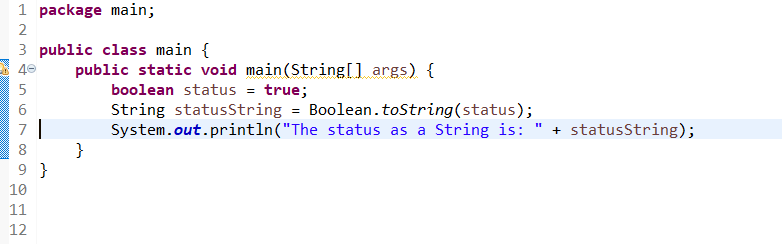
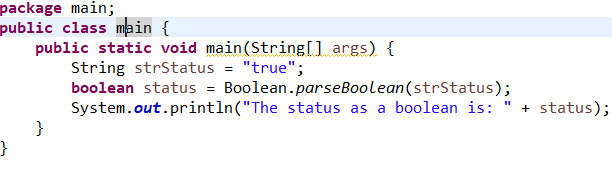
#### ****1. Working with**** java.lang.Boolean

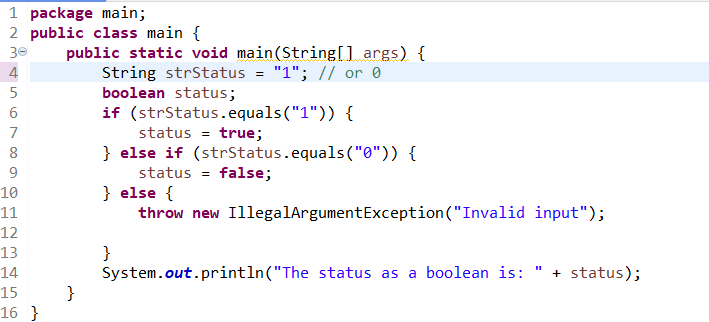
**b.** Declare a method-local variable status of type boolean with the value true and convert it to a String using the toString method. (Hint: Use Boolean.toString(Boolean) ).



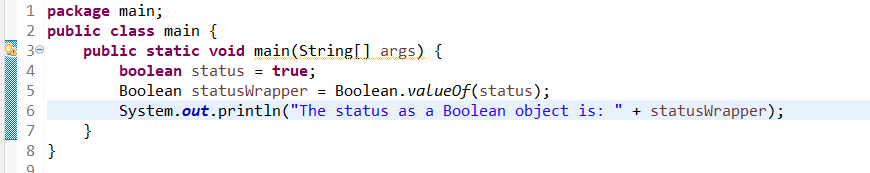
**c.** Declare a method-local variable strStatus of type String with the value "true" and convert it to a boolean using the parseBoolean method. (Hint: Use Boolean.parseBoolean(String)).



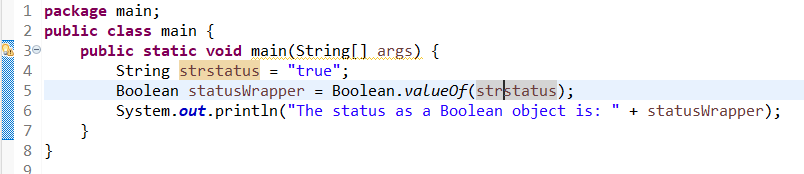
**d.** Declare a method-local variable strStatus of type String with the value "1" or "0" and attempt to convert it to a boolean. (Hint: parseBoolean method will not work as expected with "1" or "0").



**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)).



**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)).

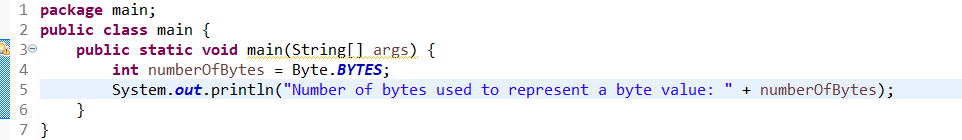


**g.** Experiment with converting a boolean value into other primitive types or vice versa and observe the results.

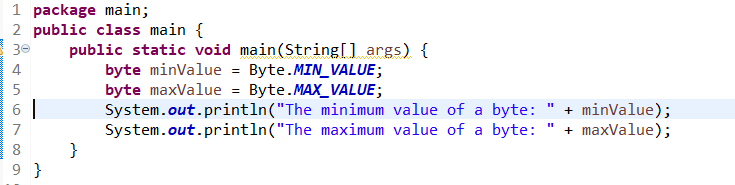
Ans: In Java, a boolean value cannot be directly converted into other primitive types like int, double, or char, and vice versa, because boolean is distinct from other numeric or character types. However, you can use indirect approaches, such as conditional expressions or parsing strings.

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

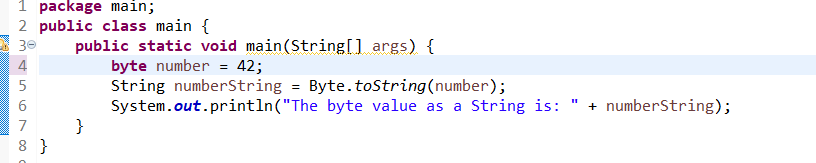
**b.** Write a program to test how many bytes are used to represent a byte value using the BYTES field. (Hint: Use Byte.BYTES).



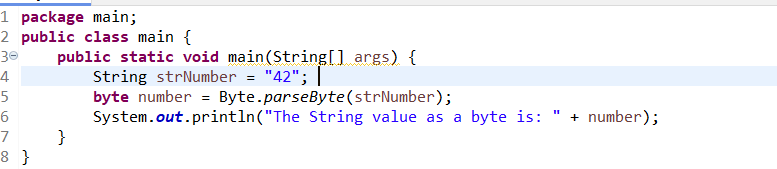
**c.** Write a program to find the minimum and maximum values of byte using the MIN\_VALUE and MAX\_VALUE fields. (Hint: Use Byte.MIN\_VALUE and Byte.MAX\_VALUE).



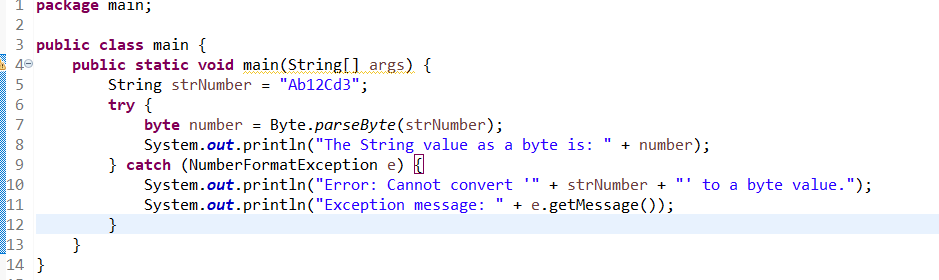
**d.** Declare a method-local variable number of type byte with some value and convert it to a String using the toString method. (Hint: Use Byte.toString(byte)).



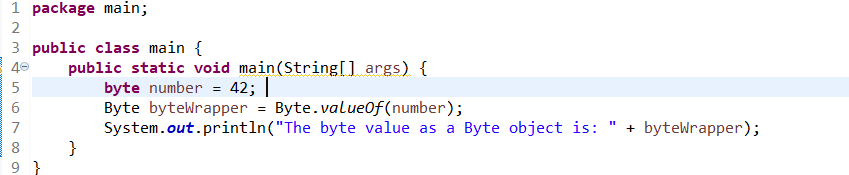
**e.** Declare a method-local variable strNumber of type String with some value and convert it to a byte value using the parseByte method. (Hint: Use Byte.parseByte(String)).



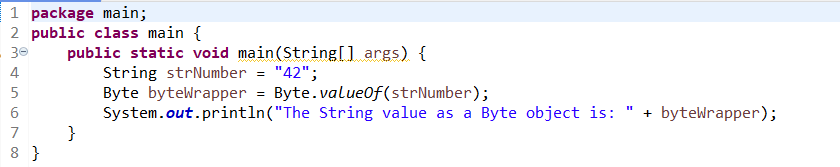
**f.** Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to a byte value. (Hint: parseByte method will throw a NumberFormatException).



**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)).



**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)).



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

In Java, a byte value can be converted into other primitive types such as int, short, long, float, double, and char. Similarly, other primitive types can be converted to byte, although caution must be taken to avoid data loss or overflow.

**3. Working with java.lang.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).**

**package javaprograms;**

**public class ShortOperation {**

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

**System.*out*.println(Short.*BYTES*);**

**}**

**}**

**Output – 2**

**short size is 2 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).**

**package javaprograms;**

**public class ShortOperation {**

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

**System.*out*.println("Max value of Short: "+ Short.*MAX\_VALUE*);**

**System.*out*.println("Min value of Short: "+ Short.*MIN\_VALUE*);**

**}**

**}**

**Output –**

**Max value of Short: 32767**

**Min value of Short: -32768**

**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)).**

**short number = 25675;**

**String number1 = Short.*toString*(number);**

**System.*out*.println(number1);**

**Output - 25675**

**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)).**

**String strNumber = "12345";**

**short strNumber1 = Short.*parseShort*(strNumber);**

**System.*out*.println(strNumber1);**

**Output - 12345**

**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).**

**String strNumber = "Ab12Cd3";**

**short strNumber1 = Short.*parseShort*(strNumber);**

**System.*out*.println(strNumber1);**

**Output –**

**Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3"**

**NumberFormatException exception occurs for String “Ab12Cd3” conversion to Short**

**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)).**

**short number= 25675;**

**Short number1 = Short.*valueOf*(number);**

**System.*out*.println(number1);**

**Output - 25675**

**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)).**

**String strNumber = "-25675";**

**Short strNumber1 = Short.*valueOf*(strNumber);**

**System.*out*.println(strNumber1);**

**Output : -25675**

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

**Short shortvalue = 2345;**

**int intvalue = shortvalue;**

**System.*out*.println("Conversion from short to Integer: "+intvalue);**

**short shortvalue1 = (short)intvalue;**

**System.*out*.println("Conversion from Integer to Short: "+intvalue);**

**Output :**

**Conversion from short to Integer: 2345**

**Conversion from Integer to Short: 2345**

**4. Working with java.lang.Integer**

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

**package javaprograms;**

**public class IntegerOperation {**

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

**System.*out*.println("Integer represents "+ Integer.*BYTES* + " bytes");**

**}**

**}**

**Output :**

**Integer represents 4 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).**

**package javaprograms;**

**public class IntegerOperation {**

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

**System.*out*.println("Max value of Integer: "+ Integer.*MAX\_VALUE*);**

**System.*out*.println("Min value of Integer: "+ Integer.*MIN\_VALUE*);**

**}**

**}**

**Output :**

**Max value of Integer: 2147483647**

**Min value of Integer: -2147483648**

**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)).**

**int nummber =12345678;**

**String number1 = Integer.*toString*(nummber);**

**System.*out*.println(number1);**

**Output : 12345678**

**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)).**

**String strNumber = "68683234";**

**int strNumber1 = Integer.*parseInt*(strNumber);**

**System.*out*.println(strNumber1);**

**Output : 68683234**

**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).**

**String strNumber = "Ab12Cd3";**

**int strNumber1 = Integer.*parseInt*(strNumber);**

**System.*out*.println(strNumber1);**

**Output :**

**Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3"**

**NumberFormatException exception occurs while converting String containing alphanumeric value.**

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

**int number = 87787887;**

**Integer number1 = Integer.*valueOf*(number);**

**System.*out*.println(number1);**

**Output : 87787887**

**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)).**

**String strNumber = "987654321";**

**Integer strNumber1 = Integer.*valueOf*(strNumber);**

**System.*out*.println(strNumber1);**

**Output : 987654321**

1. **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)).**

**int num1 = 10;**

**int num2 = 20;**

**System.*out*.println("Sum of num1 and num2 is "+ Integer.*sum*(num1, num2));**

**Output : Sum of num1 and num2 is 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)).**

**int num1 = 10;**

**int num2 = 20;**

**System.*out*.println("Minimum value = "+ Integer.*min*(num1, num2));**

**System.*out*.println("Maximum value = "+ Integer.*max*(num1, num2));**

**Output :**

**Minimum value = 10**

**Maximum value = 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)).**

**int intvalue = 7;**

**String binarystr = Integer.*toBinaryString*(intvalue);**

**System.*out*.println("Conversion to Binary Sting : "+ binarystr);**

**String octalstr = Integer.*toOctalString*(intvalue);**

**System.*out*.println("Conversion to Octal Sting : "+ octalstr);**

**String hexadecimalstr = Integer.*toHexString*(intvalue);**

**System.*out*.println("Conversion to HexaDecimal Sting : "+ hexadecimalstr);**

**Output :**

**Conversion to Binary Sting : 111**

**Conversion to Octal Sting : 7**

**Conversion to HexaDecimal Sting : 7**

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

**int intvalue = 80;**

**double doublevalue = intvalue;**

**System.*out*.println("Conversion from Intger to Double : "+ doublevalue);**

**double doublevalue1 = 7864.2343;**

**int intvalue1 = (int)doublevalue1;**

**System.*out*.println("Conversion from Double to Integer : "+ intvalue1);**

**int intvalue2 = 40;**

**short shortvalue = (short)intvalue2;**

**System.*out*.println("Conversion from Double to Integer : "+ shortvalue);**

**Output :**

**Conversion from Intger to Double : 80.0**

**Conversion from Double to Integer : 7864**

**Conversion from Double to Integer : 40**

#### 

#### 

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

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

#### package javaprograms;

#### public class LongOperation {

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

#### 

#### System.*out*.println(Long.*BYTES* + " bytes are used to represent Long value");

#### }

#### }

#### Output :

#### 8 bytes are used to represent Long value

#### 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).

#### package javaprograms;

#### public class LongOperation {

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

#### 

#### System.*out*.println("Max value of Long: "+Long.*MAX\_VALUE*);

#### System.*out*.println("Min value of Long: "+Long.*MIN\_VALUE*);

#### }

#### }

#### Output :

#### Max value of Long: 9223372036854775807

#### Min value of Long: -9223372036854775808

#### 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)).

#### long number = 2233445566l;

#### String str = Long.*toString*(number);

#### System.*out*.println(str);

#### Output : 2233445566

#### 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)).

#### String strNumber = "876687687";

#### long longnum = Long.*parseLong*(strNumber);

#### System.*out*.println(longnum);

#### Output : 876687687

#### 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).

#### String strNumber = "Ab12Cd3";

#### long longnum = Long.*parseLong*(strNumber);

#### System.*out*.println(longnum);

#### Output :

#### Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3" at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)

#### Conversion of nonnumber String to Long – NumberFormatException occurs.

#### 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)).

#### long number = 7877987987l;

#### Long number1 = Long.*valueOf*(number);

#### System.*out*.println(number1);

#### Output : 7877987987

#### 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)).

#### String strNumber = "799255766";

#### long longnum = Long.*valueOf*(strNumber);

#### System.*out*.println(longnum);

#### Output : 799255766

#### 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)).

#### long num1 = 1123;

#### long num2 = 9845;

#### long sum = Long.*sum*(num1, num2);

#### System.*out*.println("Sum of 2 long numbers is : "+ sum);

#### Output : Sum of 2 long numbers is : 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)).

#### long num1 = 1122;

#### long num2 = 5566;

#### long min = Long.*min*(num1, num2);

#### System.*out*.println("Minimum Long = "+ min);

#### long max = Long.*max*(num1, num2);

#### System.*out*.println("Maximum Long = "+ max);

#### Output :

#### Minimum Long = 1122

#### Maximum Long = 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)).

#### long longvalue = 7;

#### String binaryval = Long.*toBinaryString*(longvalue);

#### String octalval = Long.*toOctalString*(longvalue);

#### String hexadecival = Long.*toHexString*(longvalue);

#### 

#### System.*out*.println("Binary String: "+ binaryval );

#### System.*out*.println("Octal String: "+ octalval);

#### System.*out*.println("Hexadecimal string: "+hexadecival);

#### Output :

#### Binary String: 111

#### Octal String: 7

#### Hexadecimal string: 7

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

#### long longval = 25;

#### double doubleval = longval;

#### System.*out*.println("Double calue: " + doubleval);

#### 

#### long longval1 = (long) doubleval;

#### System.*out*.println("Long value: "+longval1);

#### 

#### long longval2 = 30;

#### int intval = (int)longval2;

#### System.*out*.println("Integer Value: "+ intval);

#### Output :

#### Double calue: 25.0

#### Long value: 25

#### Integer Value: 30

#### 6. Working with java.lang.Float

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

#### package javaprograms;

#### public class FloatOperation {

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

#### 

#### System.*out*.println("Byte representaion for Float is : "+ Float.*BYTES*);

#### 

#### }

#### }

#### Output :

#### Byte representaion for Float is : 4

#### 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).

#### System.*out*.println("Min value of Float: " + Float.*MIN\_VALUE*);

#### System.*out*.println("Max value of Float: " + Float.*MAX\_VALUE*);

#### Output :

#### Min value of Float: 1.4E-45

#### Max value of Float: 3.4028235E38

#### 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)).

#### float number = 235f;

#### String strnum = Float.*toString*(number);

#### System.*out*.println("Float = "+number);

#### System.*out*.println("String = "+strnum);

#### Output :

#### Float = 235.0

#### String = 235.0

#### 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)).

#### String strNumber = "455";

#### float floatno = Float.*parseFloat*(strNumber);

#### System.*out*.println("Float value = "+floatno);

#### Output: Float value = 455.0

#### 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).

#### String strNumber = "Ab12Cd3";

#### float floatno = Float.*parseFloat*(strNumber);

#### System.*out*.println("Float value = "+floatno);

#### Output :

#### NumberFormatException exception occurs as the String is not in proper format to convert into float value.

#### 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)).

#### float number = 250f;

#### Float floatval = Float.*valueOf*(number);

#### System.*out*.println(floatval);

#### Output : 250.0

#### 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)).

#### String strNumber = "234567";

#### float floatno = Float.*valueOf*(strNumber);

#### System.*out*.println("Float value = "+floatno);

#### Output : Float value = 234567.0

#### 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)).

#### float num1 = 112.3f;

#### float num2 = 984.5f;

#### 

#### float sum = Float.*sum*(num1, num2);

#### System.*out*.println("Sum of float = "+ sum);

#### Output : Sum of float = 1096.8

#### 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)).

#### float num1 = 112.2f;

#### float num2 = 556.6f;

#### 

#### System.*out*.println("Max value = "+ Float.*max*(num1, num2));

#### System.*out*.println("Min value = "+ Float.*min*(num1, num2));

#### Output :

#### Max value = 556.6

#### Min value = 112.2

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

#### float val = -25.0f;

#### float sqrtval = (float) Math.*sqrt*(val);

#### System.*out*.println("Square root of -25.0f = "+ sqrtval);

#### Output :

#### Square root of -25.0f = NaN

#### As of the negative float value , square root of negative values are not defined in the real number system

#### float val = -25.0f;

#### float sqrtval = (float) Math.*sqrt*(val);

#### System.*out*.println("Square root of -25.0f = "+ sqrtval);

#### 

#### //trying to negative value to positive value and calculating the squareroot

#### if(val < 0) {

#### float sqrtval1 = (float) Math.*sqrt*(-val);

#### System.*out*.println("Squareroot = "+ sqrtval1);

#### }

#### Output :

#### Square root of -25.0f = NaN

#### Squareroot = 5.0

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

#### float n1 = 0.0f;

#### float n2 = 0.0f;

#### float result = n1/n2;

#### System.*out*.println("result of division: " +result);

#### Output :

#### result of division: NaN

#### NaN represents Not a Number , undefined or unpredictable result in floating point calculation.

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

#### float fval = 255.0f;

#### double dval = fval;

#### System.*out*.println("Double Value = "+dval);

#### 

#### int ival = (int) fval;

#### System.*out*.println("Integer value = "+ival);

#### 

#### int ival1 = 20;

#### float fval1 = ival;

#### System.*out*.println("Float value = "+fval1);

#### 

#### double dval2 = 5040.055;

#### float fval2 = (float) dval2;

#### System.*out*.println("Float Value = "+fval2);

#### Output :

#### Double Value = 255.0

#### Integer value = 255

#### Float value = 255.0

#### Float Value = 5040.055

#### 7. Working with java.lang.Double

#### a. Explore the [Java API documentation for java.lang.Double](https://docs.oracle.com/javase/8/docs/api/java/lang/Double.html) 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).

#### package javaprograms;

#### public class DoubleOperation {

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

#### 

#### System.*out*.println("Byte representation for Double value= "+ Double.*BYTES*);

#### }

#### }

#### Output : Byte representation for Double value= 8

#### 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).

#### package javaprograms;

#### public class DoubleOperation {

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

#### 

#### System.*out*.println("Max value of Double : "+ Double.*MAX\_VALUE*);

#### System.*out*.println("Min value of Double : "+ Double.*MIN\_VALUE*);

#### }

#### }

#### Output :

#### Max value of Double : 1.7976931348623157E308

#### Min value of Double : 4.9E-324

#### 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)).

#### double number = 797789.82;

#### String str = Double.*toString*(number);

#### System.*out*.println(str);

#### Output : 797789.82

#### 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)).

#### String strNumber = "9876.2345";

#### double strnumber1 = Double.*parseDouble*(strNumber);

#### System.*out*.println(strnumber1);

#### Output : 9876.2345

#### 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).

#### String strNumber = "Ab12Cd3";

#### double strnumber1 = Double.*parseDouble*(strNumber);

#### System.*out*.println(strnumber1);

#### Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3"

#### 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)).

#### double number = 52525.09;

#### Double number1 = Double.*valueOf*(number);

#### System.*out*.println(number1);

#### Output : 52525.09

#### 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)).

#### String strNumber = "9595.123";

#### double strnumber1 = Double.*valueOf*(strNumber);

#### System.*out*.println(strnumber1);

#### Output : 9595.123

#### 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)).

#### double n1 = 112.3;

#### double n2 = 984.5;

#### double sum = Double.*sum*(n1, n2);

#### System.*out*.println("Sum of Double = "+sum);

#### Output : Sum of Double = 1096.8

#### 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)).

#### double n1 = 112.2;

#### double n2 = 556.6;

#### double min = Double.*min*(n1, n2);

#### double max = Double.*max*(n1, n2);

#### System.*out*.println("Min = "+min);

#### System.*out*.println("Max = "+max);

#### Output :

#### Min = 112.2

#### Max = 556.6

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

#### double val = -25.0;

#### double sqrtval = (double) Math.*sqrt*(val);

#### System.*out*.println("Square root of -25.0 = "+sqrtval);

#### 

#### if(val < 0) {

#### double sqrtval1 = (double) Math.*sqrt*(-val);

#### System.*out*.println("Square root of -25.0 = "+sqrtval1);

#### }

#### Output :

#### Square root of -25.0 = NaN

#### Square root of -25.0 = 5.0

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

#### double n1 = 0.0;

#### double n2 = 0.0;

#### double div = n1 / n2;

#### System.*out*.println("Division = "+div);

#### Output : Division = NaN { NaN represents Not a Number}

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

#### double dnum = 2456.09;

#### int inum = (int) dnum;

#### System.*out*.println("Int num = "+ inum);

#### float fnum = (float) dnum;

#### System.*out*.println("Float num = "+ fnum);

#### long lnum = (long) dnum;

#### System.*out*.println("Long num = "+ lnum);

#### double idnum = inum;

#### System.*out*.println("Integer to Double num = "+ idnum);

#### double fdnum = fnum;

#### System.*out*.println("Float to Double num = "+ fdnum);

#### double ldnum = lnum;

#### System.*out*.println("Long to Double num = "+ ldnum);

#### Output :

#### Int num = 2456

#### Float num = 2456.09

#### Long num = 2456

#### Integer to Double num = 2456.0

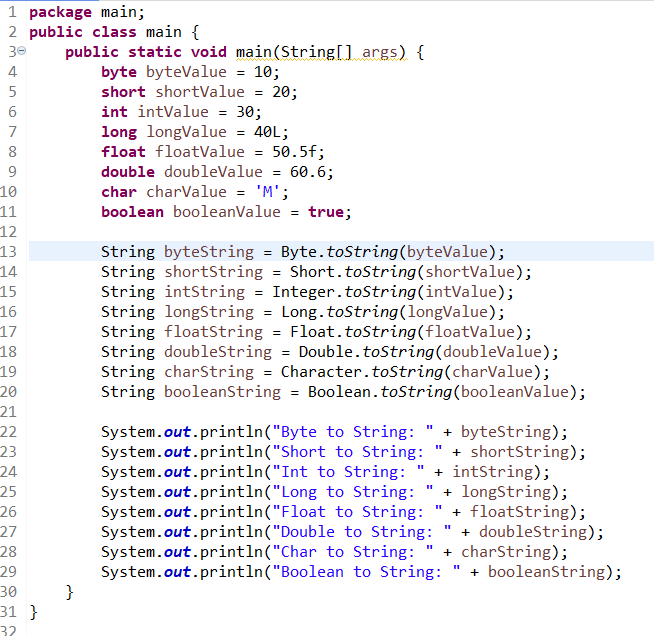
#### Float to Double num = 2456.090087890625

#### Long to Double num = 2456.0

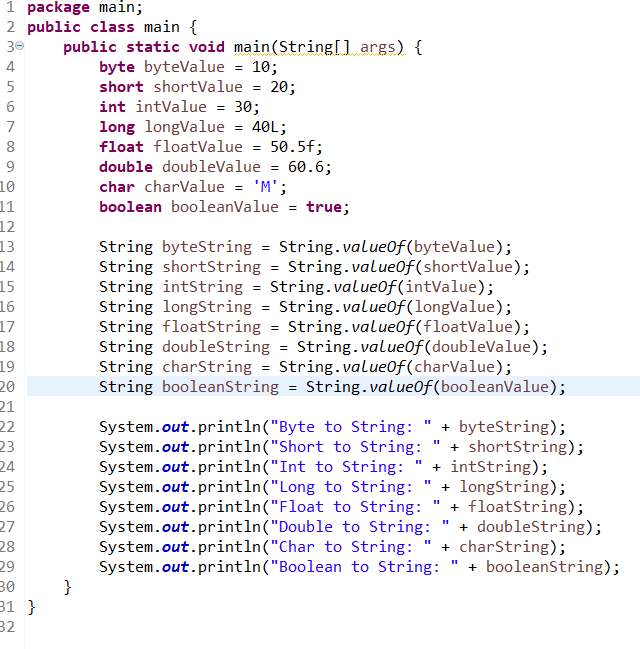
#### ****8. Conversion between Primitive Types and Strings****

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()).

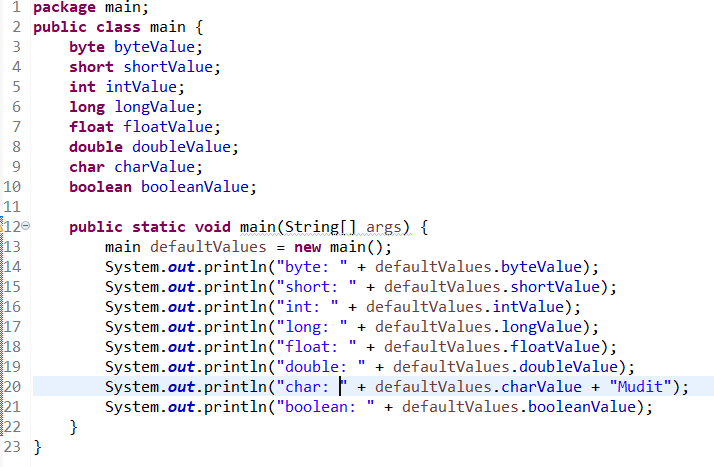


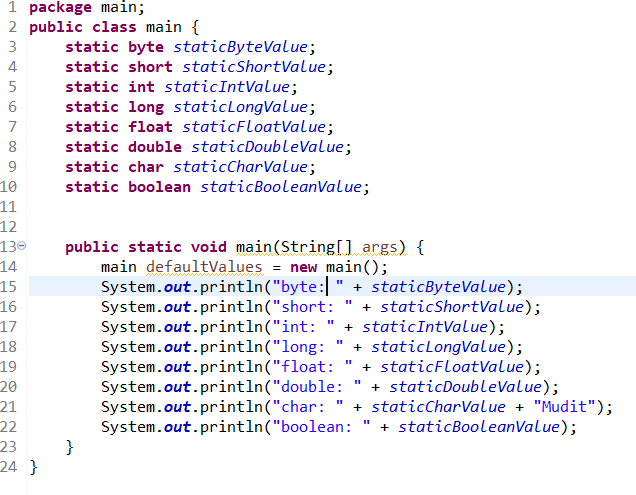
* + Then, use the valueOf method of the String class. (e.g., String.valueOf()).



#### ****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).





#### ****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).

