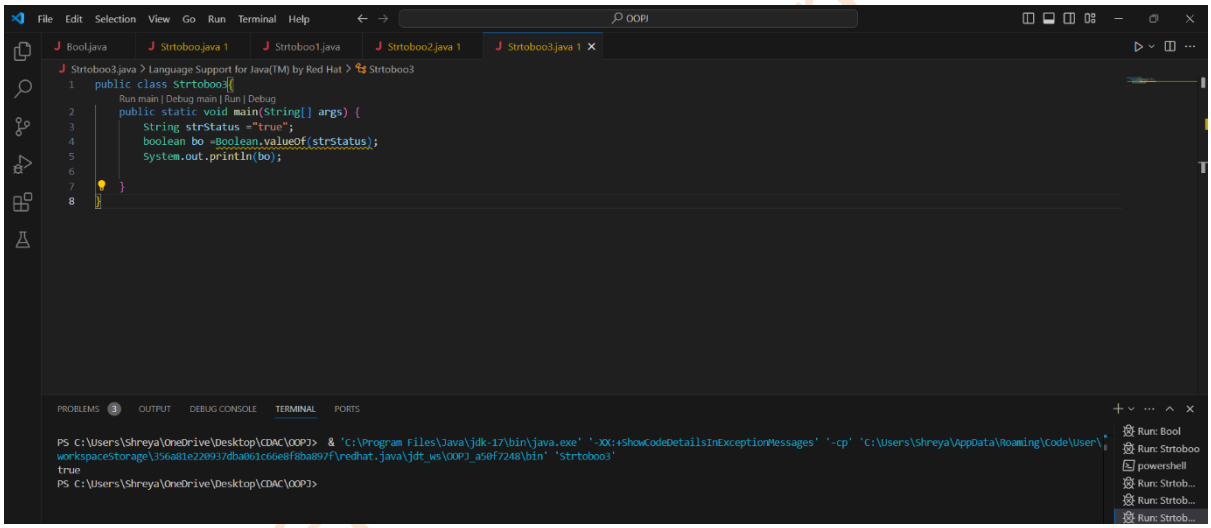
e. Declare a method-local variable `status` of type `boolean` with the value `true` and convert it to the corresponding wrapper class using `Boolean.valueOf()`. (Hint: Use **`Boolean.valueOf(boolean)`**).

```
public class Strtoboo2{

    public static void main(String[] args) {

        String status ="true";

        boolean bo =Boolean.valueOf(status);

        System.out.println(bo);

    }

}
```



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);

   }

}
```



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.

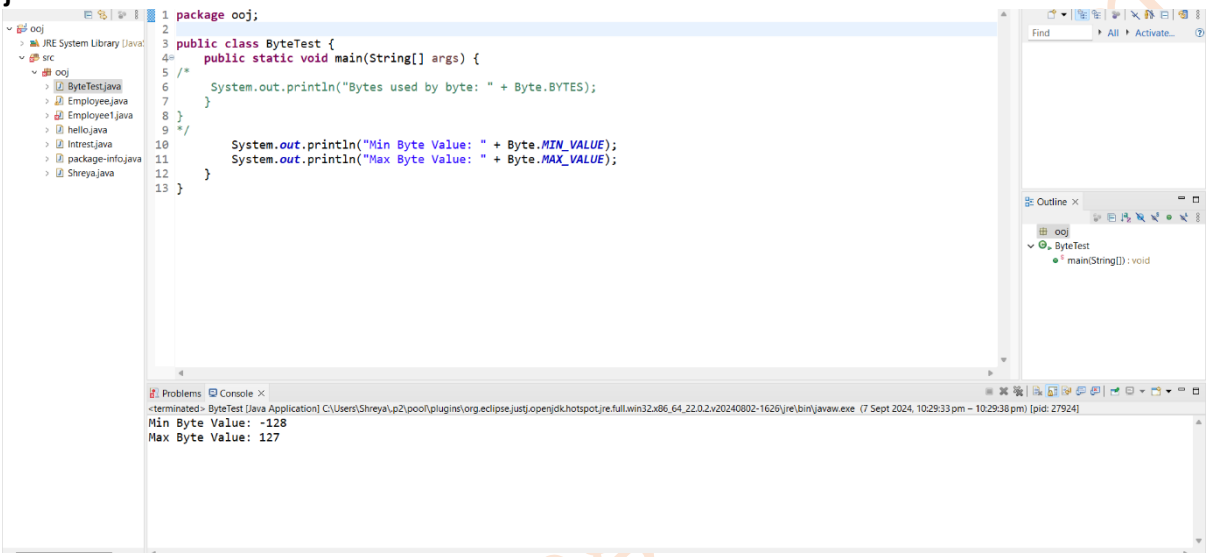
Modifier and Type	Field and Description
`static int`	**BYTES** The number of bytes used to represent a `byte` value in two's complement binary form.
`static byte`	**MAX_VALUE** A constant holding the maximum value a `byte` can have, 2^7-1.
`static byte`	**MIN_VALUE** A constant holding the minimum value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

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

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

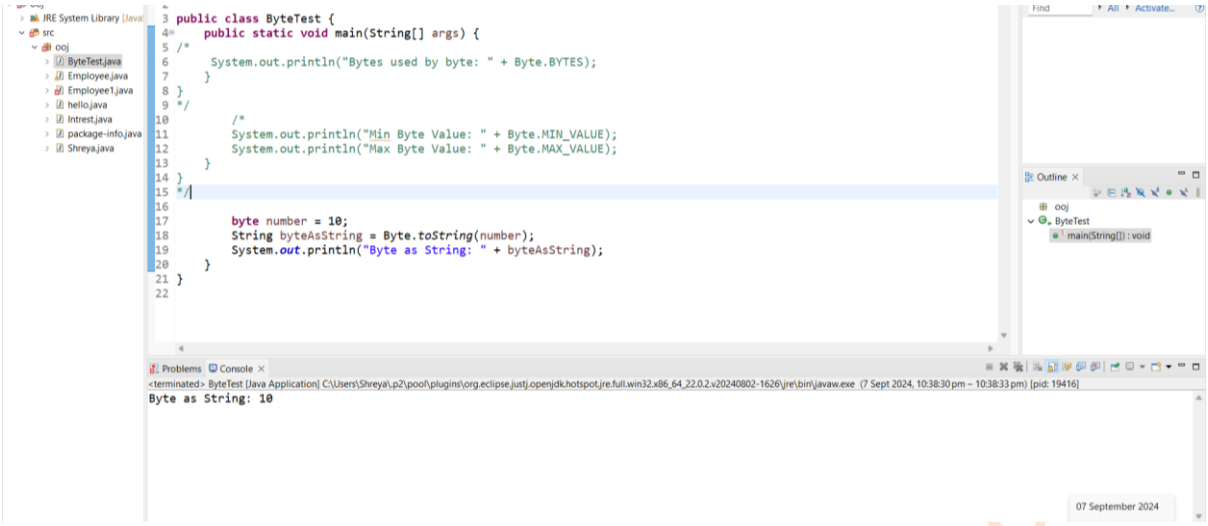
c. Write a program to find the minimum and maximum values of `byte` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Byte.MIN_VALUE` and `Byte.MAX_VALUE`).

```
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);
   }
}
```



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)`).

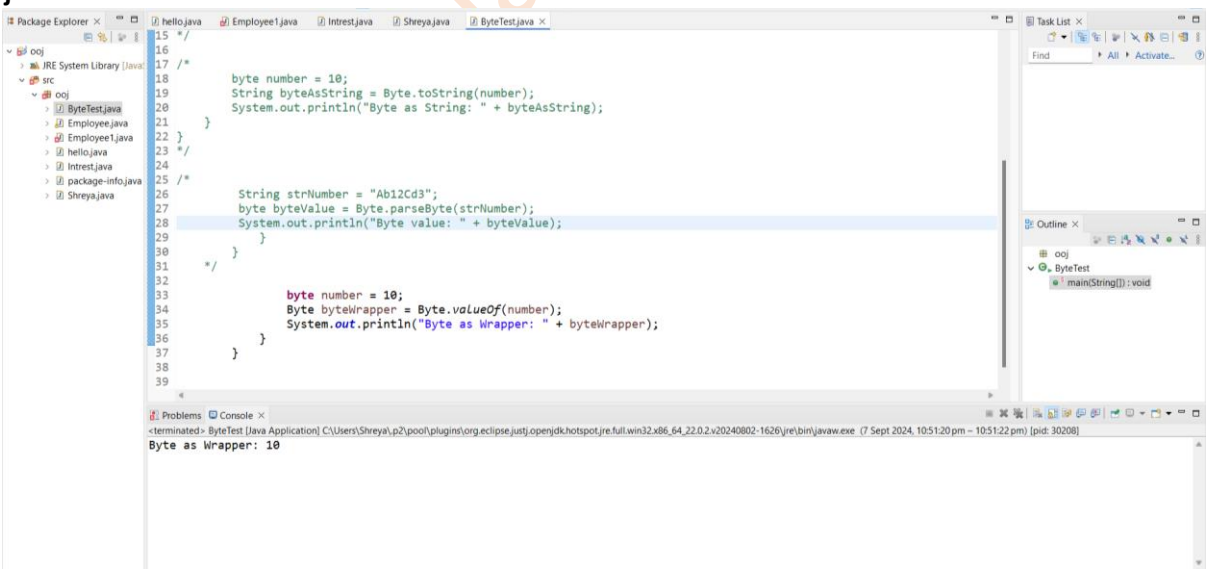
```
public class StringToByte {
    public static void main(String[] args) {
        String strNumber = "12";
        byte byteValue = Byte.parseByte(strNumber);
        System.out.println("String to Byte: " + byteValue);
    }
}
```

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);
    }
}
```

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)`).

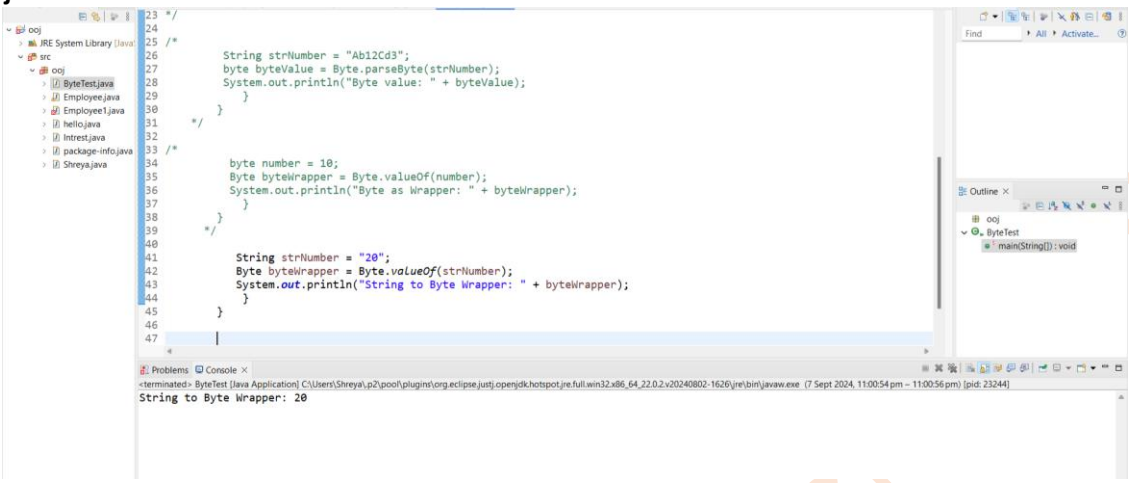
```
public class ByteTest {
   public static void main(String[] args) {
      byte number = 10;
      Byte byteWrapper = Byte.valueOf(number);
      System.out.println("Byte as Wrapper: " + byteWrapper);
   }
}
```



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)`).

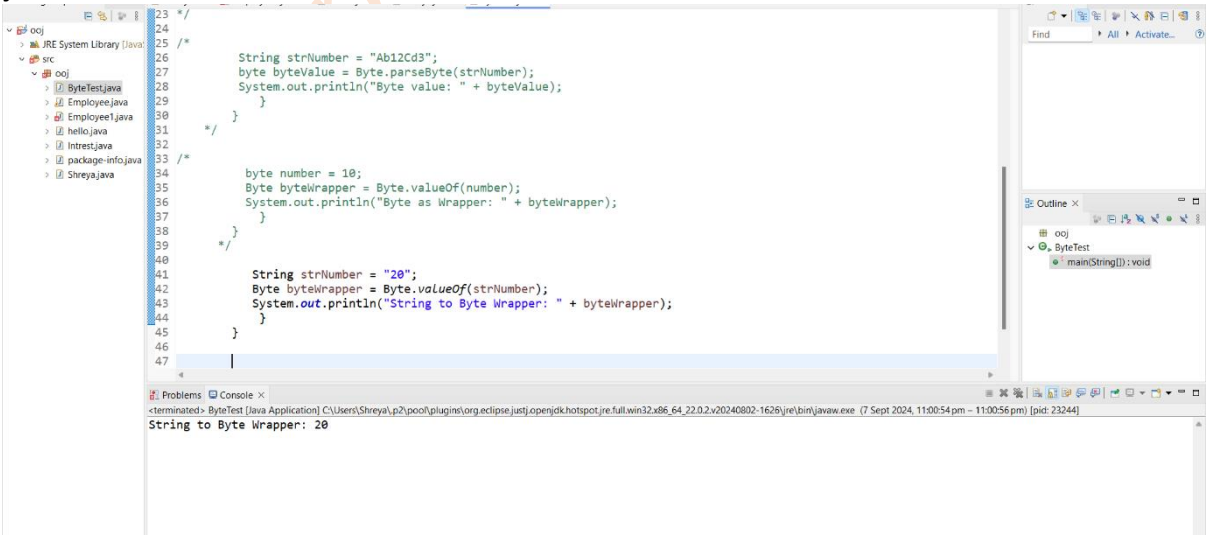
```
public class StringToByteValueOf {
   public static void main(String[] args) {
```

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



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);
    }
}
```



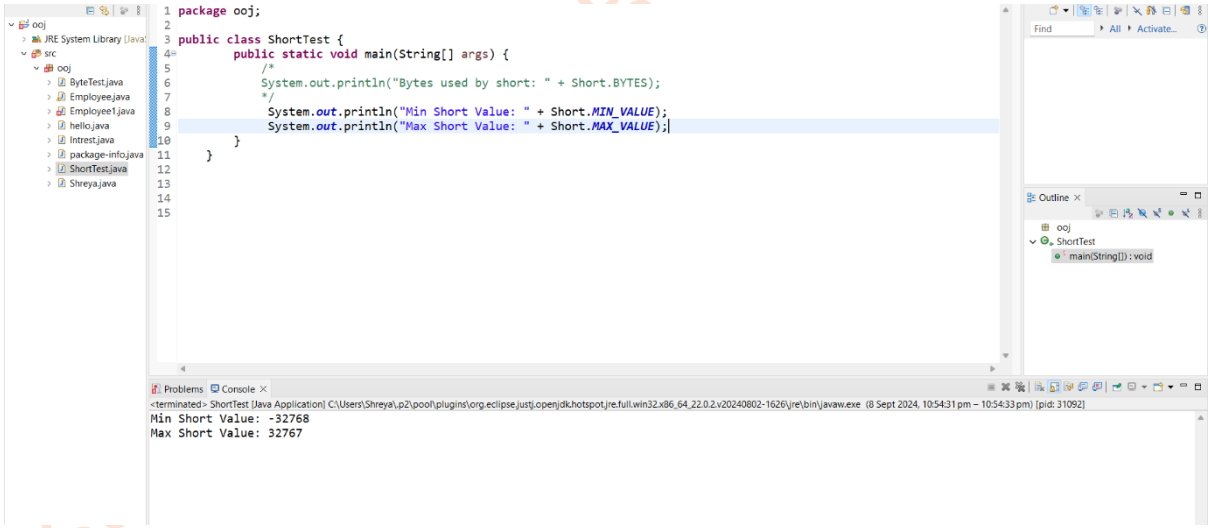
3. Working with `java.lang.Short`

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

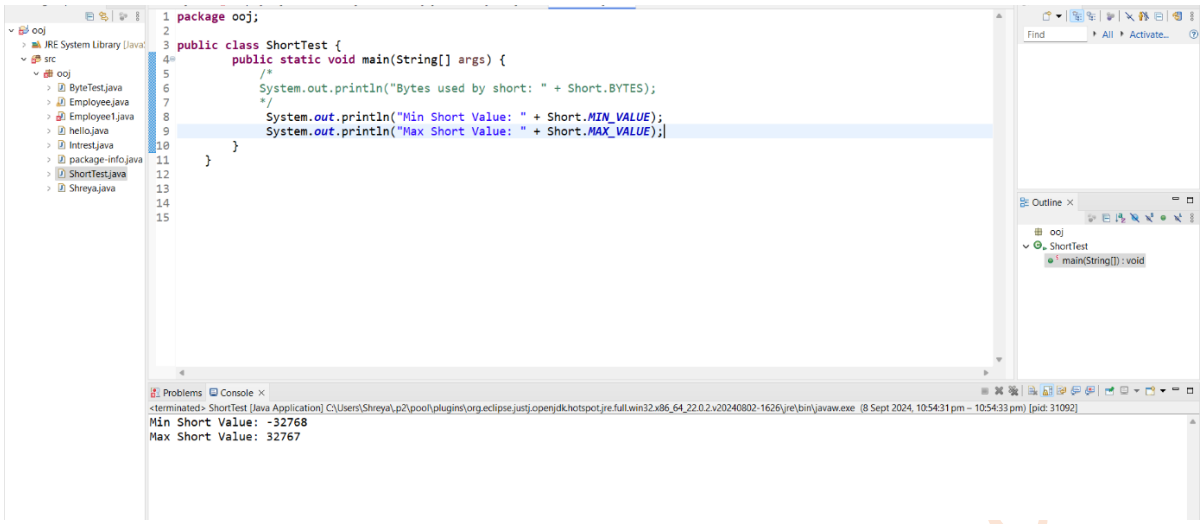
b. Write a program to test how many bytes are used to represent a `short` value using the `BYTES` field. (Hint: Use `Short.BYTES`).

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



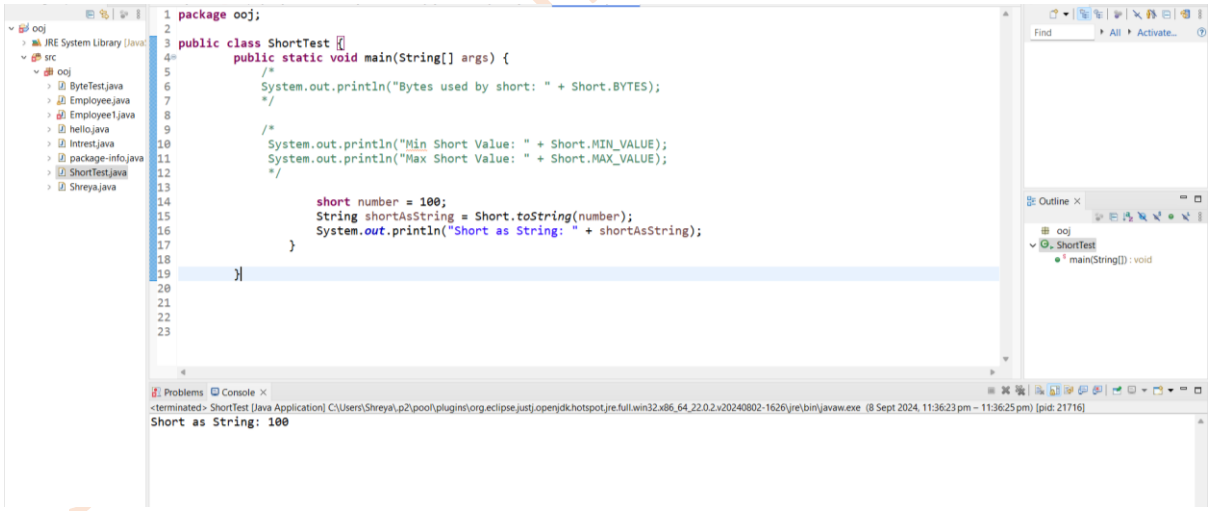
c. Write a program to find the minimum and maximum values of `short` using the `MIN_VALUE` and `MAX_VALUE` fields. (Hint: Use `Short.MIN_VALUE` and `Short.MAX_VALUE`).

```
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);
    }
}
```



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)`).

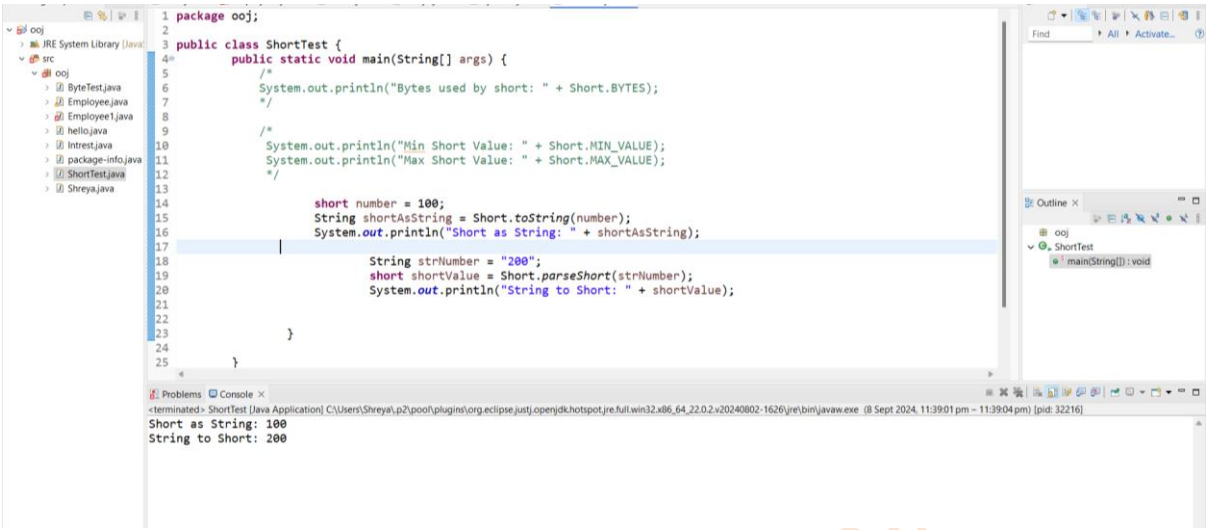
```
public class ShortTest {
   public static void main(String[] args) {
      short number = 100;
      String shortAsString = Short.toString(number);
      System.out.println("Short as String: " + shortAsString);
   }
}
```



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);
```

```
    }
}
```

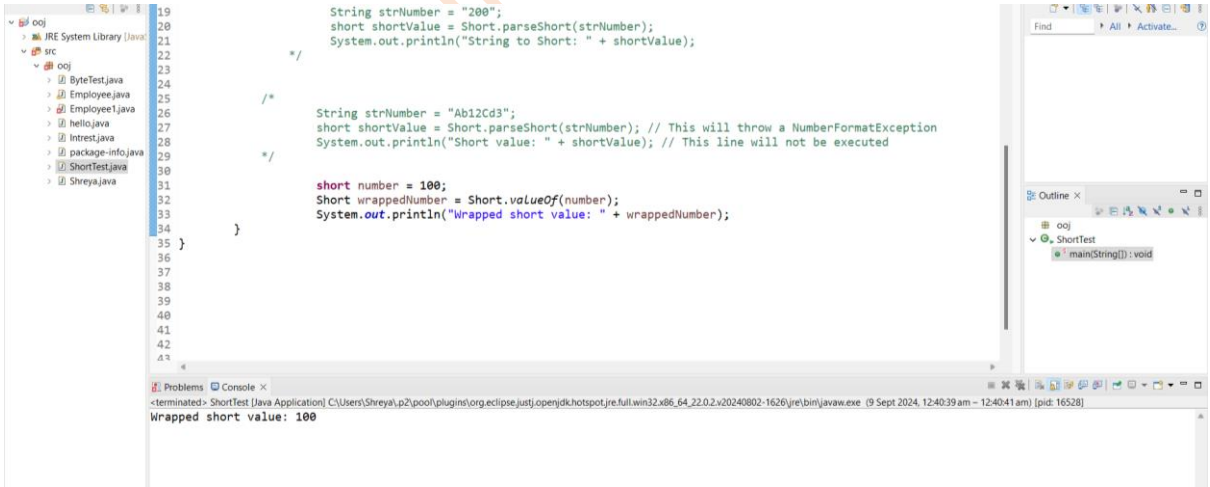


f. Declare a method-local variable `strNumber` of type `String` with the value `"Ab12Cd3"` and attempt to convert it to a `short` value. (Hint: `parseShort` method will throw a `NumberFormatException`).

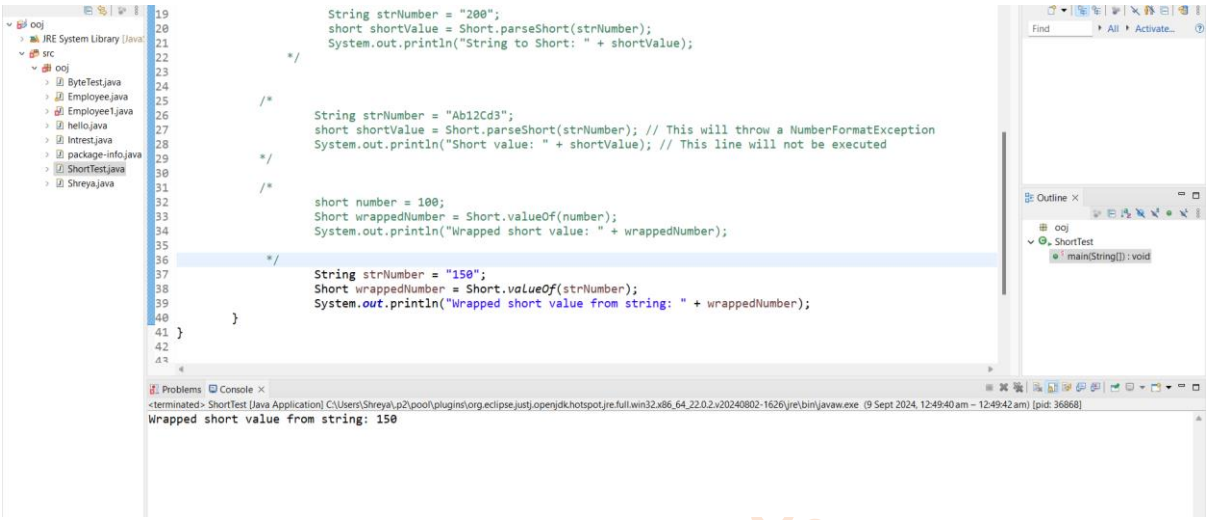
```
public class ShortTest {

    public static void main(String[] args) {

        String strNumber = "Ab12Cd3";

        short shortValue = Short.parseShort(strNumber);

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

    }

}
```

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);
    }
}
```

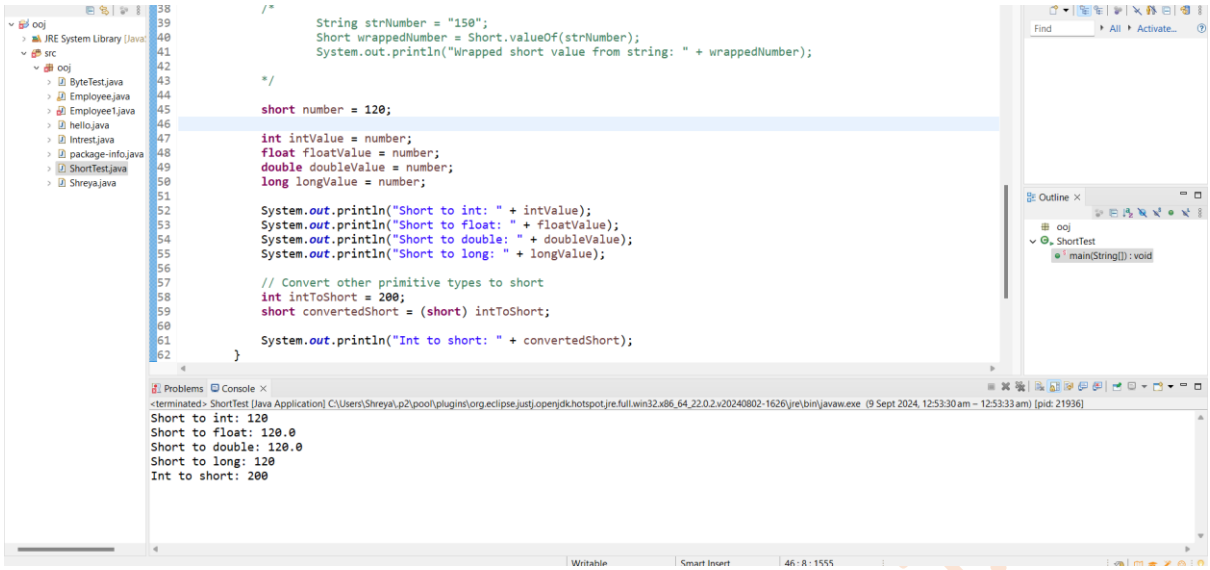


h. Declare a method-local variable `strNumber` of type `String` with some `short` value and convert it to the corresponding wrapper class using `Short.valueOf()`. (Hint: Use `Short.valueOf(String)`).



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

```
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);
   }
}
```



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

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

```
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);
   }

}
```

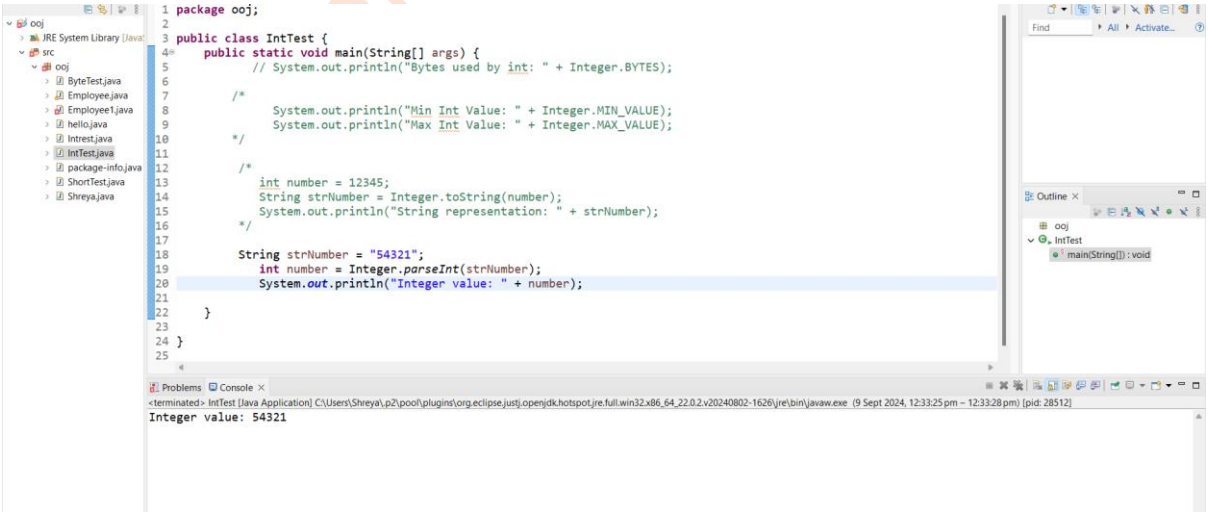
d. Declare a method-local variable `number` of type `int` with some value and convert it to a `String` using the `toString` method. (Hint: Use `Integer.toString(int)`).

```
public class IntTest {
   public static void main(String[] args) {
      int number = 12345;
```

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

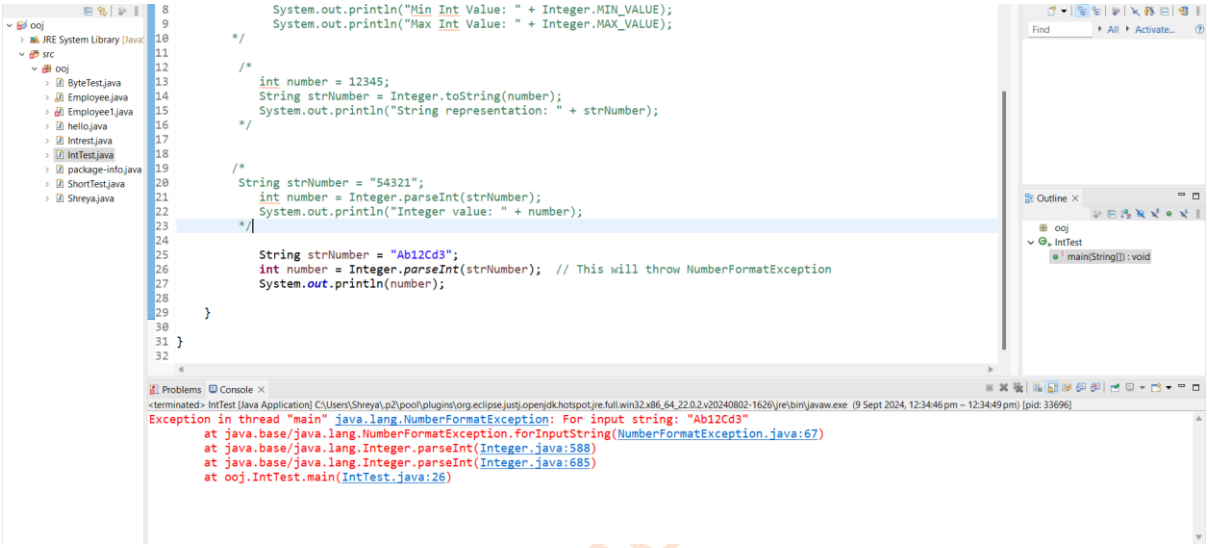
e. Declare a method-local variable `strNumber` of type `String` with some value and convert it to an `int` value using the `parseInt` method. (Hint: Use `Integer.parseInt(String)`).

```
public class IntTest {
    public static void main(String[] args) {
        String strNumber = "54321";
        int number = Integer.parseInt(strNumber);
        System.out.println("Integer value: " + number);
    }
}
```



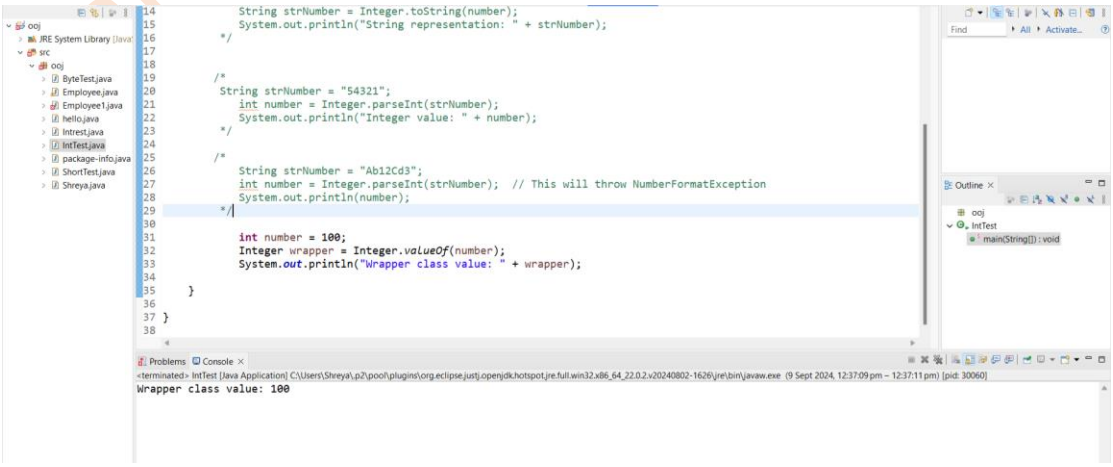
f. Declare a method-local variable `strNumber` of type `String` with the value `"Ab12Cd3"` and attempt to convert it to an `int` value. (Hint: `parseInt` method will throw a `NumberFormatException`).

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



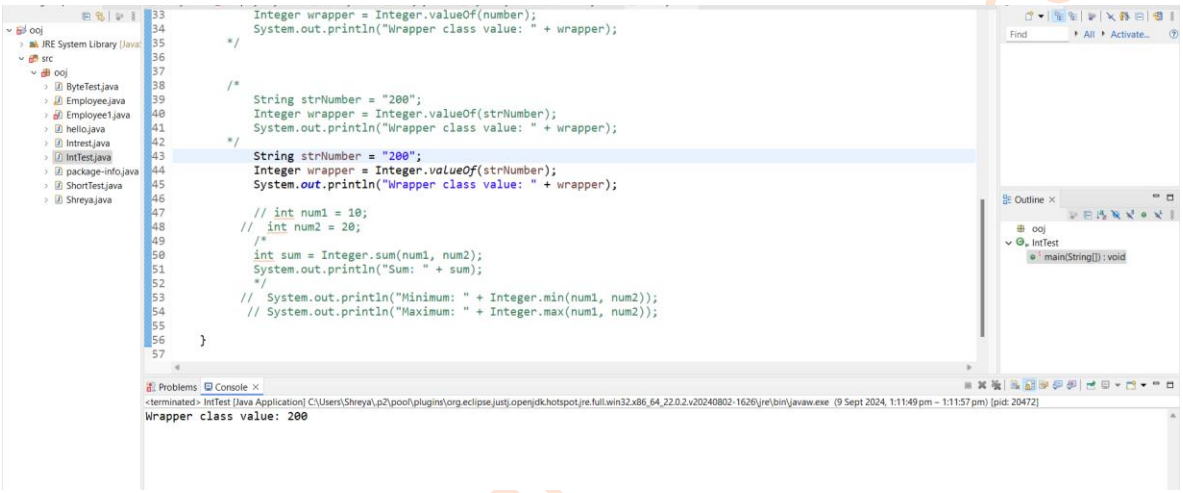
g. Declare a method-local variable `number` of type `int` with some value and convert it to the corresponding wrapper class using `Integer.valueOf()`. (Hint: Use `Integer.valueOf(int)`).

```
public class IntTest {
    public static void main(String[] args) {
        int number = 100;
        Integer wrapper = Integer.valueOf(number);
        System.out.println("Wrapper class value: " + wrapper);
    }
}
```



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);
   }
}
```

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));
   }
}
```

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)`).

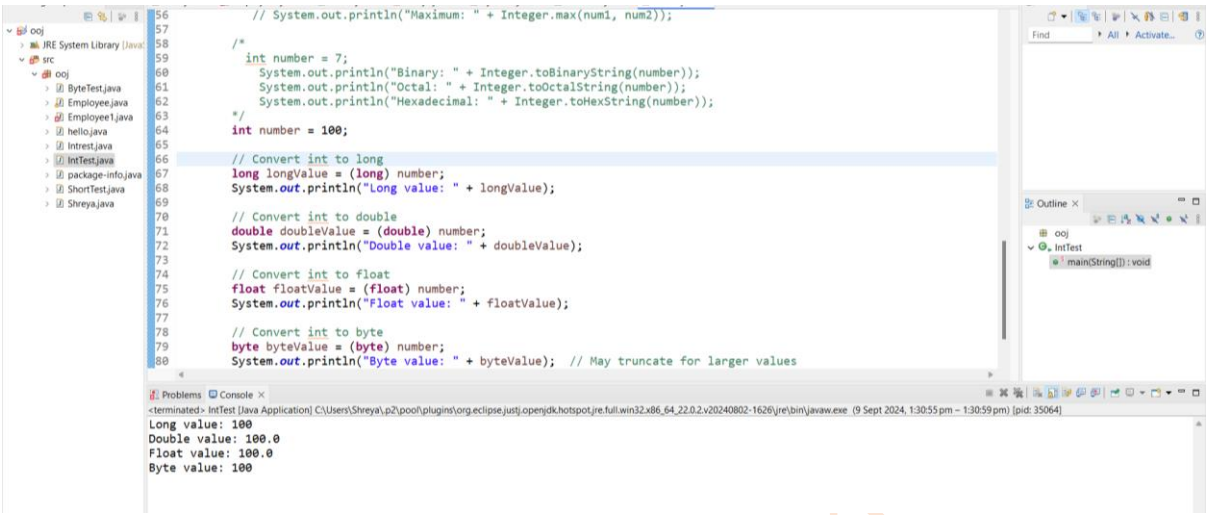
```
public class IntTest {

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

```
        int number = 7;

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

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

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

    }

}
```

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;
```

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



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

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)`).

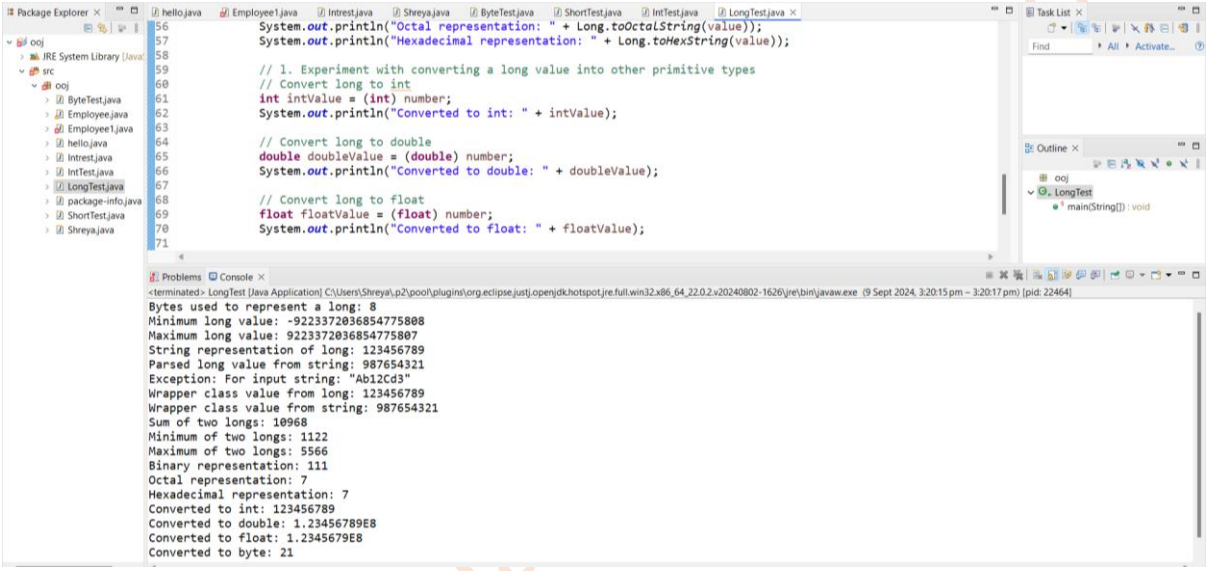
k. Declare a long variable with the value `7`. Convert it to binary, octal, and hexadecimal strings using methods from the `Long` class. (Hint: Use `Long.toBinaryString(long)`, `Long.toOctalString(long)`, and `Long.toHexString(long)`).

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);
    }
}
```



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`).

ASSIGNMENT NO.2

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).

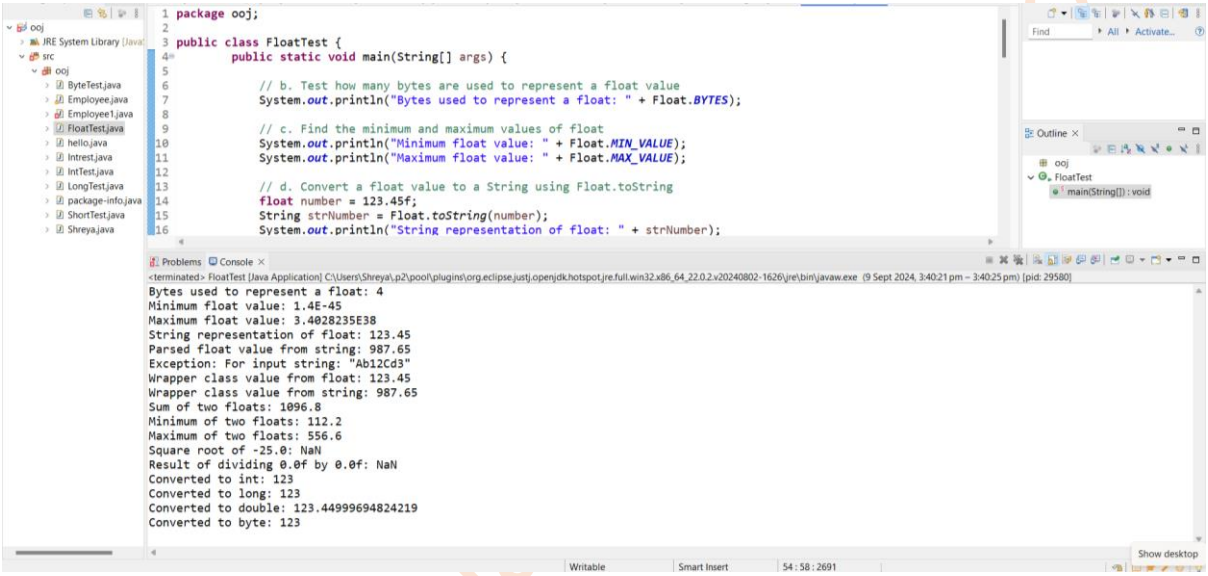
l. Declare two float variables with the same value, `0.0f`, and divide them. (Hint: Observe the result and any special floating-point behavior).

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);
    }
}
```



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)`).

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).

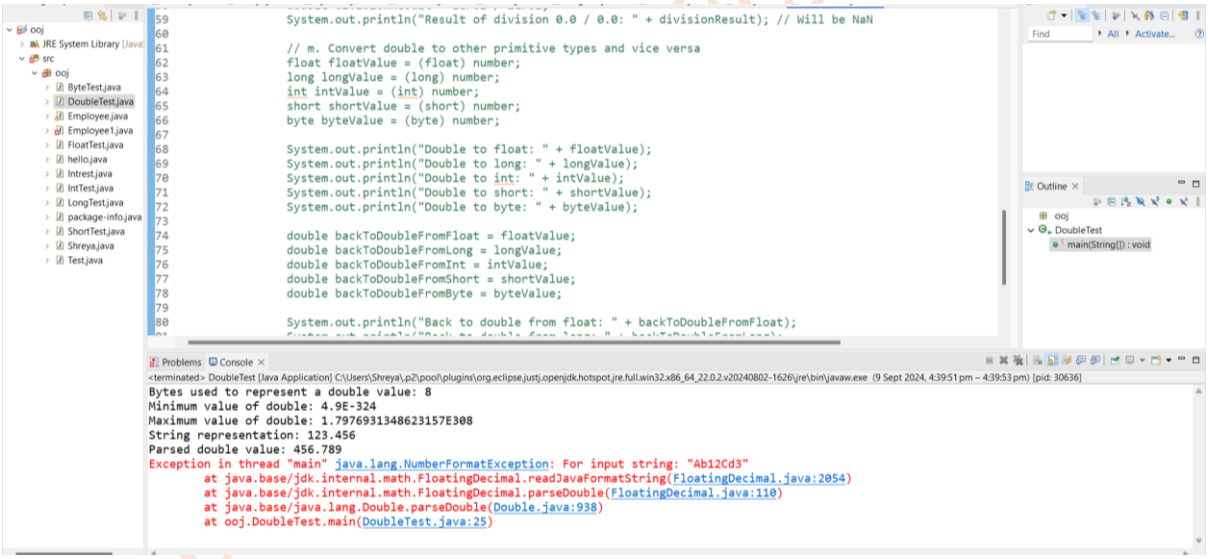
l. Declare two double variables with the same value, `0.0`, and divide them. (Hint: Observe the result and any special floating-point behavior).

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;
```

```
        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);
    }
}
```



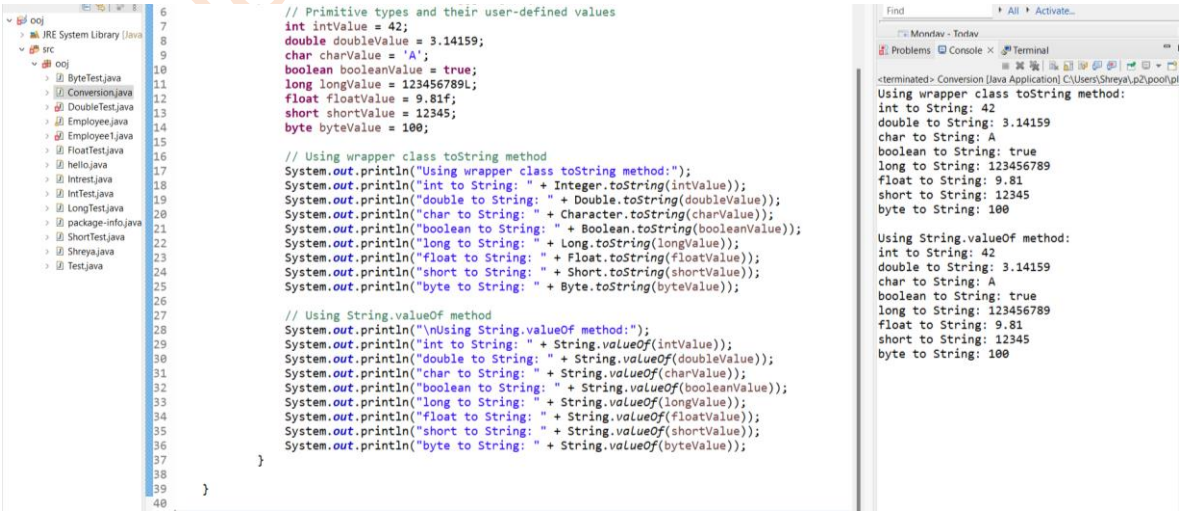
8. Conversion between Primitive Types and Strings

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
```

```
        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));
    }
}
```

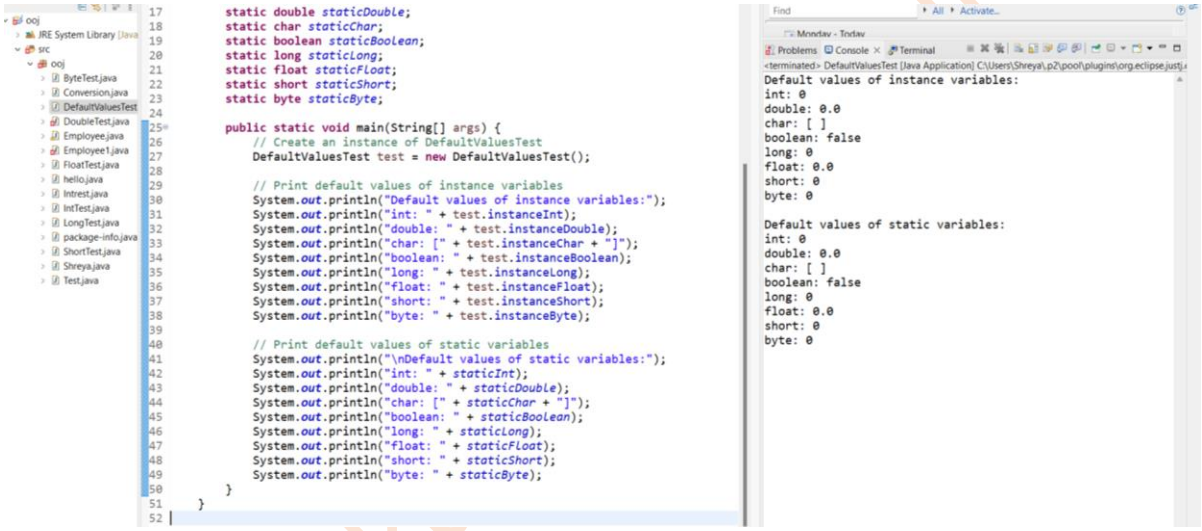


9. Default Values of Primitive Types

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 ooj;

    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);
    }
}
```

```
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).

```
package ooj;
public class Arithematicswitch {
   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);
    }
}
```