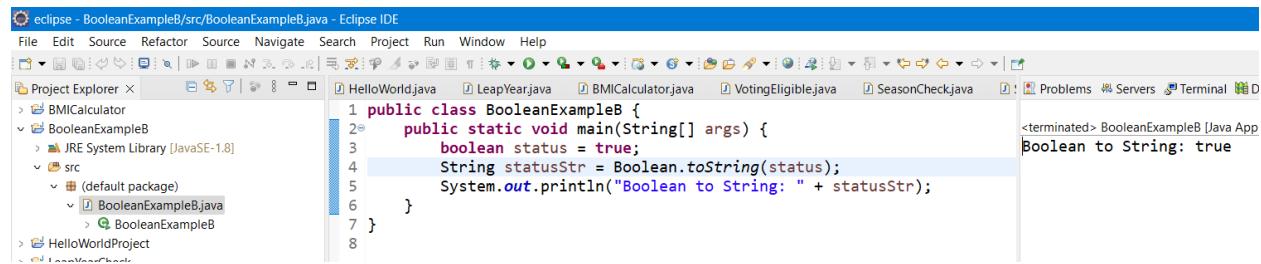


Assignment

1. Working with java.lang. Boolean

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

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)`).

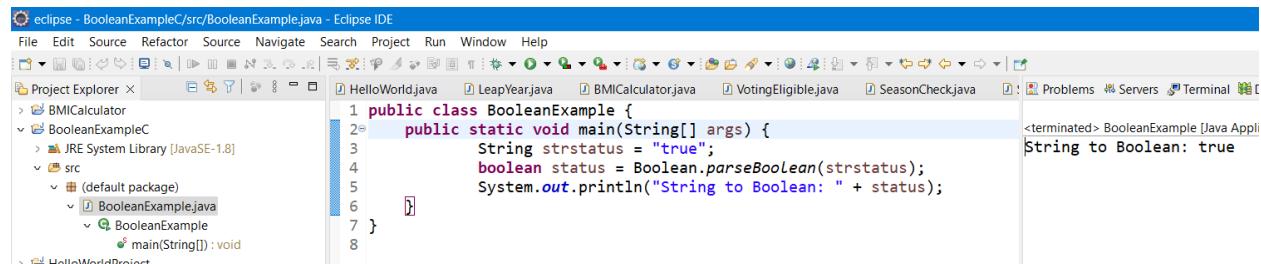


The screenshot shows the Eclipse IDE interface with the title bar "eclipse - BooleanExampleB/src/BooleanExampleB.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer on the left shows a project structure with files like HelloWorld.java, LeapYear.java, BMIcalculator.java, VotingEligible.java, SeasonCheck.java, and BooleanExampleB.java. The code editor window displays the following Java code:

```
1 public class BooleanExampleB {
2     public static void main(String[] args) {
3         boolean status = true;
4         String statusStr = Boolean.toString(status);
5         System.out.println("Boolean to String: " + statusStr);
6     }
7 }
```

The status bar at the bottom right shows the output: <terminated> BooleanExampleB [Java App] Boolean to String: true.

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



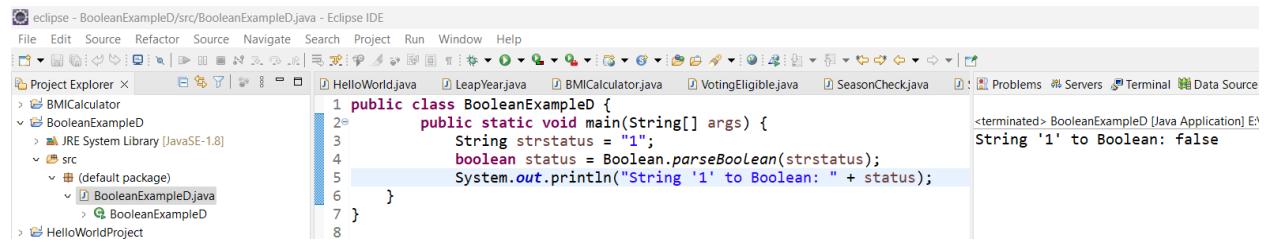
The screenshot shows the Eclipse IDE interface with the title bar "eclipse - BooleanExampleC/src/BooleanExample.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer on the left shows a project structure with files like HelloWorld.java, LeapYear.java, BMIcalculator.java, VotingEligible.java, SeasonCheck.java, and BooleanExample.java. The code editor window displays the following Java code:

```
1 public class BooleanExample {
2     public static void main(String[] args) {
3         String strstatus = "true";
4         boolean status = Boolean.parseBoolean(strstatus);
5         System.out.println("String to Boolean: " + status);
6     }
7 }
```

The status bar at the bottom right shows the output: <terminated> BooleanExample [Java App] String to Boolean: true.

d. Declare a method-local variable `strstatus` of type `String` with the value "1" or "0" and attempt to convert it to a `boolean`. (Hint: `parseBoolean`

method will not work as expected with "1" or "0").

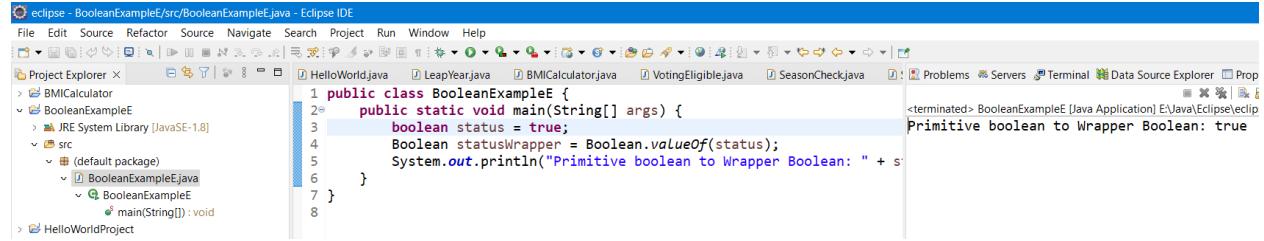


Screenshot of Eclipse IDE showing the code for BooleanExampleD.java. The code uses `Boolean.parseBoolean(strstatus)` to convert the string "1" to a boolean value, which is then printed as false.

```
public class BooleanExampleD {
    public static void main(String[] args) {
        String strstatus = "1";
        boolean status = Boolean.parseBoolean(strstatus);
        System.out.println("String '1' to Boolean: " + status);
    }
}
```

Output window shows: <terminated> BooleanExampleD [Java Application] E:\Java\Eclipse\...\src\BooleanExampleD.java:1: String '1' to Boolean: false

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)).

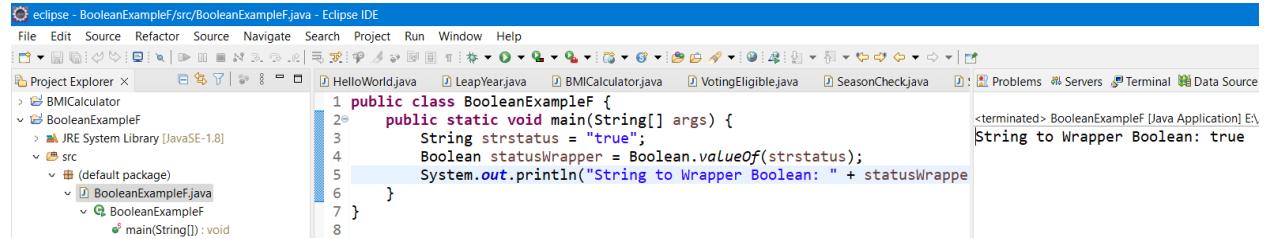


Screenshot of Eclipse IDE showing the code for BooleanExampleE.java. It converts a primitive boolean value to a Boolean wrapper object using `Boolean.valueOf(status)`.

```
public class BooleanExampleE {
    public static void main(String[] args) {
        boolean status = true;
        Boolean statusWrapper = Boolean.valueOf(status);
        System.out.println("Primitive boolean to Wrapper Boolean: " + s
    }
}
```

Output window shows: <terminated> BooleanExampleE [Java Application] E:\Java\Eclipse\...\src\BooleanExampleE.java:1: Primitive boolean to Wrapper Boolean: true

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



Screenshot of Eclipse IDE showing the code for BooleanExampleF.java. It converts a string "true" to a Boolean wrapper object using `Boolean.valueOf(strstatus)`.

```
public class BooleanExampleF {
    public static void main(String[] args) {
        String strstatus = "true";
        Boolean statusWrapper = Boolean.valueOf(strstatus);
        System.out.println("String to Wrapper Boolean: " + statusWrapper
    }
}
```

Output window shows: <terminated> BooleanExampleF [Java Application] E:\Java\Eclipse\...\src\BooleanExampleF.java:1: String to Wrapper Boolean: true

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

The image contains two side-by-side screenshots of the Eclipse IDE interface. Both screenshots show the 'BooleanExampleG' project in the Project Explorer. The top screenshot shows a Java file named 'BooleanExampleG.java' with the following code:

```

1 public class BooleanExampleG {
2     public static void main(String[] args) {
3         boolean status = true;
4         int boolAsInt = status ? 1 : 0;
5         System.out.println("Boolean to int: " + boolAsInt);
6     }
7 }

```

The bottom screenshot shows another Java file named 'BooleanExampleG.java' with the following code:

```

1 public class BooleanExampleG {
2     public static void main(String[] args) {
3         int num = 1;
4         boolean intAsBool = (num != 0);
5         System.out.println("int to Boolean: " + intAsBool);
6     }
7 }

```

In both cases, the output terminal shows the results of the program execution.

2. Working with `java.lang. Byte`

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

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

The screenshot shows the 'ByteExampleB' project in the Project Explorer. The 'ByteExampleB.java' file contains the following code:

```

1 public class ByteExampleB {
2     public static void main(String[] args) {
3         System.out.println("Bytes used by byte: " + Byte.BYTES);
4     }
5 }

```

The output terminal shows the result of the program execution.

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

The screenshot shows the 'ByteExampleC' project in the Project Explorer. The 'ByteExampleC.java' file contains the following code:

```

1 public class ByteExampleC {
2     public static void main(String[] args) {
3         System.out.println("Minimum value of byte: " + Byte.MIN_VALUE);
4         System.out.println("Maximum value of byte: " + Byte.MAX_VALUE);
5     }
6 }

```

The output terminal shows the results of the program execution.

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)).

Screenshot of Eclipse IDE showing the ByteExampleD.java file. The code converts a byte value to a string:

```
1 public class ByteExampleD {  
2     public static void main(String[] args) {  
3         byte number = 8;  
4         String numberStr = Byte.toString(number);  
5         System.out.println("Byte to String: " + numberStr);  
6     }  
7 }  
8  
9  
10  
11 }
```

The output window shows: <terminated> ByteExampleD [Java Application] Byte to String: 8

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)).

Screenshot of Eclipse IDE showing the ByteExampleE.java file. The code converts a string to a byte:

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

The output window shows: <terminated> ByteExampleE [Java Application] String to Byte: 12

f. Declare a method-local variable strNumber of type string with the value "Ab12cd3" and attempt to convert it to a byte value. (Hint: parseByte method will throw a NumberFormatException).

Screenshot of Eclipse IDE showing the ByteExample.java file. The code attempts to convert an invalid string to a byte, resulting in a NumberFormatException:

```
1 public class ByteExample {  
2     public static void main(String[] args) {  
3         try {  
4             String strNumber = "Ab12cd3";  
5             byte number = Byte.parseByte(strNumber);  
6             System.out.println("String to Byte: " + number);  
7         } catch (NumberFormatException e) {  
8             System.out.println("NumberFormatException: " + e.getMessage());  
9         }  
10    }  
11 }  
12  
13  
14  
15  
16 }
```

The output window shows: <terminated> ByteExample [Java Application] E:\Java\Eclipse\plugins\org.eclipse.jdt.core\src\ByteExample.java: NumberFormatException: For input string: "Ab12cd3"

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 ByteG {
    public static void main(String[] args) {
        byte number = 11;
        Byte byteWrapper = Byte.valueOf(number);
        System.out.println("Primitive byte to Wrapper Byte: " + byteWrapper);
    }
}

```

<terminated> ByteG [Java Application] E:\Java\Eclipse\ecide Primitive byte to Wrapper Byte: 11

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 ByteH {
    public static void main(String[] args) {
        String strNumber = "10";
        Byte byteWrapper = Byte.valueOf(strNumber);
        System.out.println("String to Wrapper Byte: " + byteWrapper);
    }
}

```

<terminated> ByteH [Java Application] E:\Java\Eclipse\ecide String to Wrapper Byte: 10

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

```

public class ByteI {
    public static void main(String[] args) {
        int number = 120;
        byte numberAsByte = (byte) number;
        System.out.println("Int to Byte: " + numberAsByte);
    }
}

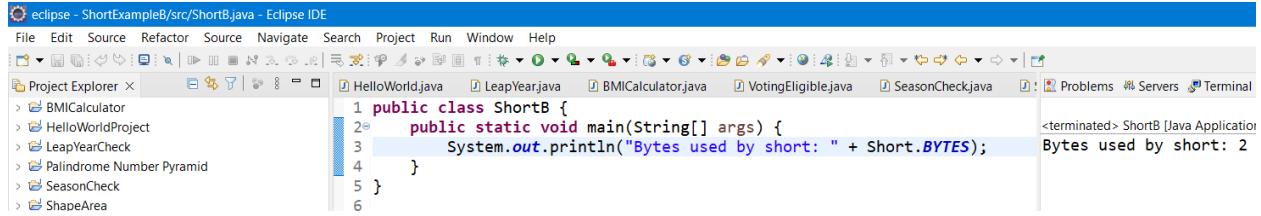
```

<terminated> ByteI [Java Application] E:\Java\Eclipse\ecide Int to Byte: 120

3. Working with java.lang.Short

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

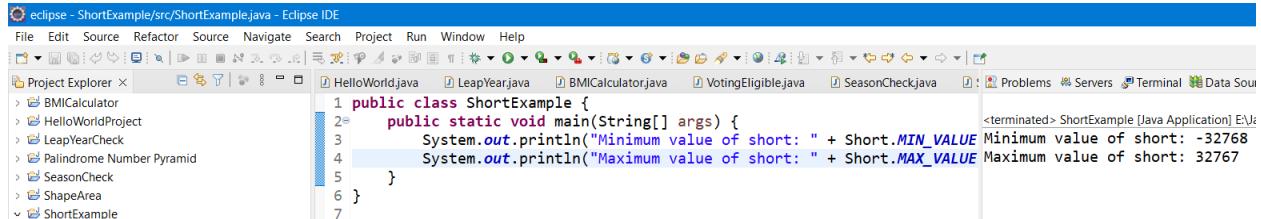
b. Write a program to test how many bytes are used to represent a short value using the BYTES field. (Hint: Use Short.BYTES).



```
eclipse - ShortExampleB/src/ShortB.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer X HelloWorld.java LeapYear.java BMICalculator.java VotingEligible.java SeasonCheck.java Problems Servers Terminal
1 public class ShortB {
2     public static void main(String[] args) {
3         System.out.println("Bytes used by short: " + Short.BYTES);
4     }
5 }
6
```

<terminated> ShortB [Java Application] Bytes used by short: 2

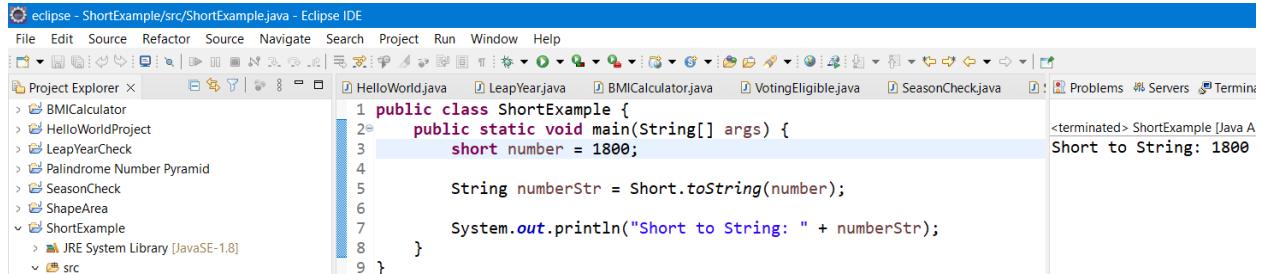
c. Write a program to find the minimum and maximum values of short using the MIN VALUE and MAX VALUE fields. (Hint: Use Short.MIN_VALUE and Short.MAX_VALUE).



```
eclipse - ShortExample/src/ShortExample.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer X HelloWorld.java LeapYear.java BMICalculator.java VotingEligible.java SeasonCheck.java Problems Servers Terminal Data Sou
1 public class ShortExample {
2     public static void main(String[] args) {
3         System.out.println("Minimum value of short: " + Short.MIN_VALUE);
4         System.out.println("Maximum value of short: " + Short.MAX_VALUE);
5     }
6 }
7
```

<terminated> ShortExample [Java Application] E:\Java\ShortExample\src Minimum value of short: -32768 Maximum value of short: 32767

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)).



```
eclipse - ShortExample/src/ShortExample.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer X HelloWorld.java LeapYear.java BMICalculator.java VotingEligible.java SeasonCheck.java Problems Servers Terminal
1 public class ShortExample {
2     public static void main(String[] args) {
3         short number = 1800;
4
5         String numberStr = Short.toString(number);
6
7         System.out.println("Short to String: " + numberStr);
8     }
9 }
```

<terminated> ShortExample [Java Application] Short to String: 1800

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)).`

The screenshot shows the Eclipse IDE interface with the title bar "eclipse - ShortExample/src/ShortExample.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer view on the left lists several Java projects and files. The code editor in the center contains the following Java code:

```
1 public class ShortExample {
2     public static void main(String[] args) {
3         String strNumber = "1103";
4
5         short number = Short.parseShort(strNumber);
6
7         System.out.println("String to Short: " + number);
8     }
9 }
```

The status bar at the bottom right shows the output: <terminated> ShortExample [Java Application] String to Short: 1103.

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).

The screenshot shows the Eclipse IDE interface with the title bar "eclipse - ShortExample/src/ShortExample.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer view on the left lists several Java projects and files. The code editor in the center contains the following Java code:

```
1 public class ShortExample {
2     public static void main(String[] args) {
3         try {
4             String strNumber = "Ab12cd3";
5
6             short number = Short.parseShort(strNumber);
7
8             System.out.println("String to Short: " + number);
9         } catch (NumberFormatException e) {
10             System.out.println("NumberFormatException: " + e.getMessage());
11         }
12     }
13 }
```

The status bar at the bottom right shows the output: <terminated> ShortExample [Java Application] E:\Java\Eclipse\plugins\org.eclipse.jdt.core\src\ShortExample.java:10: NumberFormatException: For input string: "Ab12cd3".

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)).

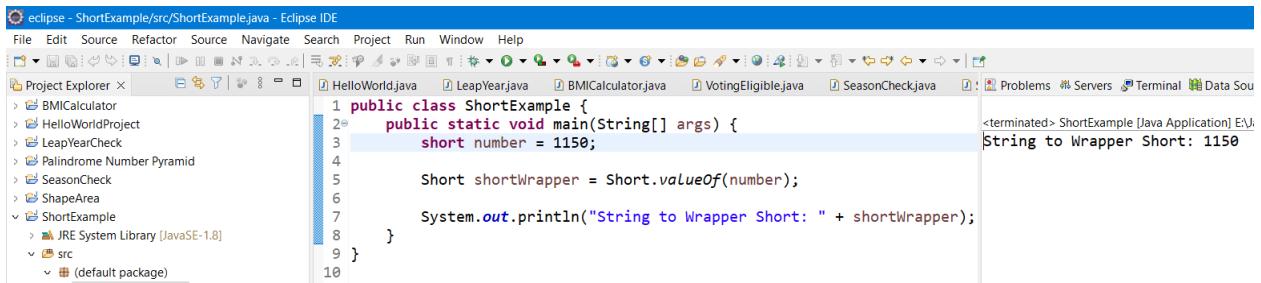
The screenshot shows the Eclipse IDE interface with the title bar "eclipse - ShortExample/src/ShortExample.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer view on the left lists several Java projects and files. The code editor in the center contains the following Java code:

```
1 public class ShortExample {
2     public static void main(String[] args) {
3         short number = 1250;
4
5         Short shortWrapper = Short.valueOf(number);
6
7         System.out.println("Primitive short to Wrapper Short: " + shortWrapper);
8     }
9 }
```

The status bar at the bottom right shows the output: <terminated> ShortExample [Java Application] E:\Java\Eclipse\plugins\org.eclipse.jdt.core\src\ShortExample.java:7: Primitive short to Wrapper Short: 1250.

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().

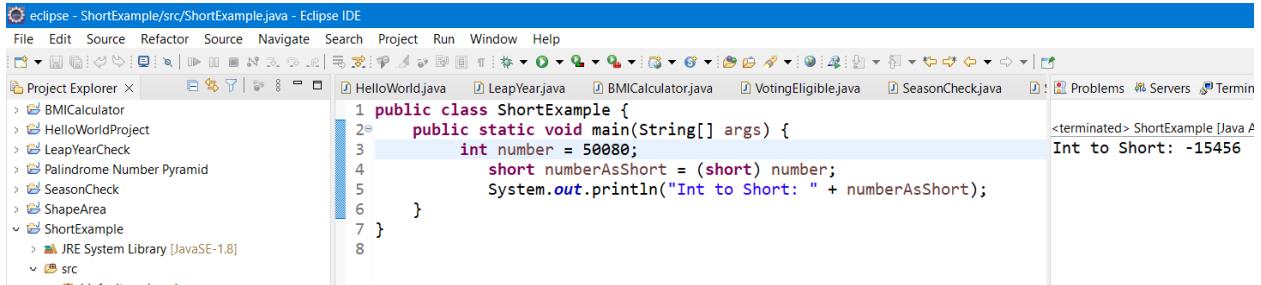
(Hint: Use Short.valueOf (String)).



Screenshot of Eclipse IDE showing a Java file named ShortExample.java. The code converts a string "1150" to a short integer using the Short.valueOf method. The output window shows the result: "String to Wrapper Short: 1150".

```
1 public class ShortExample {
2     public static void main(String[] args) {
3         short number = 1150;
4
5         Short shortWrapper = Short.valueOf(number);
6
7         System.out.println("String to Wrapper Short: " + shortWrapper);
8     }
9 }
10
```

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



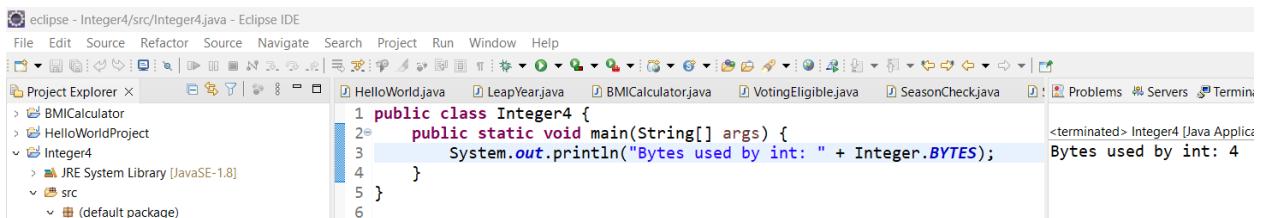
Screenshot of Eclipse IDE showing a Java file named ShortExample.java. The code converts an integer 50080 to a short integer using a cast operator. The output window shows the result: "Int to Short: -15456".

```
1 public class ShortExample {
2     public static void main(String[] args) {
3         int number = 50080;
4         short numberAsShort = (short) number;
5         System.out.println("Int to Short: " + numberAsShort);
6     }
7 }
8
```

4. Working with java.lang. Integer

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

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



Screenshot of Eclipse IDE showing a Java file named Integer4.java. The code prints the value of Integer.BYTES, which is 4. The output window shows the result: "Bytes used by int: 4".

```
1 public class Integer4 {
2     public static void main(String[] args) {
3         System.out.println("Bytes used by int: " + Integer.BYTES);
4     }
5 }
6
```

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).

```

eclipse - Integer4/src/Integer4.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer HelloWord.java LeapYear.java BMIcalculator.java VotingEligible.java SeasonCheck.java Problems Servers Terminal Data Source Ex
HelloWorldProject JRE System Library [JavaSE-1.8]
Integer4 src (default package) Integer4.java
1 public class Integer4 {
2     public static void main(String[] args) {
3         System.out.println("Minimum value of int: " + Integer.MIN_VALUE)
4         System.out.println("Maximum value of int: " + Integer.MAX_VALUE)
5     }
6 }
7

```

```

<terminated> Integer4 [Java Application] E:\Java\Eclipse
Minimum value of int: -2147483648
Maximum value of int: 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)).

```

eclipse - Integer4/src/Integer4.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer HelloWord.java LeapYear.java BMIcalculator.java VotingEligible.java SeasonCheck.java Problems Servers Terminal
HelloWorldProject JRE System Library [JavaSE-1.8]
Integer4 src (default package) Integer4.java Integer4.java main(String[])
1 public class Integer4 {
2     public static void main(String[] args) {
3         int number = 12345;
4
5         String numberStr = Integer.toString(number);
6         System.out.println("Int to String: " + numberStr);
7     }
8 }
9 main(String[])

```

```

<terminated> Integer4 [Java Application]
Int to String: 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)).

```

eclipse - Integer4/src/Integer4.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Project Explorer HelloWord.java LeapYear.java BMIcalculator.java VotingEligible.java SeasonCheck.java Problems Servers Terminal
HelloWorldProject JRE System Library [JavaSE-1.8]
Integer4 src (default package) Integer4.java Integer4.java main(String[])
1 public class Integer4 {
2     public static void main(String[] args) {
3         String strNumber = "12345";
4
5         int number = Integer.parseInt(strNumber);
6
7         System.out.println("String to Int: " + number);
8     }
9 }
10 main(String[])

```

```

<terminated> Integer4 [Java Application]
String to Int: 12345

```

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).

Screenshot of Eclipse IDE showing the code for Integer4.java:

```

1 public class Integer4 {
2     public static void main(String[] args) {
3         try {
4             String strNumber = "Ab12Cd3";
5             int number = Integer.parseInt(strNumber);
6             System.out.println("String to Int: " + number);
7         } catch (NumberFormatException e) {
8             System.out.println("NumberFormatException: " + e.getMessage());
9         }
10    }
11 }
12
13
14

```

The status bar at the bottom right shows the output: <terminated> Integer4 [Java Application] E:\Java\Eclipse\plugins\org.eclipse NumberFormatException: For input string: "Ab12Cd3"

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)).

Screenshot of Eclipse IDE showing the code for Integer4.java:

```

1 public class Integer4 {
2     public static void main(String[] arg) {
3         int number = 12345;
4         Integer integerWrapper = Integer.valueOf(number);
5         System.out.println("Primitive int to Wrapper Integer: " + integerWrapper);
6     }
7 }
8
9

```

The status bar at the bottom right shows the output: <terminated> Integer4 [Java Application] E:\Java\Eclipse\plugins\org.eclipse Primitive int to Wrapper Integer: 12345

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)).

Screenshot of Eclipse IDE showing the code for Integer4.java:

```

1 public class Integer4 {
2     public static void main(String[] arg) {
3         String strNumber = "22345";
4         Integer integerWrapper = Integer.valueOf(strNumber);
5         System.out.println("String to Wrapper Integer: " + integerWrapper);
6     }
7 }
8
9

```

The status bar at the bottom right shows the output: <terminated> Integer4 [Java Application] E:\Java\Eclipse\plugins\org.eclipse String to Wrapper Integer: 22345

- 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)).

The screenshot shows the Eclipse IDE interface with the title bar "eclipse - Integer4/src/Integer4.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer on the left shows a project named "Integer4" with a "src" folder containing "Integer4.java". The code editor displays the following Java code:

```

1 public class Integer4 {
2     public static void main(String[] arg) {
3         int a = 10;
4         int b = 20;
5         int sum = Integer.sum(a, b);
6
7         System.out.println("Sum: " + sum);
8     }
9 }

```

The status bar on the right shows "<terminated> Integ Sum: 30".

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

The screenshot shows the Eclipse IDE interface with the title bar "eclipse - Integer4/src/Integer4.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer on the left shows a project named "Integer4" with a "src" folder containing "Integer4.java". The code editor displays the following Java code:

```

1 public class Integer4 {
2     public static void main(String[] arg) {
3         int a = 10;
4         int b = 20;
5
6         int minValue = Integer.min(a, b);
7         int maxValue = Integer.max(a, b);
8
9         System.out.println("Minimum: " + minValue);
10        System.out.println("Maximum: " + maxValue);
11    }
12 }

```

The status bar on the right shows "<terminated> Integer4 Minimum: 10 Maximum: 20".

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

The screenshot shows the Eclipse IDE interface with the title bar "eclipse - Integer4/src/Integer4.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer on the left shows a project named "Integer4" with a "src" folder containing "Integer4.java". The code editor displays the following Java code:

```

1 public class Integer4 {
2     public static void main(String[] arg) {
3         int number = 7;
4
5         String binary = Integer.toBinaryString(number);
6         String octal = Integer.toOctalString(number);
7         String hex = Integer.toHexString(number);
8
9         System.out.println("Binary: " + binary);
10        System.out.println("Octal: " + octal);
11        System.out.println("Hexadecimal: " + hex);
12    }
13 }

```

The status bar on the right shows "<terminated> Integer4 [Binary: 111 Octal: 7 Hexadecimal: 7".

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

The screenshot shows the Eclipse IDE interface with the title bar "eclipse - Integer4/src/Integer4.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer on the left lists several Java files: HelloWorld.java, LeapYear.java, BMIcalculator.java, VotingEligible.java, SeasonCheck.java, ShapeA.java, Problems, Servers, Terminal, JRE System Library [JavaSE-1.8], and Integer4.java. The Integer4.java file is selected and shown in the editor area. The code is as follows:

```
1 public class Integer4 {
2     public static void main(String[] args) {
3         int number = 300;
4
5         long longValue = (long) number;
6         System.out.println("int to long: " + longValue);
7
8         float floatValue = (float) number;
9         System.out.println("int to float: " + floatValue);
10
11         double doubleValue = (double) number;
12         System.out.println("int to double: " + doubleValue);
13
14         byte byteValue = (byte) number;
15         System.out.println("int to byte: " + byteValue);
16     }
17 }
18
```

The right side of the interface shows the terminal window with the output of the program:

```
<terminated> Integer4 [Java Application]
int to long: 300
int to float: 300.0
int to double: 300.0
int to byte: 44
```

5. Working with java.lang. Long

a. Explore the lava API documentation for java.lang. Long and observe its modifiers and super types. (Reading)

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

The screenshot shows the Eclipse IDE interface with the title bar "eclipse - Long5/src/Long5.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer on the left lists several Java files: HelloWorld.java, LeapYear.java, BMIcalculator.java, VotingEligible.java, SeasonCheck.java, ShapeA.java, Problems, Servers, Terminal, JRE System Library [JavaSE-1.8], and Long5.java. The Long5.java file is selected and shown in the editor area. The code is as follows:

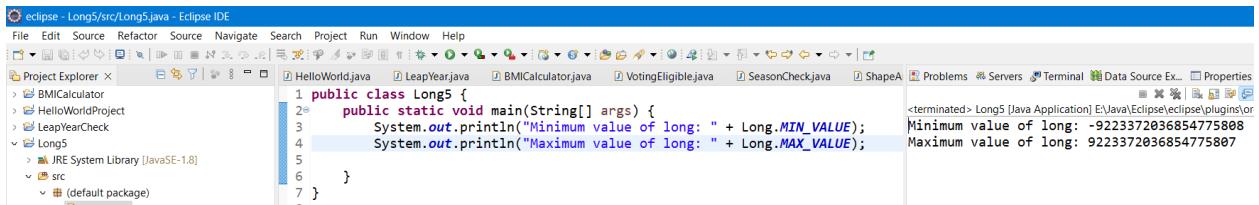
```
1 public class Long5 {
2     public static void main(String[] args) {
3         System.out.println("Bytes used to represent a long: " + Long.BYTES);
4     }
5 }
6
7
```

The right side of the interface shows the terminal window with the output of the program:

```
<terminated> Long5 [Java Application]
Bytes used to represent a long: 8
```

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).`



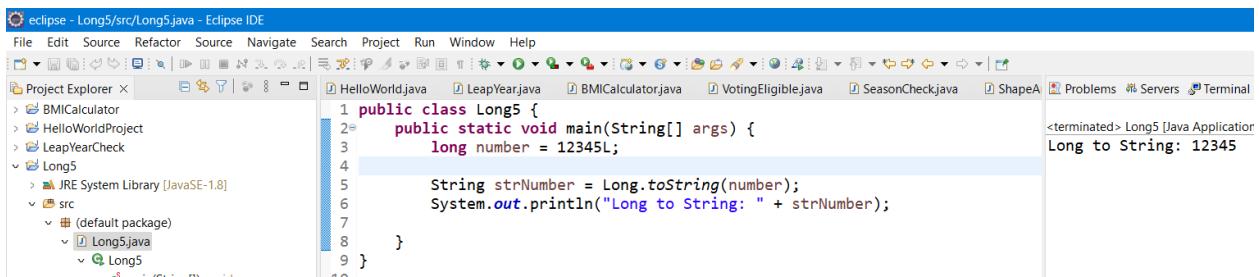
The screenshot shows the Eclipse IDE interface with the title bar "eclipse - Long5/src/Long5.java - Eclipse IDE". The Project Explorer shows a project named "Long5" with files like HelloWorld.java, LeapYear.java, BMIcalculator.java, VotingEligible.java, SeasonCheck.java, and ShapeA.java. The code editor displays the following Java code:

```
1 public class Long5 {
2     public static void main(String[] args) {
3         System.out.println("Minimum value of long: " + Long.MIN_VALUE);
4         System.out.println("Maximum value of long: " + Long.MAX_VALUE);
5     }
6 }
7 }
```

The terminal window shows the output of the application:

```
<terminated> Long5 [Java Application] E:\Java\Eclipse\plugins\org  
Minimum value of long: -9223372036854775808  
Maximum value of long: 9223372036854775807
```

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)`).



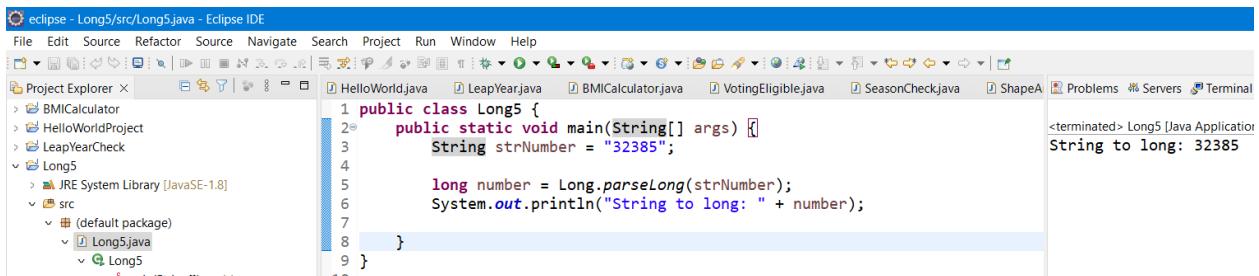
The screenshot shows the Eclipse IDE interface with the title bar "eclipse - Long5/src/Long5.java - Eclipse IDE". The Project Explorer shows a project named "Long5" with files like HelloWorld.java, LeapYear.java, BMIcalculator.java, VotingEligible.java, SeasonCheck.java, and ShapeA.java. The code editor displays the following Java code:

```
1 public class Long5 {
2     public static void main(String[] args) {
3         long number = 12345L;
4
5         String strNumber = Long.toString(number);
6         System.out.println("Long to String: " + strNumber);
7     }
8 }
9 }
```

The terminal window shows the output of the application:

```
<terminated> Long5 [Java Application]  
Long to String: 12345
```

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)`).



The screenshot shows the Eclipse IDE interface with the title bar "eclipse - Long5/src/Long5.java - Eclipse IDE". The Project Explorer shows a project named "Long5" with files like HelloWorld.java, LeapYear.java, BMIcalculator.java, VotingEligible.java, SeasonCheck.java, and ShapeA.java. The code editor displays the following Java code:

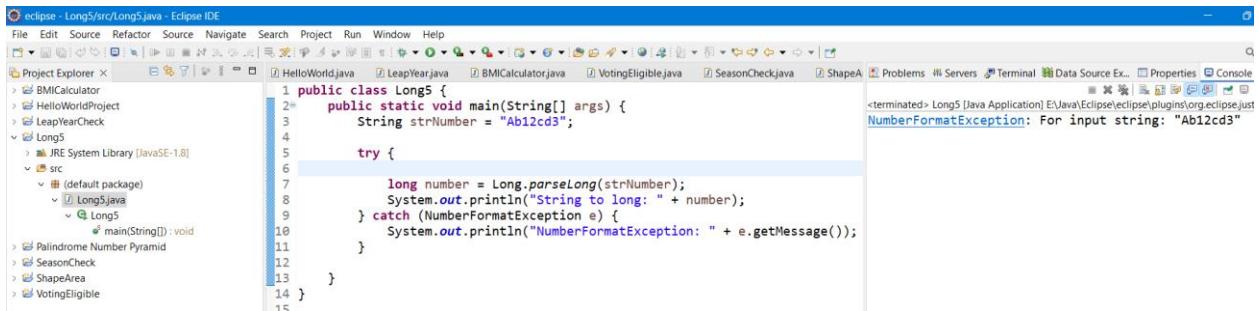
```
1 public class Long5 {
2     public static void main(String[] args) {
3         String strNumber = "32385";
4
5         long number = Long.parseLong(strNumber);
6         System.out.println("String to long: " + number);
7     }
8 }
9 }
10 }
```

The terminal window shows the output of the application:

```
<terminated> Long5 [Java Application]  
String to long: 32385
```

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).

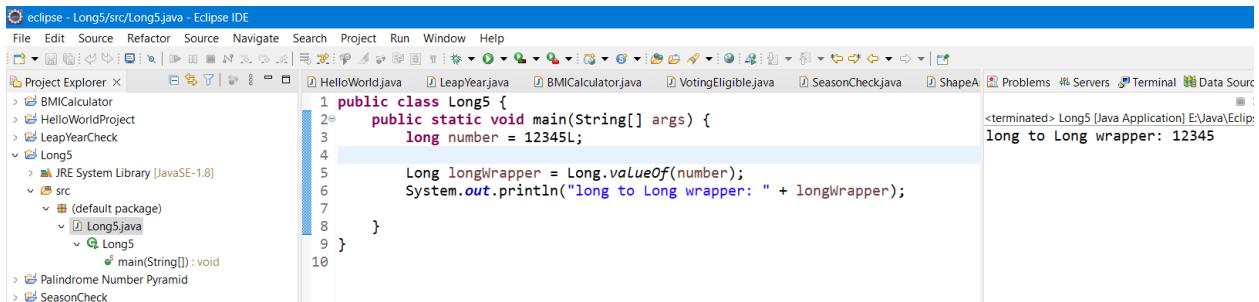


The screenshot shows the Eclipse IDE interface with the title bar "eclipse - Long5/src/Long5.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer view on the left shows a project named "Long5" with several Java files like HelloWorld.java, LeapYear.java, etc. The main editor window contains the following code:

```
1 public class Long5 {
2     public static void main(String[] args) {
3         String strNumber = "Ab12cd3";
4
5         try {
6             long number = Long.parseLong(strNumber);
7             System.out.println("String to long: " + number);
8         } catch (NumberFormatException e) {
9             System.out.println("NumberFormatException: " + e.getMessage());
10        }
11    }
12}
13}
14}
```

The status bar at the bottom right says "<terminated> Long5 [Java Application] E:\Java\Eclipse\plugins\org.eclipse.jdt.core\src\Long5.java: NumberFormatException: For input string: \"Ab12cd3\"".

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)).

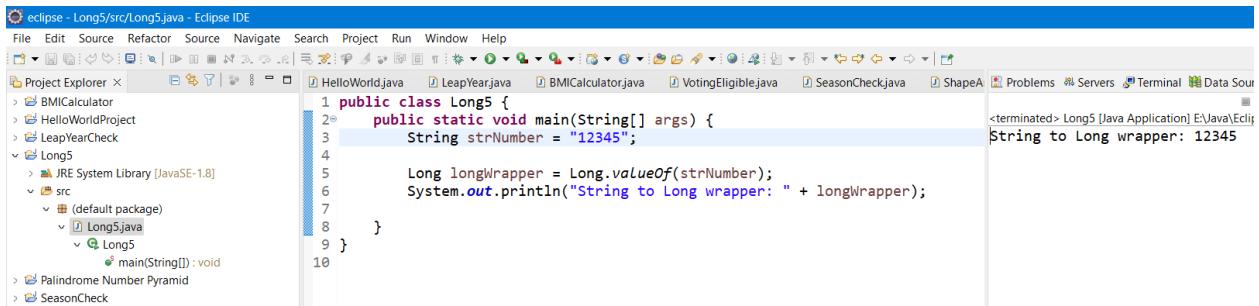


The screenshot shows the Eclipse IDE interface with the title bar "eclipse - Long5/src/Long5.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer view on the left shows a project named "Long5" with several Java files like HelloWorld.java, LeapYear.java, etc. The main editor window contains the following code:

```
1 public class Long5 {
2     public static void main(String[] args) {
3         long number = 12345L;
4
5         Long longWrapper = Long.valueOf(number);
6         System.out.println("long to Long wrapper: " + longWrapper);
7     }
8 }
9 }
```

The status bar at the bottom right says "<terminated> Long5 [Java Application] E:\Java\Eclipse\plugins\org.eclipse.jdt.core\src\Long5.java: long to Long wrapper: 12345L".

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)).

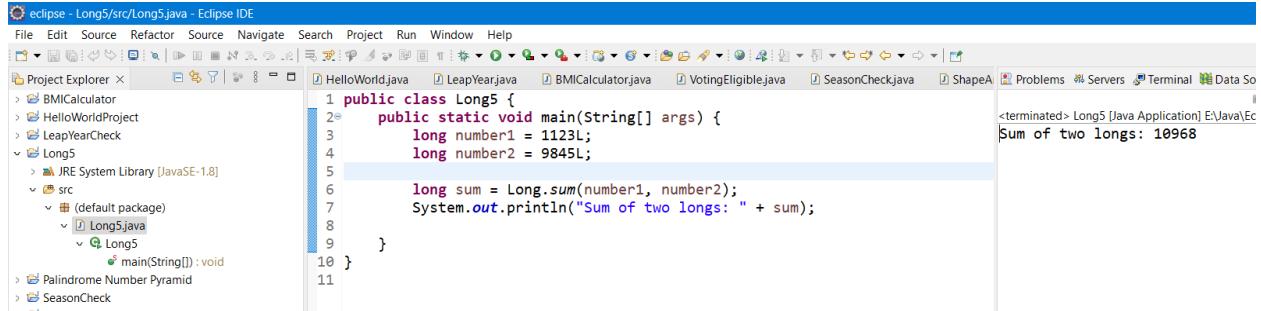


The screenshot shows the Eclipse IDE interface with the title bar "eclipse - Long5/src/Long5.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The Project Explorer view on the left shows a project named "Long5" with several Java files like HelloWorld.java, LeapYear.java, etc. The main editor window contains the following code:

```
1 public class Long5 {
2     public static void main(String[] args) {
3         String strNumber = "12345";
4
5         Long longWrapper = Long.valueOf(strNumber);
6         System.out.println("String to Long wrapper: " + longWrapper);
7     }
8 }
9 }
```

The status bar at the bottom right says "<terminated> Long5 [Java Application] E:\Java\Eclipse\plugins\org.eclipse.jdt.core\src\Long5.java: String to Long wrapper: 12345".

- 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))



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

- Project Explorer:** Shows a project named "Long5" with a "src" folder containing a "Long5.java" file.
- Code Editor:** Displays the following Java code in the "Long5.java" file:

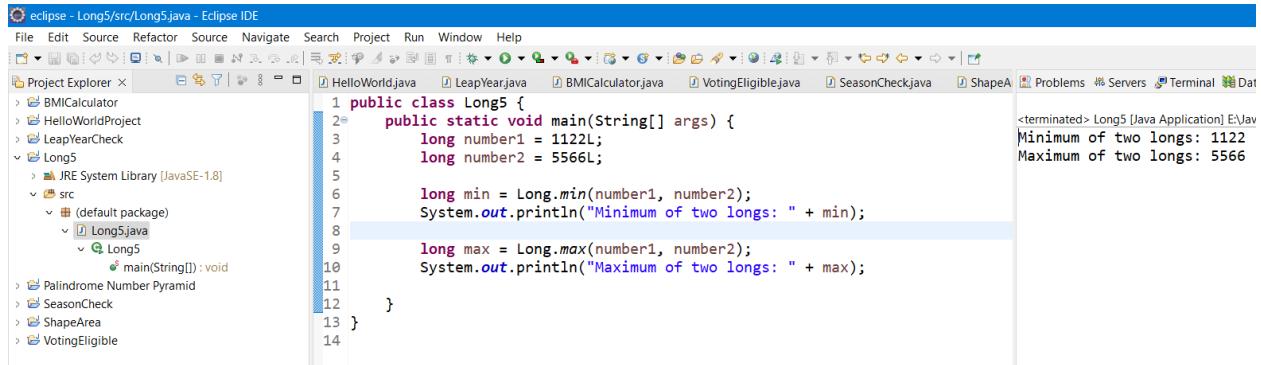
```

1 public class Long5 {
2     public static void main(String[] args) {
3         long number1 = 1123L;
4         long number2 = 9845L;
5
6         long sum = Long.sum(number1, number2);
7         System.out.println("Sum of two longs: " + sum);
8     }
9 }
10
11

```

- Terminal:** Shows the output of the application: "Sum of two longs: 10968".

- 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)).



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

- Project Explorer:** Shows a project named "Long5" with a "src" folder containing a "Long5.java" file.
- Code Editor:** Displays the following Java code in the "Long5.java" file:

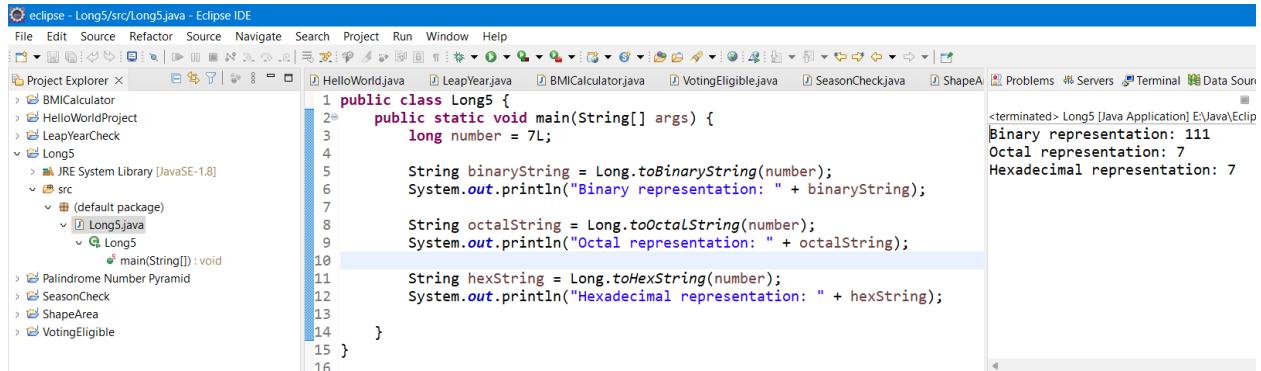
```

1 public class Long5 {
2     public static void main(String[] args) {
3         long number1 = 1122L;
4         long number2 = 5566L;
5
6         long min = Long.min(number1, number2);
7         System.out.println("Minimum of two longs: " + min);
8
9         long max = Long.max(number1, number2);
10        System.out.println("Maximum of two longs: " + max);
11
12    }
13 }
14

```

- Terminal:** Shows the output of the application: "Minimum of two longs: 1122" and "Maximum of two longs: 5566".

- 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)).



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

- Project Explorer:** Shows a project named "Long5" with a "src" folder containing a "Long5.java" file.
- Code Editor:** Displays the following Java code in the "Long5.java" file:

```

1 public class Long5 {
2     public static void main(String[] args) {
3         long number = 7L;
4
5         String binaryString = Long.toBinaryString(number);
6         System.out.println("Binary representation: " + binaryString);
7
8         String octalString = Long.toOctalString(number);
9         System.out.println("Octal representation: " + octalString);
10
11        String hexString = Long.toHexString(number);
12        System.out.println("Hexadecimal representation: " + hexString);
13
14    }
15 }
16

```

- Terminal:** Shows the output of the application: "Binary representation: 111", "Octal representation: 7", and "Hexadecimal representation: 7".

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

```
eclipse - Long5/src/Long5.java - Eclipse IDE
File Edit Source Refactor Source Navigate Project Run Window Help
Project Explorer  HelloWorldjava  LeapYearjava  BMICalculatorjava  VotingEligible.java  SeasonCheckjava  ShapeA  Problems  Servers  Terminal
1 public class Long5 {
2     public static void main(String[] args) {
3         long number = 12345L;
4
5         int intValue = (int) number;
6         System.out.println("long to int: " + intValue);
7
8         float floatValue = (float) number;
9         System.out.println("long to float: " + floatValue);
10
11        double doubleValue = (double) number;
12        System.out.println("long to double: " + doubleValue);
13
14        byte byteValue = (byte) number;
15        System.out.println("long to byte: " + byteValue);
16    }
17 }
18
```

```
<terminated> Long5 [Java Application] E:
long to int: 12345
long to float: 12345.0
long to double: 12345.0
long to byte: 57
```

6. Working with java.lang. Float

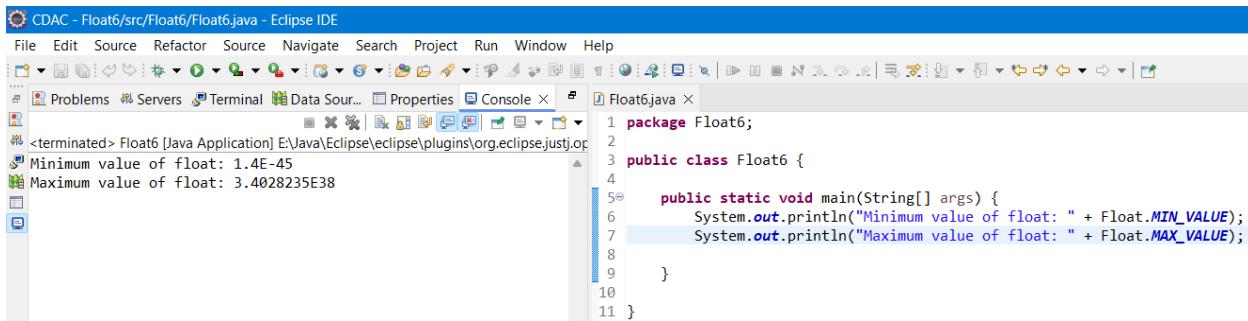
a. Explore the Java API documentation for java.lang. Float and observe its modifiers and super types. (Reading)

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

```
CDAC - Float6/src/Float6/Float6.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Problems  Servers  Terminal  Data Sour...  Properties  Console X  Float6.java X
<terminated> Float6 [Java Application] E:\Java\Eclipse\plugins\org.eclipse.justj.op
Bytes used to represent a float: 4
1 package Float6;
2
3 public class Float6 {
4
5     public static void main(String[] args) {
6         System.out.println("Bytes used to represent a float: " + Float.BYTES);
7     }
8 }
9
10 }
```

c. Write a program to find the minimum and maximum values of float using the MIN VALUE and MAX VALUE fields. (Hint: Use Float. MIN VALUE and

Float.MAX_VALUE).

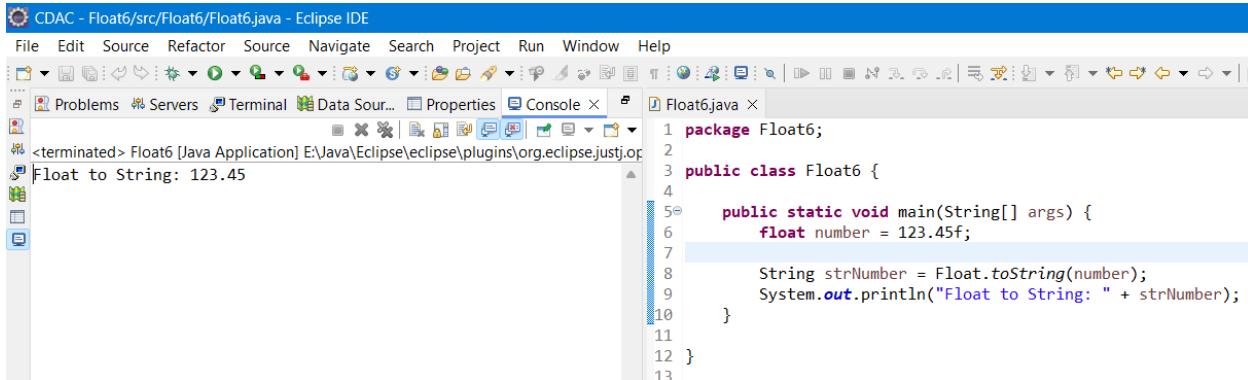


The screenshot shows the Eclipse IDE interface with the title bar "CDAC - Float6/src/Float6/Float6.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left sidebar shows "Problems", "Servers", "Terminal", "Data Sour...", "Properties", and "Console X". The right sidebar shows "Float6.java X". The code editor contains the following Java code:

```
1 package Float6;
2
3 public class Float6 {
4
5     public static void main(String[] args) {
6         System.out.println("Minimum value of float: " + Float.MIN_VALUE);
7         System.out.println("Maximum value of float: " + Float.MAX_VALUE);
8     }
9
10 }
```

The "Console" tab shows the output: "Minimum value of float: 1.4E-45" and "Maximum value of float: 3.4028235E38".

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

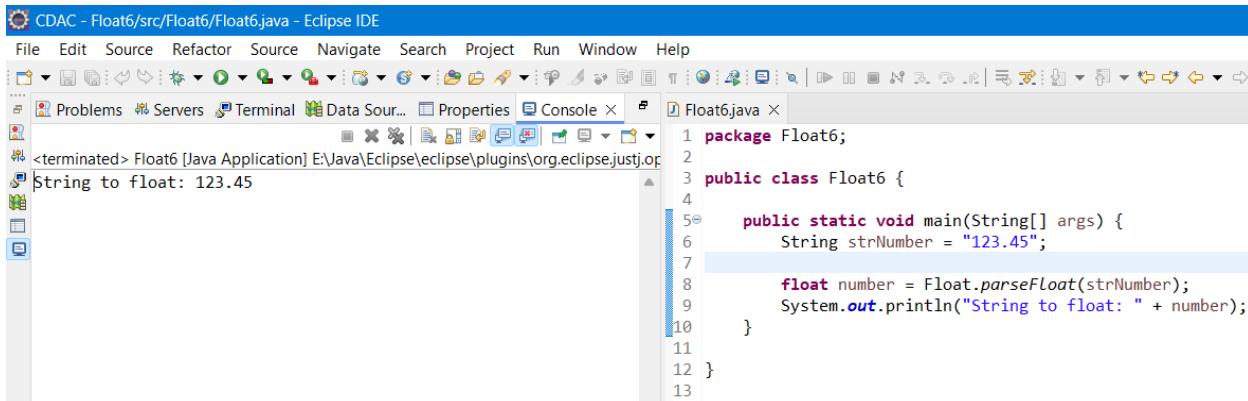


The screenshot shows the Eclipse IDE interface with the title bar "CDAC - Float6/src/Float6/Float6.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left sidebar shows "Problems", "Servers", "Terminal", "Data Sour...", "Properties", and "Console X". The right sidebar shows "Float6.java X". The code editor contains the following Java code:

```
1 package Float6;
2
3 public class Float6 {
4
5     public static void main(String[] args) {
6         float number = 123.45f;
7
8         String strNumber = Float.toString(number);
9         System.out.println("Float to String: " + strNumber);
10    }
11
12 }
13
```

The "Console" tab shows the output: "Float to String: 123.45".

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)`).

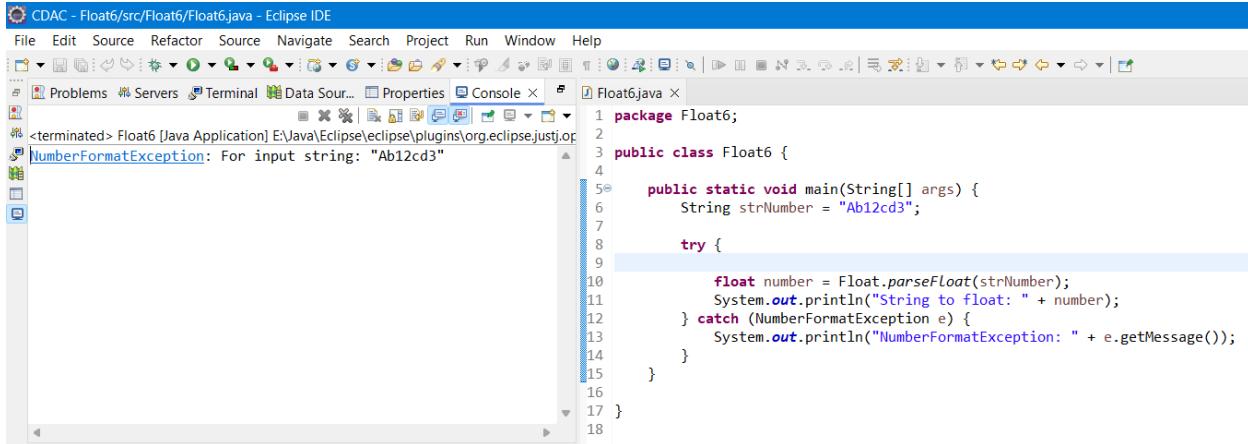


The screenshot shows the Eclipse IDE interface with the title bar "CDAC - Float6/src/Float6/Float6.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left sidebar shows "Problems", "Servers", "Terminal", "Data Sour...", "Properties", and "Console X". The right sidebar shows "Float6.java X". The code editor contains the following Java code:

```
1 package Float6;
2
3 public class Float6 {
4
5     public static void main(String[] args) {
6         String strNumber = "123.45";
7
8         float number = Float.parseFloat(strNumber);
9         System.out.println("String to float: " + number);
10    }
11
12 }
13
```

The "Console" tab shows the output: "String to float: 123.45".

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).

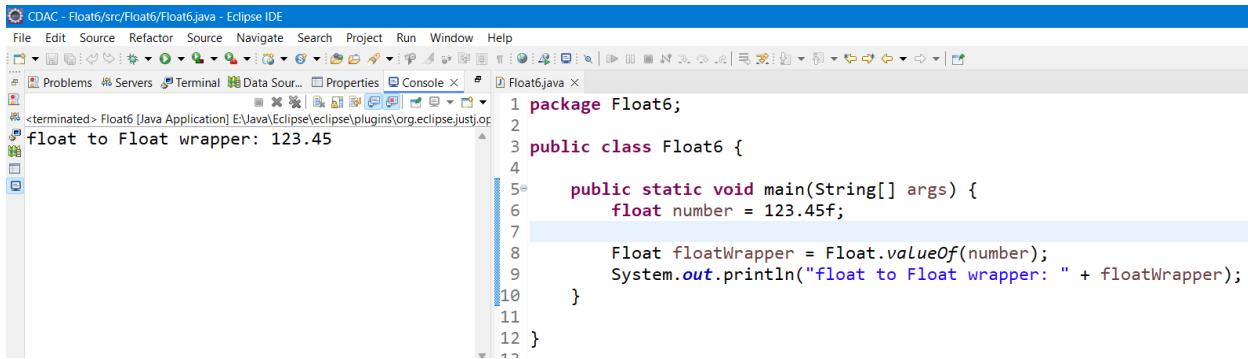


The screenshot shows the Eclipse IDE interface with the title bar "CDAC - Float6/src/Float6/Float6.java - Eclipse IDE". The code editor displays the following Java code:

```
1 package Float6;
2
3 public class Float6 {
4
5     public static void main(String[] args) {
6         String strNumber = "Ab12cd3";
7
8         try {
9
10             float number = Float.parseFloat(strNumber);
11             System.out.println("String to float: " + number);
12         } catch (NumberFormatException e) {
13             System.out.println("NumberFormatException: " + e.getMessage());
14         }
15     }
16
17 }
```

The line `float number = Float.parseFloat(strNumber);` is highlighted with a blue selection bar.

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)`).



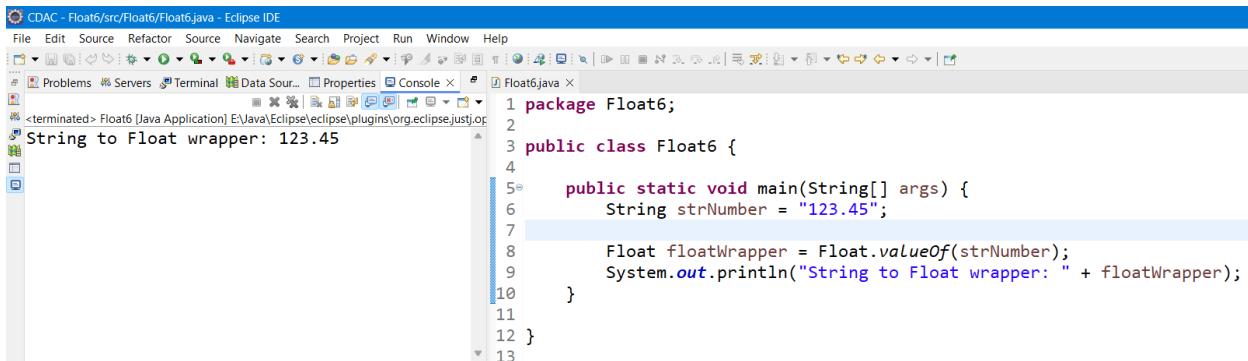
The screenshot shows the Eclipse IDE interface with the title bar "CDAC - Float6/src/Float6/Float6.java - Eclipse IDE". The code editor displays the following Java code:

```
1 package Float6;
2
3 public class Float6 {
4
5     public static void main(String[] args) {
6         float number = 123.45f;
7
8         Float floatWrapper = Float.valueOf(number);
9         System.out.println("float to Float wrapper: " + floatWrapper);
10    }
11
12 }
```

The line `Float floatWrapper = Float.valueOf(number);` is highlighted with a blue selection bar.

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()`.

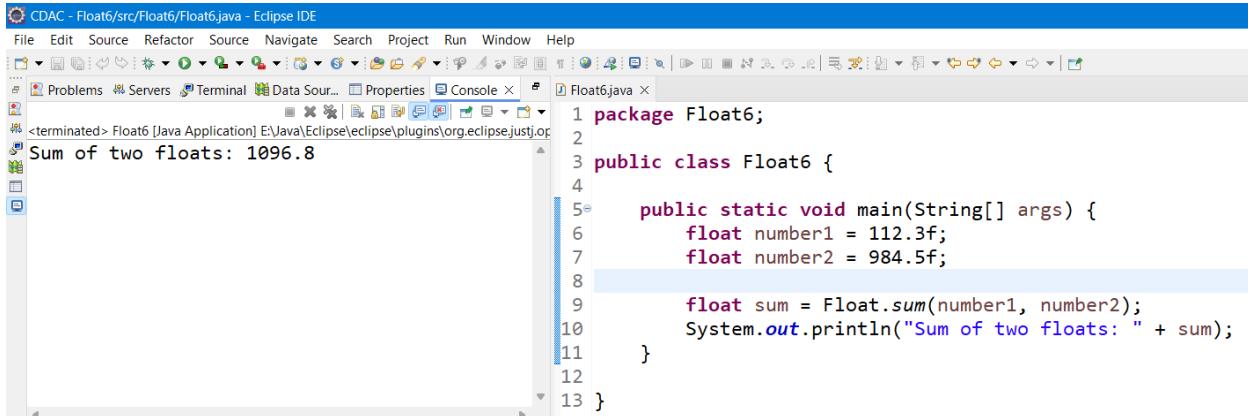
(Hint: Use `Float.valueOf(String)`).



Screenshot of the Eclipse IDE interface showing a Java code editor. The code converts a string to a float using `Float.valueOf(strNumber)`. The output in the console is "String to Float wrapper: 123.45".

```
CDAC - Float6/src/Float6/Float6.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Problems Servers Terminal Data Sour... Properties Console × Float6.java ×
<terminated> Float6 [Java Application] E:\Java\Eclipse\plugins\org.eclipse.justj.op
String to Float wrapper: 123.45
1 package Float6;
2
3 public class Float6 {
4
5     public static void main(String[] args) {
6         String strNumber = "123.45";
7
8         Float floatWrapper = Float.valueOf(strNumber);
9         System.out.println("String to Float wrapper: " + floatWrapper);
10    }
11
12 }
13
```

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)`).

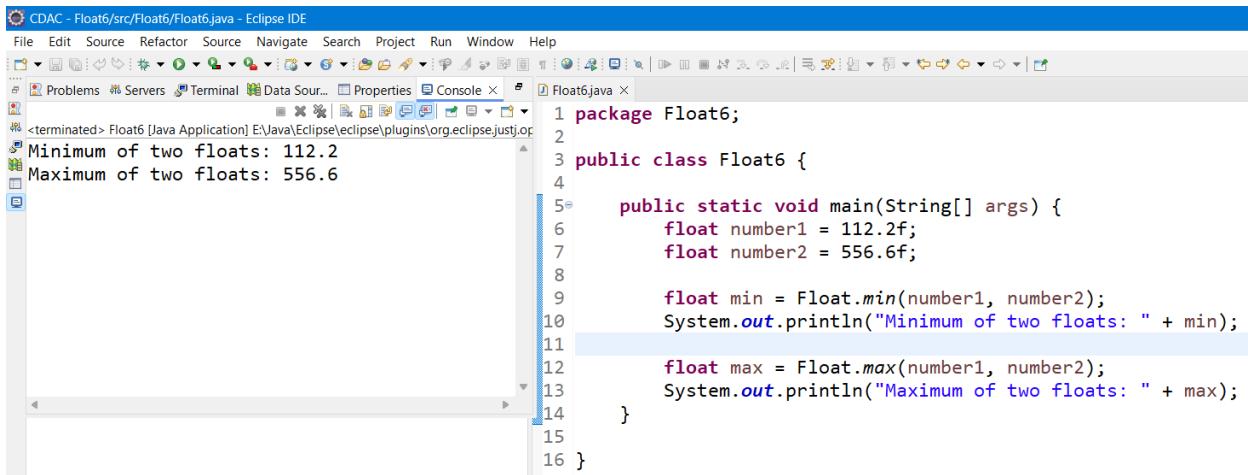


Screenshot of the Eclipse IDE interface showing a Java code editor. The code adds two float variables using `Float.sum(number1, number2)`. The output in the console is "Sum of two floats: 1096.8".

```
CDAC - Float6/src/Float6/Float6.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Problems Servers Terminal Data Sour... Properties Console × Float6.java ×
<terminated> Float6 [Java Application] E:\Java\Eclipse\plugins\org.eclipse.justj.op
Sum of two floats: 1096.8
1 package Float6;
2
3 public class Float6 {
4
5     public static void main(String[] args) {
6         float number1 = 112.3f;
7         float number2 = 984.5f;
8
9         float sum = Float.sum(number1, number2);
10        System.out.println("Sum of two floats: " + sum);
11    }
12
13 }
```

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

(float, float) and Float.max (float, float)).



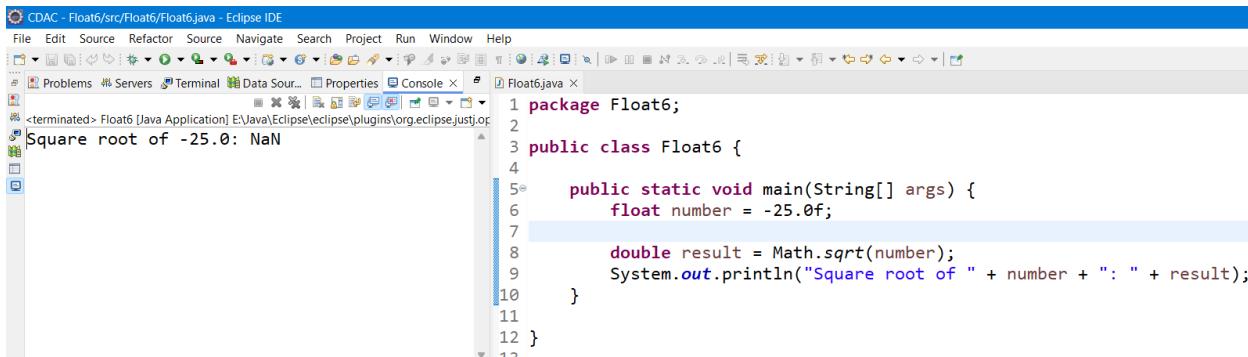
The screenshot shows the Eclipse IDE interface with the title bar "CDAC - Float6/src/Float6/Float6.java - Eclipse IDE". The code editor displays the following Java code:

```
1 package Float6;
2
3 public class Float6 {
4
5     public static void main(String[] args) {
6         float number1 = 112.2f;
7         float number2 = 556.6f;
8
9         float min = Float.min(number1, number2);
10        System.out.println("Minimum of two floats: " + min);
11
12        float max = Float.max(number1, number2);
13        System.out.println("Maximum of two floats: " + max);
14    }
15
16 }
```

The console output window shows the results of the program execution:

```
Minimum of two floats: 112.2
Maximum of two floats: 556.6
```

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



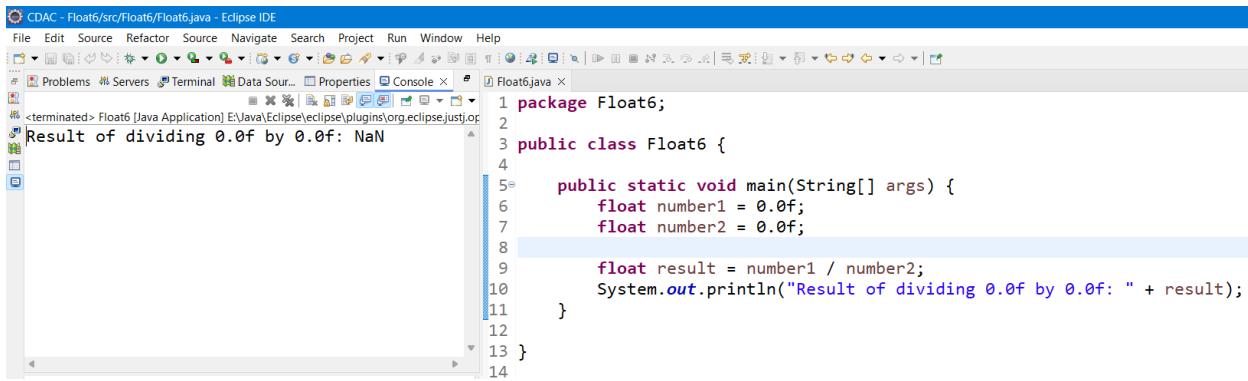
The screenshot shows the Eclipse IDE interface with the title bar "CDAC - Float6/src/Float6/Float6.java - Eclipse IDE". The code editor displays the following Java code:

```
1 package Float6;
2
3 public class Float6 {
4
5     public static void main(String[] args) {
6         float number = -25.0f;
7
8         double result = Math.sqrt(number);
9         System.out.println("Square root of " + number + ": " + result);
10    }
11
12 }
```

The console output window shows the results of the program execution:

```
Square root of -25.0: NaN
```

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



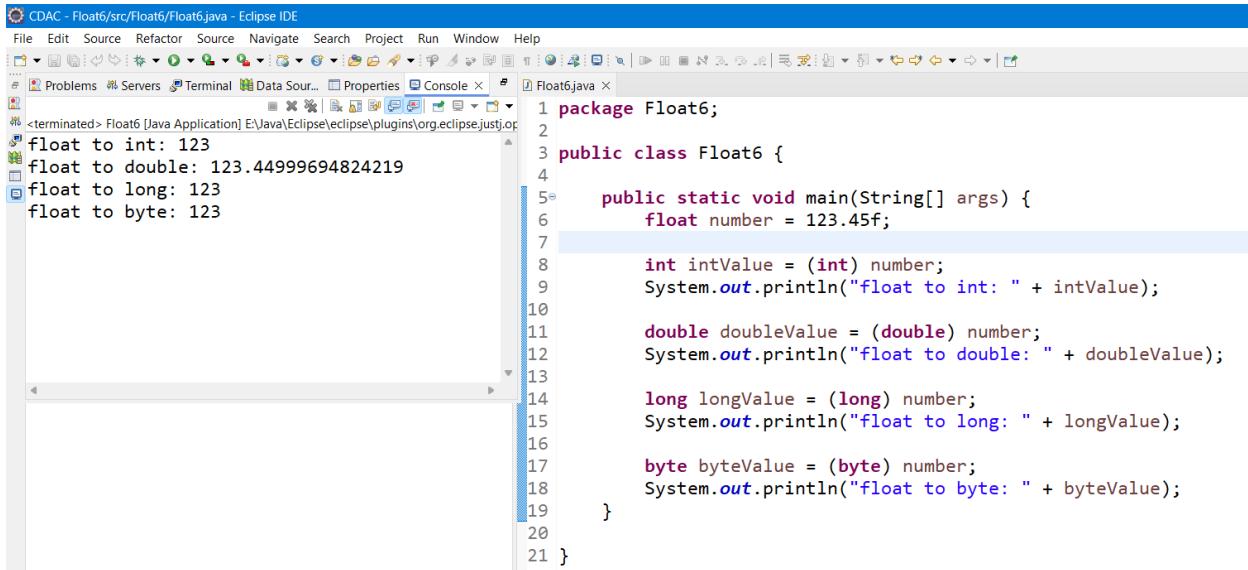
The screenshot shows the Eclipse IDE interface with the title bar "CDAC - Float6/src/Float6/Float6.java - Eclipse IDE". The code editor displays the following Java code:

```
1 package Float6;
2
3 public class Float6 {
4
5     public static void main(String[] args) {
6         float number1 = 0.0f;
7         float number2 = 0.0f;
8
9         float result = number1 / number2;
10        System.out.println("Result of dividing 0.0f by 0.0f: " + result);
11    }
12
13 }
```

The console output window shows the results of the program execution:

```
Result of dividing 0.0f by 0.0f: NaN
```

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



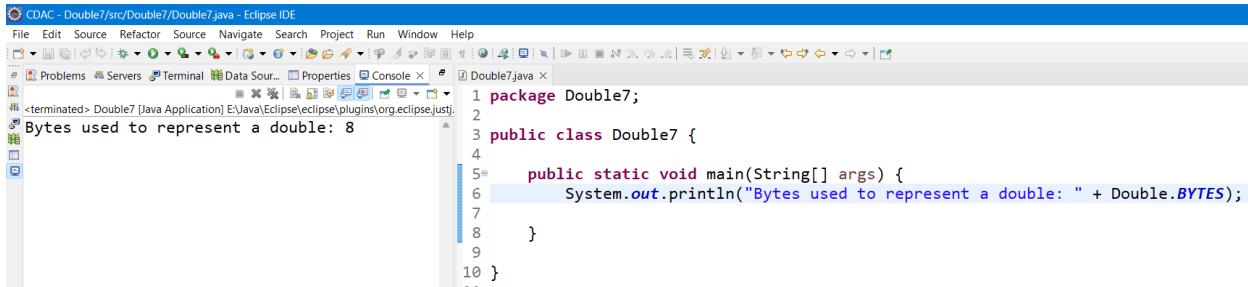
```
CDAC - Float6/src/Float6/Float6.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Problems Servers Terminal Data Sour... Properties Console X
<terminated> Float6 [Java Application] E:\Java\Eclipse\plugins\org.eclipse.jdt.core
float to int: 123
float to double: 123.44999694824219
float to long: 123
float to byte: 123

1 package Float6;
2
3 public class Float6 {
4
5     public static void main(String[] args) {
6         float number = 123.45f;
7
8         int intValue = (int) number;
9         System.out.println("float to int: " + intValue);
10
11         double doubleValue = (double) number;
12         System.out.println("float to double: " + doubleValue);
13
14         long longValue = (long) number;
15         System.out.println("float to long: " + longValue);
16
17         byte byteValue = (byte) number;
18         System.out.println("float to byte: " + byteValue);
19     }
20 }
21 }
```

7. Working with java.lang. Double

a. Explore the lava API documentation for java.lang. Double and observe its modifiers and super types. (Reading)

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

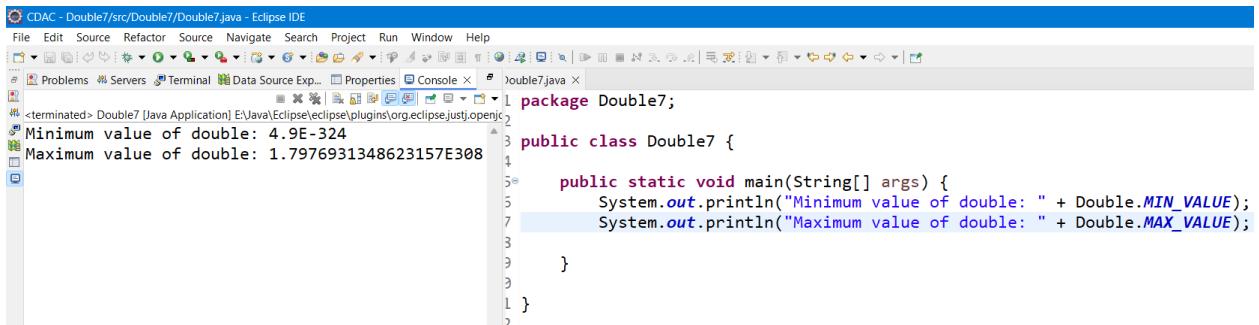


```
CDAC - Double7/src/Double7/Double7.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Problems Servers Terminal Data Sour... Properties Console X
<terminated> Double7 [Java Application] E:\Java\Eclipse\plugins\org.eclipse.jdt.core
Bytes used to represent a double: 8

1 package Double7;
2
3 public class Double7 {
4
5     public static void main(String[] args) {
6         System.out.println("Bytes used to represent a double: " + Double.BYTES);
7
8     }
9
10 }
```

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

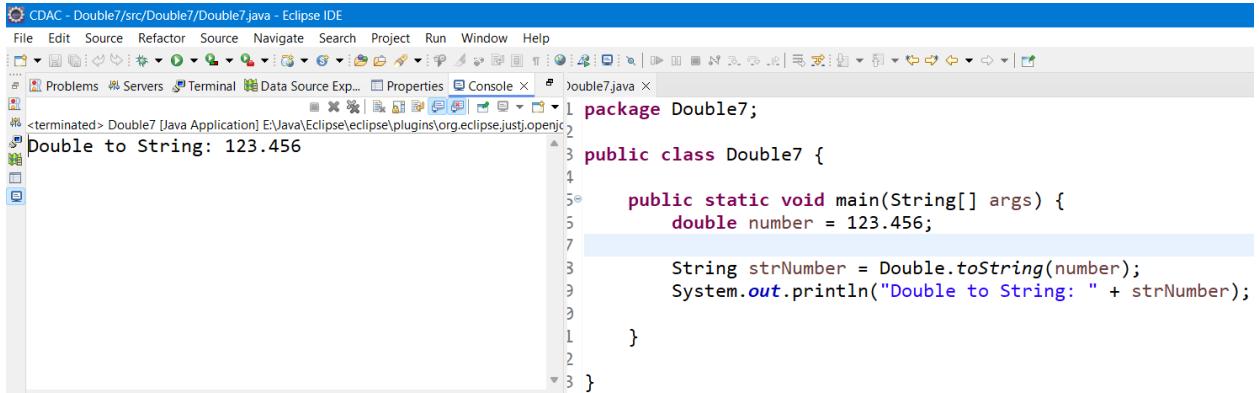
Double MAX VALUE).



```
CDAC - Double7/src/Double7/Double7.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Problems Servers Terminal Data Source Exp... Properties Console X Double7.java X
<terminated> Double7 [Java Application] E:\Java\Eclipse\eclipse\plugins\org.eclipse.jst.jdt.core\2
Minimum value of double: 4.9E-324
Maximum value of double: 1.7976931348623157E308
```

```
1 package Double7;
2
3 public class Double7 {
4
5     public static void main(String[] args) {
6         System.out.println("Minimum value of double: " + Double.MIN_VALUE);
7         System.out.println("Maximum value of double: " + Double.MAX_VALUE);
8     }
9 }
10 }
```

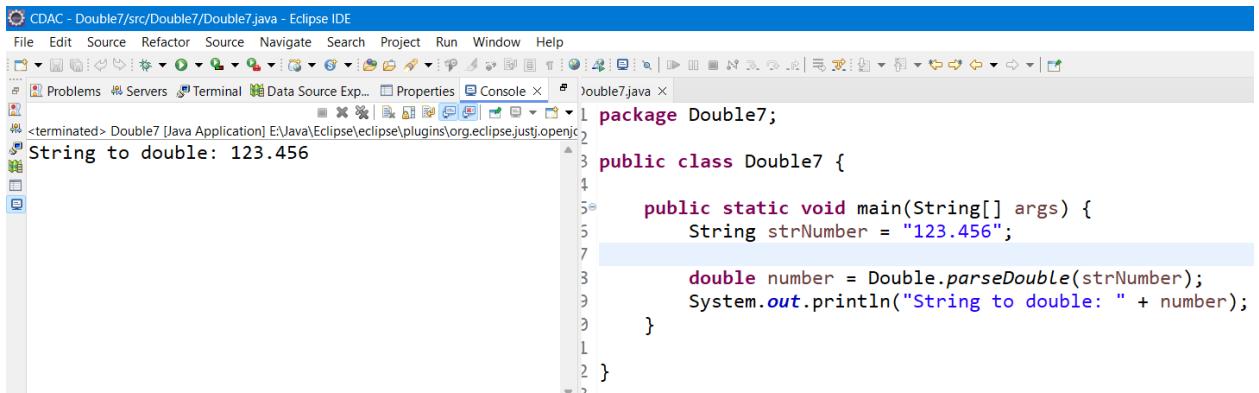
d. Declare a method-local variable number of type double with some value and convert it to a String using the `toString` method. (Hint: Use `Double.toString(double)`).



```
CDAC - Double7/src/Double7/Double7.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Problems Servers Terminal Data Source Exp... Properties Console X Double7.java X
<terminated> Double7 [Java Application] E:\Java\Eclipse\eclipse\plugins\org.eclipse.jst.jdt.core\2
double to String: 123.456
```

```
1 package Double7;
2
3 public class Double7 {
4
5     public static void main(String[] args) {
6         double number = 123.456;
7
8         String strNumber = Double.toString(number);
9         System.out.println("Double to String: " + strNumber);
10    }
11 }
```

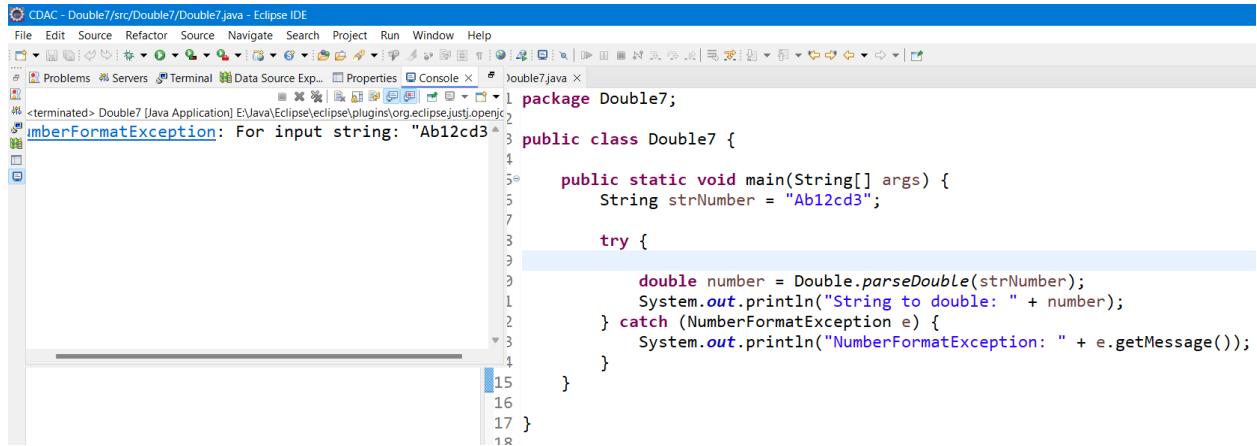
e. Declare a method-local variable strNumber of type string with some value and convert it to a double value using the `parseDouble` method. (Hint: Use `Double.parseDouble(String)`).



```
CDAC - Double7/src/Double7/Double7.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Problems Servers Terminal Data Source Exp... Properties Console X Double7.java X
<terminated> Double7 [Java Application] E:\Java\Eclipse\eclipse\plugins\org.eclipse.jst.jdt.core\2
String to double: 123.456
```

```
1 package Double7;
2
3 public class Double7 {
4
5     public static void main(String[] args) {
6         String strNumber = "123.456";
7
8         double number = Double.parseDouble(strNumber);
9         System.out.println("String to double: " + number);
10    }
11 }
```

f. Declare a method-local variable strNumber of type string with the value "Ab12cd3" and attempt to convert it to a double value. (Hint: parseDouble method will throw a NumberFormatException).

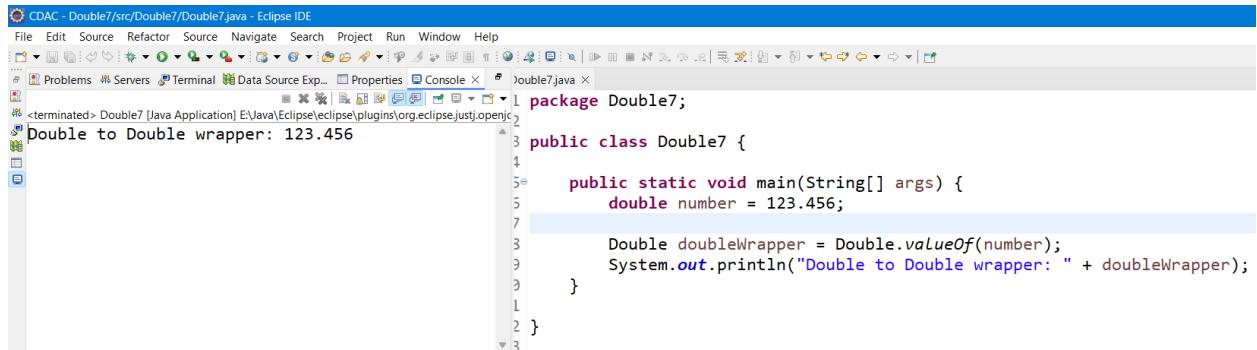


The screenshot shows the Eclipse IDE interface with the title bar "CDAC - Double7/src/Double7/Double7.java - Eclipse IDE". The code editor displays the following Java code:

```
package Double7;
public class Double7 {
    public static void main(String[] args) {
        String strNumber = "Ab12cd3";
        try {
            double number = Double.parseDouble(strNumber);
            System.out.println("String to double: " + number);
        } catch (NumberFormatException e) {
            System.out.println("NumberFormatException: " + e.getMessage());
        }
    }
}
```

A tooltip "NumberFormatException: For input string: \"Ab12cd3\"." is visible near the line where the exception is caught.

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)).

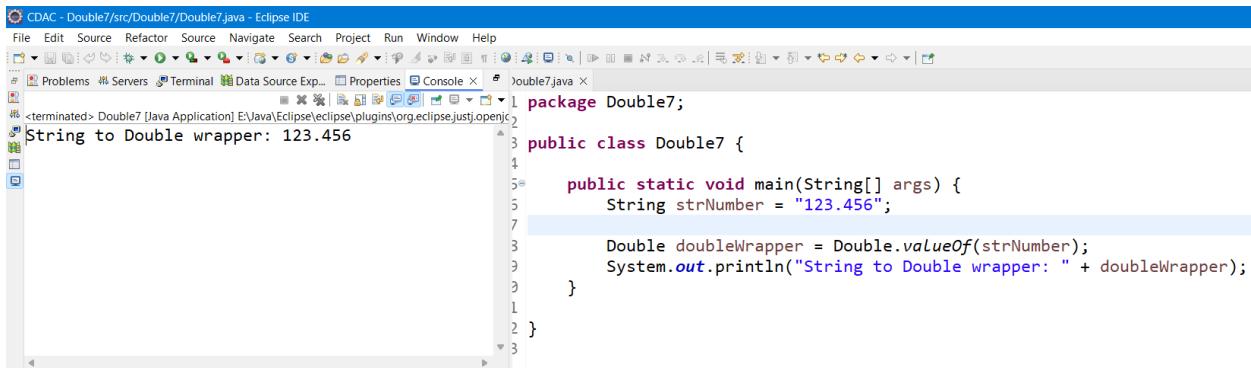


The screenshot shows the Eclipse IDE interface with the title bar "CDAC - Double7/src/Double7/Double7.java - Eclipse IDE". The code editor displays the following Java code:

```
package Double7;
public class Double7 {
    public static void main(String[] args) {
        double number = 123.456;
        Double doubleWrapper = Double.valueOf(number);
        System.out.println("Double to Double wrapper: " + doubleWrapper);
    }
}
```

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)).

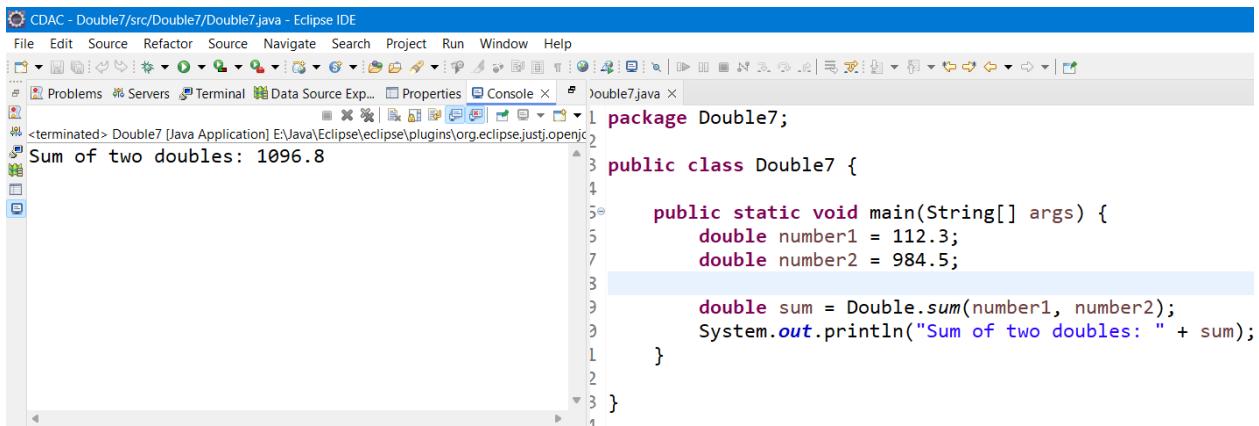


The screenshot shows the Eclipse IDE interface with the title bar "CDAC - Double7/src/Double7/Double7.java - Eclipse IDE". The code editor displays the following Java code:

```
1 package Double7;
2
3 public class Double7 {
4
5     public static void main(String[] args) {
6         String strNumber = "123.456";
7
8         Double doubleWrapper = Double.valueOf(strNumber);
9         System.out.println("String to Double wrapper: " + doubleWrapper);
10    }
11}
```

The line "Double doubleWrapper = Double.valueOf(strNumber);" is highlighted in blue, indicating it is selected or being edited.

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)).



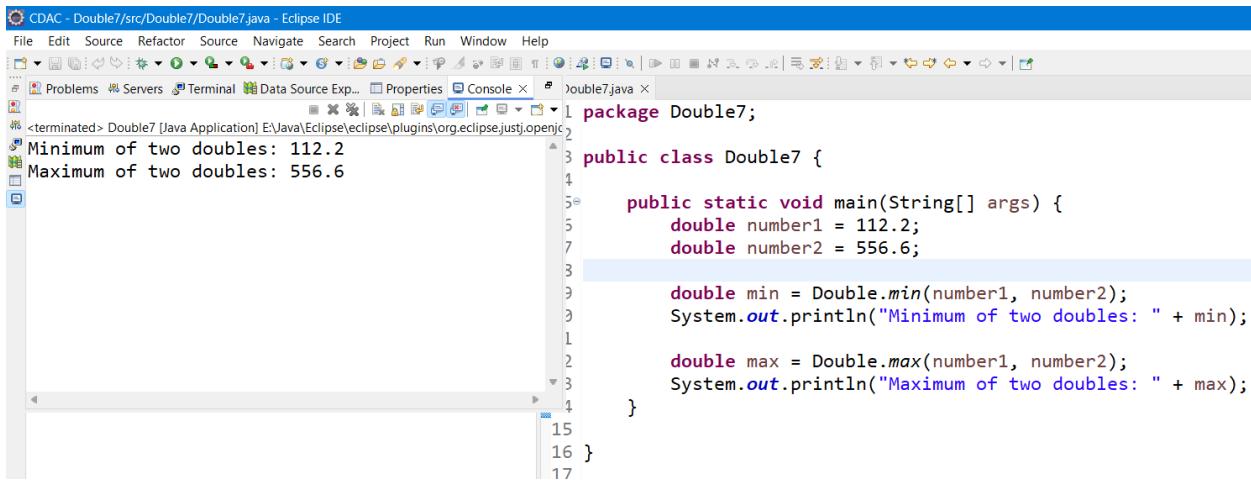
The screenshot shows the Eclipse IDE interface with the title bar "CDAC - Double7/src/Double7/Double7.java - Eclipse IDE". The code editor displays the following Java code:

```
1 package Double7;
2
3 public class Double7 {
4
5     public static void main(String[] args) {
6         double number1 = 112.3;
7         double number2 = 984.5;
8
9         double sum = Double.sum(number1, number2);
10        System.out.println("Sum of two doubles: " + sum);
11    }
12}
```

The line "double sum = Double.sum(number1, number2);" is highlighted in blue, indicating it is selected or being edited.

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)).



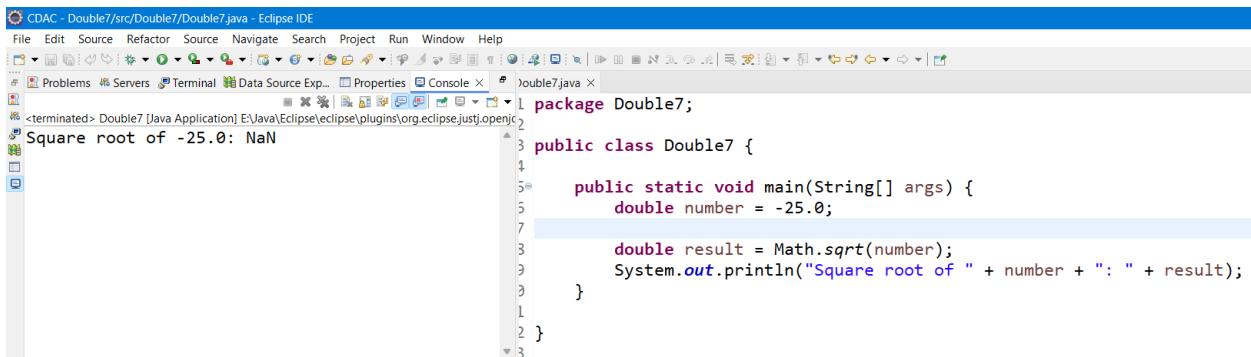
```
CDAC - Double7/src/Double7/Double7.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Problems Servers Terminal Data Source Exp... Properties Console <terminated> Double7 [Java Application] E:\Java\Eclipse\plugins\org.eclipse.justj.openj...
package Double7;
public class Double7 {
    public static void main(String[] args) {
        double number1 = 112.2;
        double number2 = 556.6;

        double min = Double.min(number1, number2);
        System.out.println("Minimum of two doubles: " + min);

        double max = Double.max(number1, number2);
        System.out.println("Maximum of two doubles: " + max);
    }
}
```

The screenshot shows the Eclipse IDE interface with the code for Double7.java. The code prints the minimum and maximum of two doubles, 112.2 and 556.6, respectively. The output window shows the results: "Minimum of two doubles: 112.2" and "Maximum of two doubles: 556.6".

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

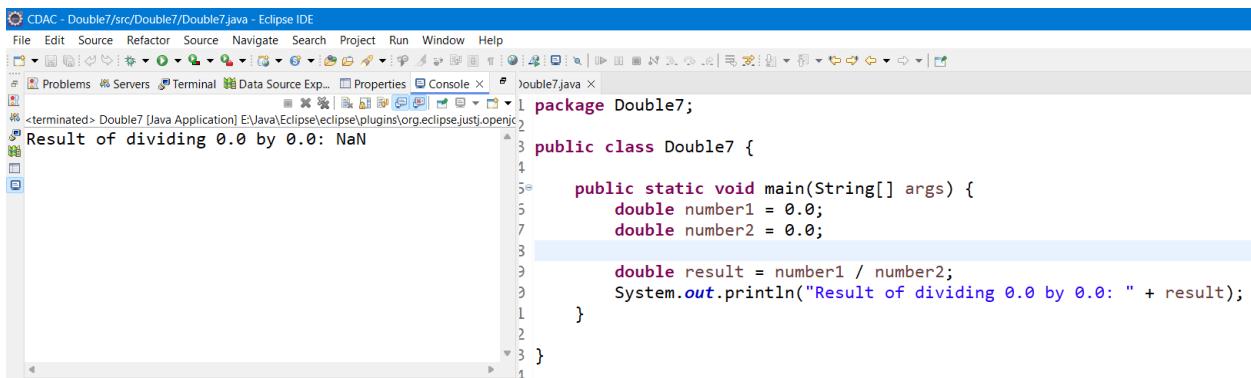


```
CDAC - Double7/src/Double7/Double7.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Problems Servers Terminal Data Source Exp... Properties Console <terminated> Double7 [Java Application] E:\Java\Eclipse\plugins\org.eclipse.justj.openj...
package Double7;
public class Double7 {
    public static void main(String[] args) {
        double number = -25.0;

        double result = Math.sqrt(number);
        System.out.println("Square root of " + number + ": " + result);
    }
}
```

The screenshot shows the Eclipse IDE interface with the code for Double7.java. It attempts to calculate the square root of -25.0, which results in NaN (Not a Number). The output window shows "Square root of -25.0: NaN".

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

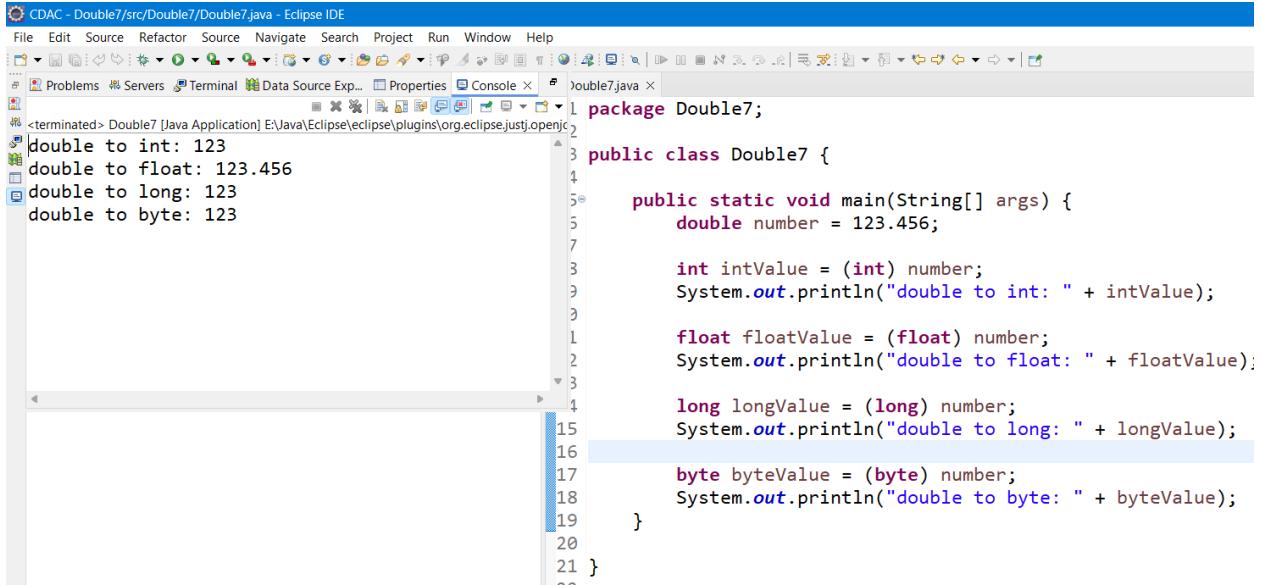


```
CDAC - Double7/src/Double7/Double7.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Problems Servers Terminal Data Source Exp... Properties Console <terminated> Double7 [Java Application] E:\Java\Eclipse\plugins\org.eclipse.justj.openj...
package Double7;
public class Double7 {
    public static void main(String[] args) {
        double number1 = 0.0;
        double number2 = 0.0;

        double result = number1 / number2;
        System.out.println("Result of dividing 0.0 by 0.0: " + result);
    }
}
```

The screenshot shows the Eclipse IDE interface with the code for Double7.java. It divides 0.0 by 0.0, resulting in NaN. The output window shows "Result of dividing 0.0 by 0.0: NaN".

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



The screenshot shows the Eclipse IDE interface. The title bar reads "CDAC - Double7/src/Double7/Double7.java - Eclipse IDE". The code editor displays the following Java code:

```
package Double7;
public class Double7 {
    public static void main(String[] args) {
        double number = 123.456;

        int intValue = (int) number;
        System.out.println("double to int: " + intValue);

        float floatValue = (float) number;
        System.out.println("double to float: " + floatValue);

        long longValue = (long) number;
        System.out.println("double to long: " + longValue);

        byte byteValue = (byte) number;
        System.out.println("double to byte: " + byteValue);
    }
}
```

Below the code editor, a terminal window shows the output of the program:

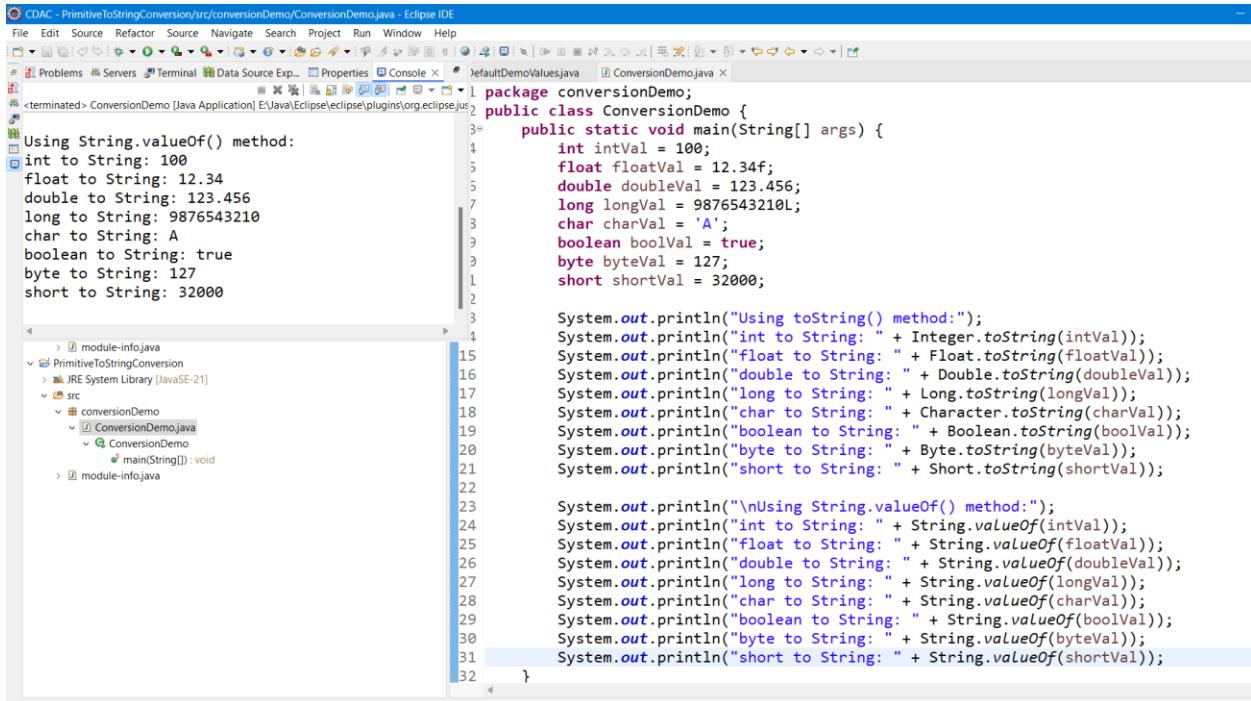
```
double to int: 123
double to float: 123.456
double to long: 123
double to byte: 123
```

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 value of method of the string class. (e.g., String.valueOf()).



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 org.example;
public class DefaultValuesDemo {

    // Instance variables (fields)
    int intValue;
    float floatVal;
    double doubleVal;
    long longVal;
    char charVal;
    boolean boolVal;
    byte byteVal;
    short shortVal;
```

```

static int staticIntVal;
static float staticFloatVal;
static double staticDoubleVal;
static long staticLongVal;
static char staticCharVal;
static boolean staticBoolVal;
static byte staticByteVal;
static short staticShortVal;

public static void main(String[] args) {

DefaultValuesDemo instance = new DefaultValuesDemo();

System.out.println("Default values of instance variables:");
System.out.println("int: " + instance.intValue);
System.out.println("float: " + instance.floatValue);
System.out.println("double: " + instance.doubleValue);
System.out.println("long: " + instance.longValue);
System.out.println("char: [" + instance.charValue + "]");
System.out.println("boolean: " + instance.boolValue);
System.out.println("byte: " + instance.byteValue);
System.out.println("short: " + instance.shortValue);

System.out.println("\nDefault values of static variables:");
System.out.println("static int: " + staticIntVal);
System.out.println("static float: " + staticFloatVal);
System.out.println("static double: " + staticDoubleVal);
System.out.println("static long: " + staticLongVal);
System.out.println("static char: [" + staticCharVal + "]");
System.out.println("static boolean: " + staticBoolVal);
System.out.println("static byte: " + staticByteVal);
System.out.println("static short: " + staticShortVal);
}
}

```

Output:

byte: 0

```
short: 0
```

```
Default values of static variables:
```

```
static int: 0
static float: 0.0
static double: 0.0
static long: 0
static char: [
```

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 commandArthimetic10;
public class CommandArthimetic10 {
public static void main(String[] args) {
if (args.length < 3) {
System.out.println("Please provide two integers and an operator (+, -, *, /).");
return;
}
try {
int num1 = Integer.parseInt(args[0]);
int num2 = Integer.parseInt(args[1]);
String operator = args[2];
int result = 0;
switch (operator) {
case "+":
result = num1 + num2;
break;
case "-":
result = num1 - num2;
break;
```

```
case "*":
result = num1 * num2;
break;
case "/":
if (num2 != 0) {
result = num1 / num2;
} else {
System.out.println("Division by zero is not allowed.");
return;
}
break;
default:
System.out.println("Invalid operator! Please use +, -, *, or /.");
return;
}
System.out.println("Result: " + result);
} catch (NumberFormatException e) {
System.out.println("Please enter valid integers.");
}
}
}
```

Output: Command-Line Input: 12 3 +

Result: 15