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

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

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

**a.** Explore the [Java API documentation for java.lang.Boolean](https://docs.oracle.com/javase/8/docs/api/java/lang/Boolean.html) and observe its modifiers and super types.

**public final class Boolean extends** [**Object**](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html)

**The access modifier for Boolean is public**

**While the non-access modifier is final that mean we can’t create child class for the Boolean class**

**The super type of the Boolean class is Object**

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

public class BooleanToString {

public static void main(String[] args) {

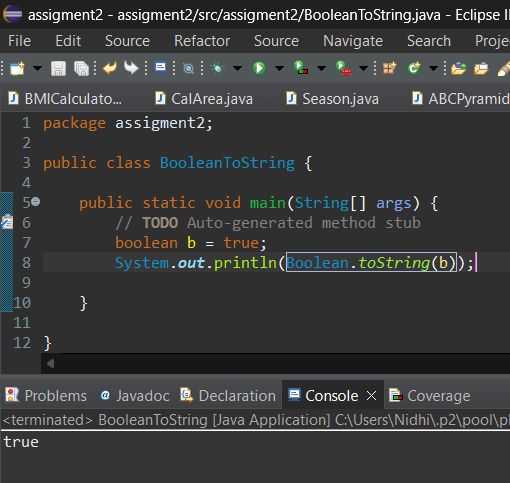
// **TODO** Auto-generated method stub

boolean b = true;

System.***out***.println(Boolean.*toString*(b));

}

}



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

public class StringtoBoolean {

public static void main(String[] args) {

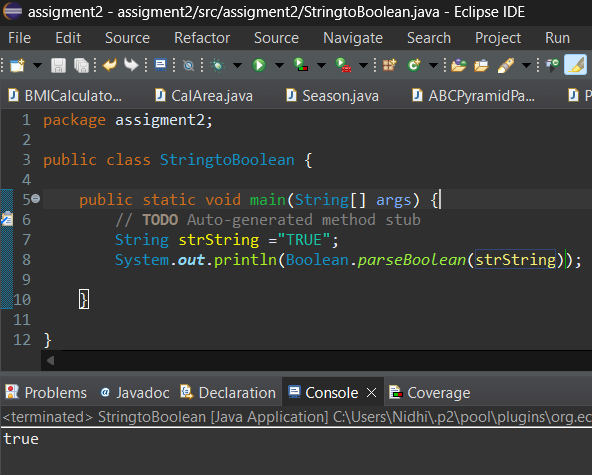
// **TODO** Auto-generated method stub

String strString ="TRUE";

System.***out***.println(Boolean.*parseBoolean*(strString));

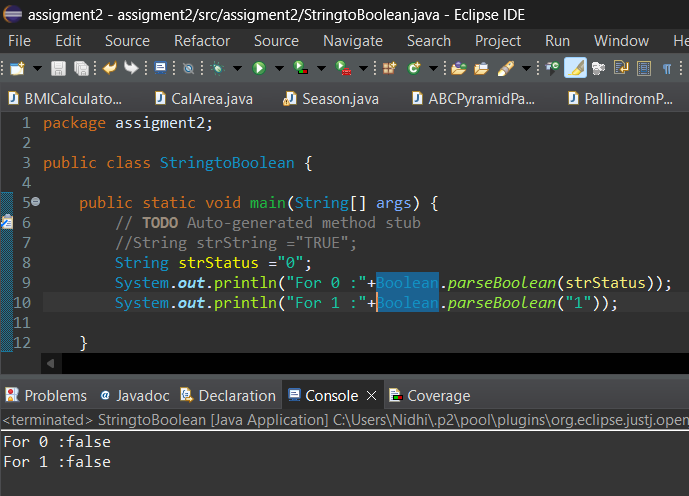
}

}



**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").

**Yes complied & executed both at CT and RT successfully But in java the 0 and 1 are not represented as false true , they are just numerical values hence using parse Boolean method we get false for both 0 and 1**



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

public class BooleanToString {

public static void main(String[] args) {

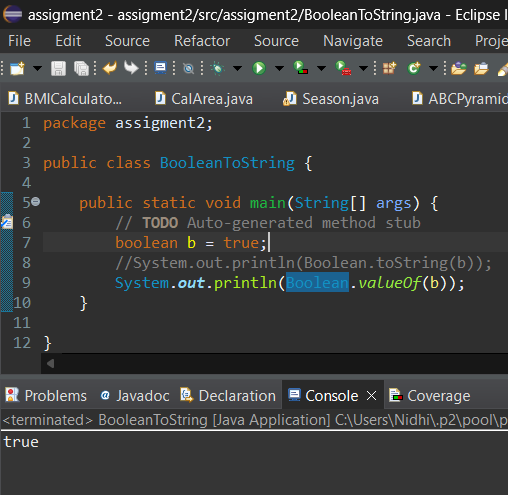
// **TODO** Auto-generated method stub

boolean b = true;

System.***out***.println(Boolean.*valueOf*(b));

}

}



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

public class StringtoBoolean {

public static void main(String[] args) {

// **TODO** Auto-generated method stub

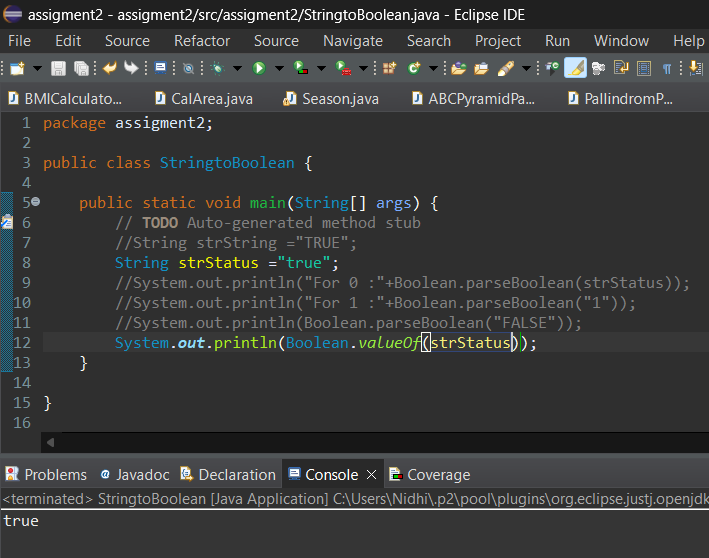
//String strString ="TRUE";

String strStatus ="true";

System.***out***.println(Boolean.*valueOf*(strStatus));

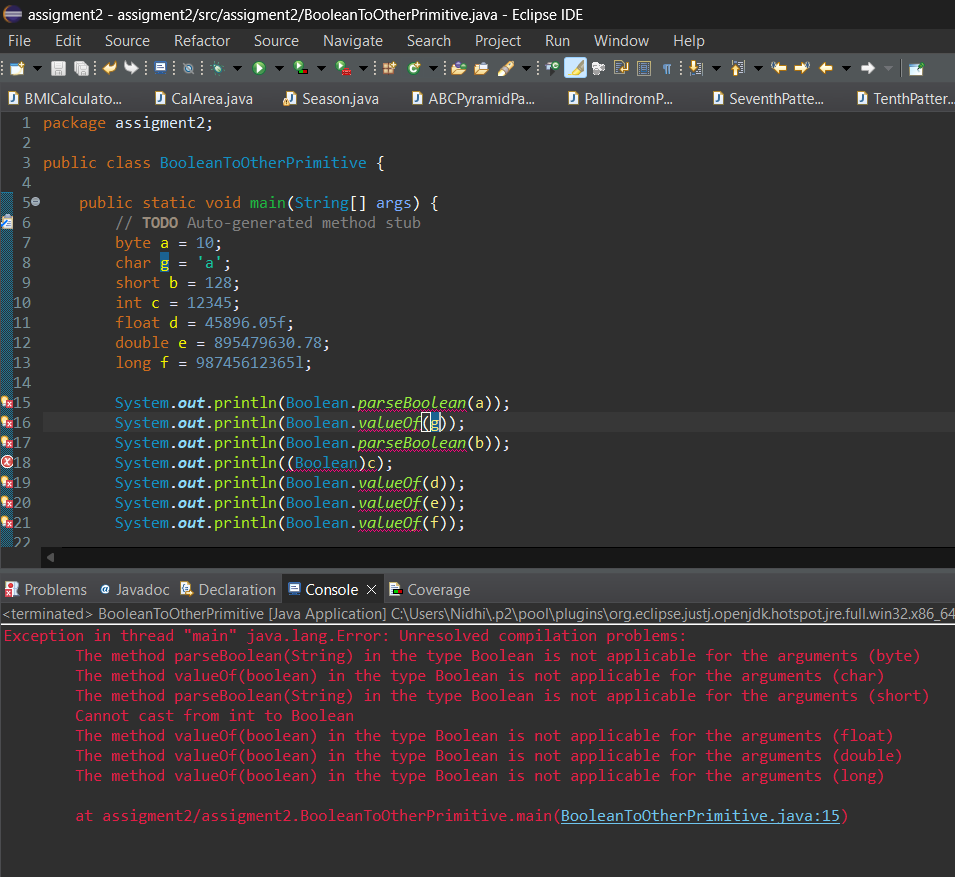
}

}

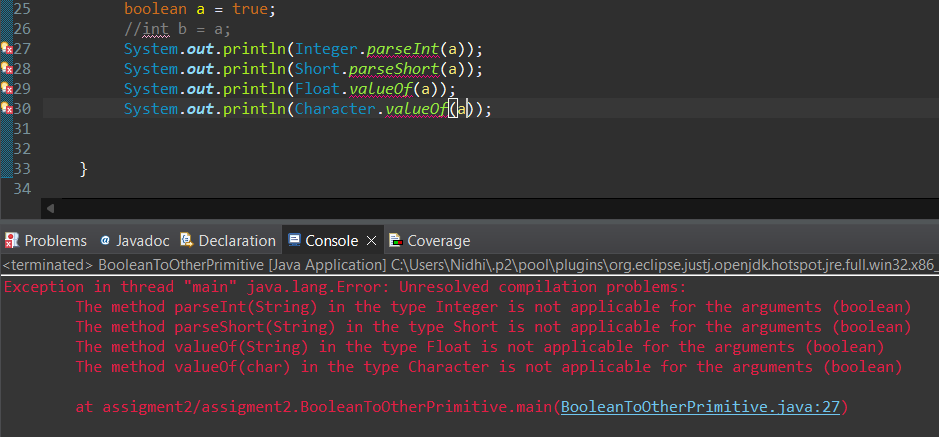


**g.** Experiment with converting a boolean value into other primitive types or vice versa and observe the results.

**From other primitive DT to Boolean or vice vera we get CTE**



Boolean to other primitive DT



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

**a.** Explore the [Java API documentation for java.lang.Byte](https://docs.oracle.com/javase/8/docs/api/java/lang/Byte.html) and observe its modifiers and super types.

public final class **Byte** extends [Number](https://docs.oracle.com/javase/8/docs/api/java/lang/Number.html)

**The access modifier for Byte** **is public**

**While the non-access modifier is final that mean we can’t create child class for the Boolean class**

The Byte class super type is Number , and the Number class super type is Object

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

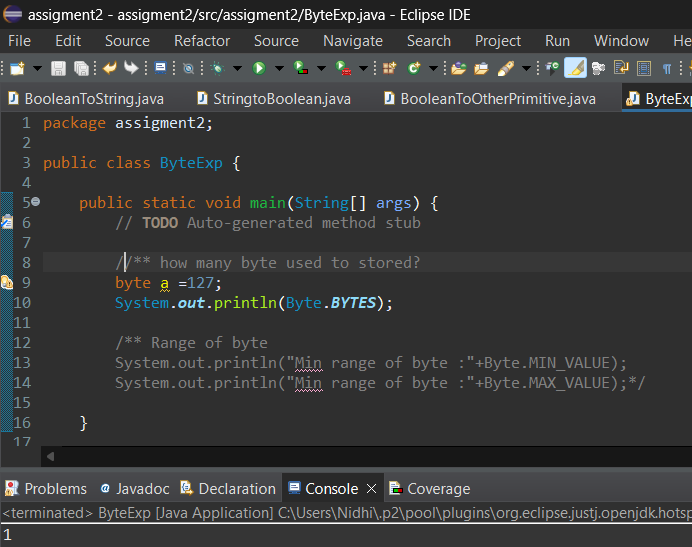
public class ByteExp {

public static void main(String[] args) {

System.out.println(Byte.BYTES);

}

}



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

public class ByteExp {

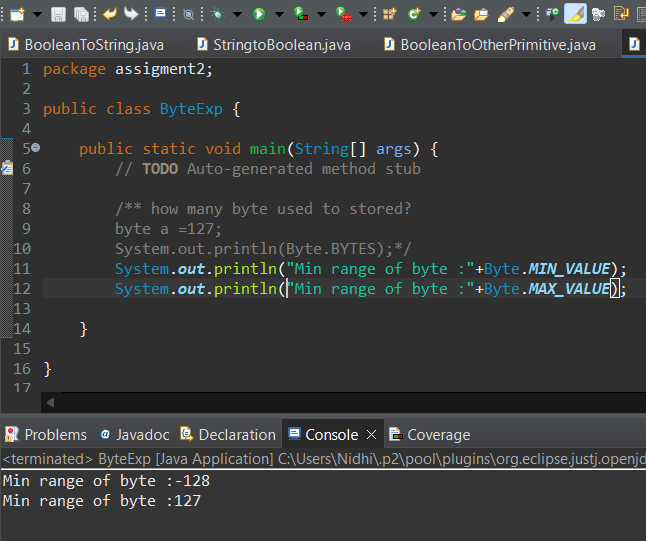
public static void main(String[] args) {

System.out.println("Min range of byte :"+Byte.MIN\_VALUE);

System.out.println("Min range of byte :"+Byte.MAX\_VALUE);

}

}



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

public class ByteExp {

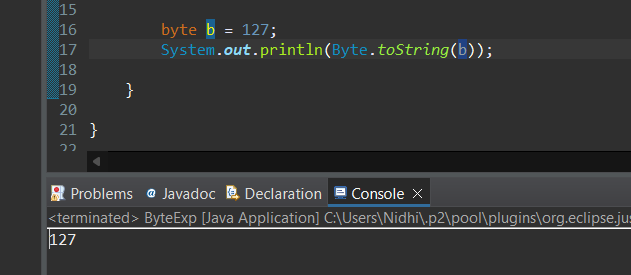
public static void main(String[] args) {

byte b = 127;

System.out.println(Byte.toString(b));

}

}



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

public class ByteExp {

public static void main(String[] args) {

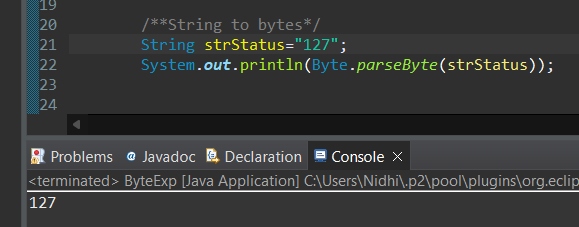
/\*\*String to bytes\*/

String strStatus="127";

System.out.println(Byte.parseByte(strStatus));

}

}



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

public class ByteExp {

public static void main(String[] args) {

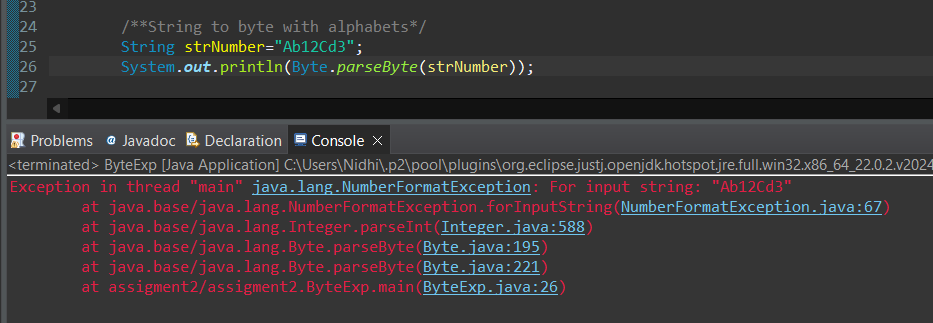
String strNumber="Ab12Cd3";

System.out.println(Byte.parseByte(strNumber));

}

}

Only pure number stored in string variable are converted to Byte, no alphanumeric value converted to short DT.



**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 ByteExp {

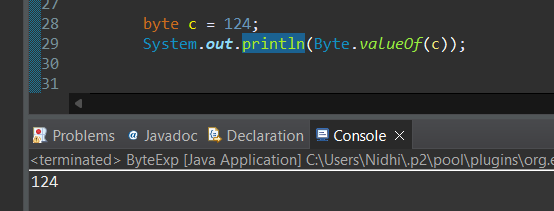
public static void main(String[] args) {

byte c = 124;

System.out.println(Byte.valueOf(c));

}

}



**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 ByteExp {

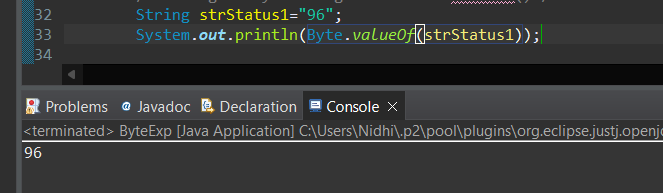
public static void main(String[] args) {

String strStatus1="96";

System.out.println(Byte.valueOf(strStatus1));

}

}



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

Byte to other primitive DT

public class ByteExp {

public static void main(String[] args) {

byte d =101;

System.out.println("Int "+Integer.valueOf(d));

System.out.println("Short "+Short.valueOf(d));

System.out.println("Charater " + (char)d);

System.out.println("float "+Float.valueOf(d));

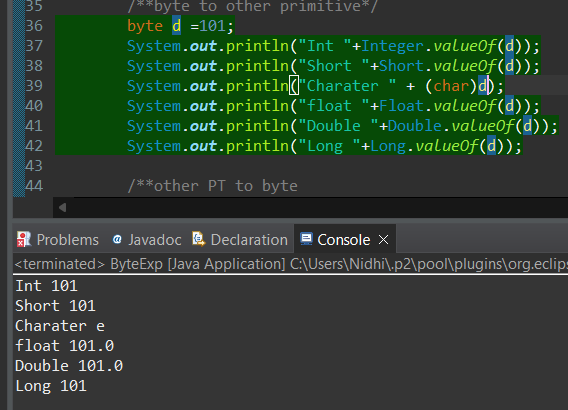
System.out.println("Double "+Double.valueOf(d));

System.out.println("Long "+Long.valueOf(d));

}

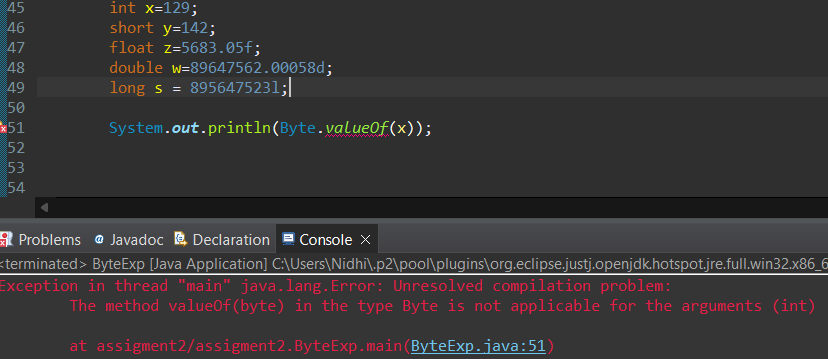
}

We can conver



Other to byte

We can’t use the valueOF function to covert other DT to byre DT  
because the Byte.valueOF(byte) only take byte DT , hence we will do narrowing here(external typecasting)



public class ByteExp {

public static void main(String[] args) {

int x=129;

short y=142;

float z=5683.05f;

double w=89647562.00058d;

long s = 895647523l;

System.out.println("Int to bye"+(byte)x);

System.out.println("Short to bye"+(byte)y);

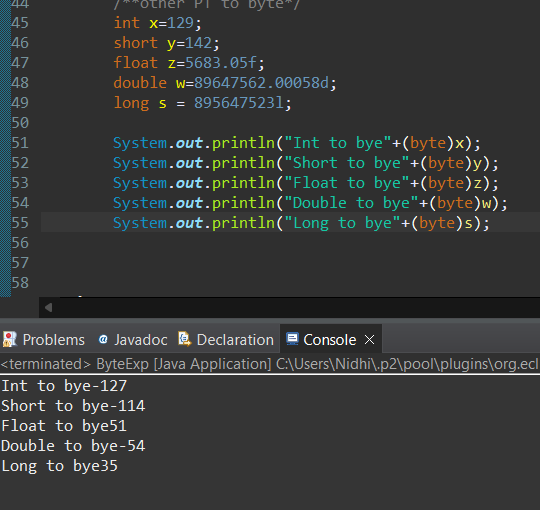
System.out.println("Float to bye"+(byte)z);

System.out.println("Double to bye"+(byte)w);

System.out.println("Long to bye"+(byte)s);

}

}



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

**a.** Explore the [Java API documentation for java.lang.Short](https://docs.oracle.com/javase/8/docs/api/java/lang/Short.html) and observe its modifiers and super types.

public final class **Short** extends [Number](https://docs.oracle.com/javase/8/docs/api/java/lang/Number.html)

**The access modifier for Short is public**

**While the non-access modifier is final that mean we can’t create child class for the Short class**

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

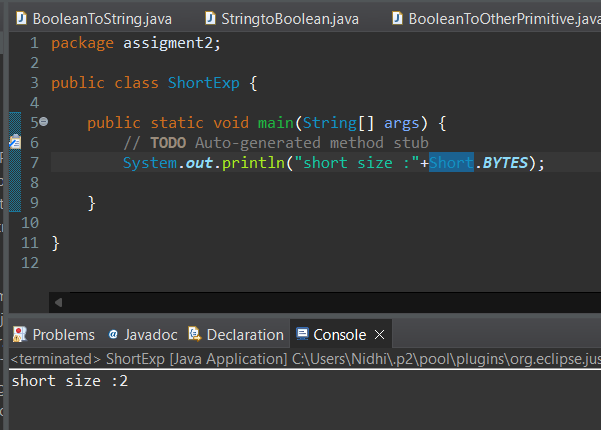
public class ShortExp {

public static void main(String[] args) {

System.out.println("short size :"+Short.BYTES);

}

}



The Byte class super type is Number , and the Number class super type is Object

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

public class ShortExp {

public static void main(String[] args) {

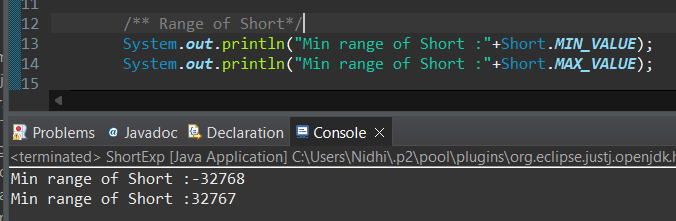
/\*\* Range of Short\*/

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

System.out.println("Min range of Short :"+Short.MAX\_VALUE);

}

}



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

public class ShortExp {

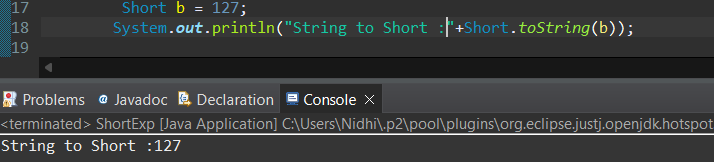
public static void main(String[] args) {

Short b = 127;

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

}

}



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

public class ShortExp {

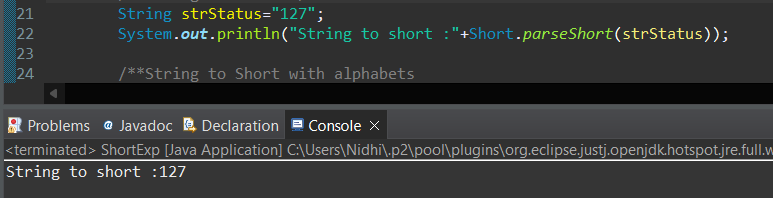
public static void main(String[] args) {

String strStatus="127";

System.out.println("String to short :"+Short.parseShort(strStatus));

}

}



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

public class ShortExp {

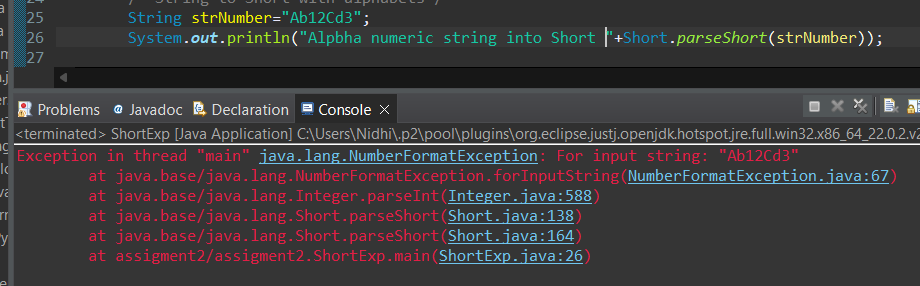
public static void main(String[] args) {

String strNumber="Ab12Cd3";

System.out.println("Alpbha numeric string into Short "+Short.parseShort(strNumber));

}

}



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

public class ShortExp {

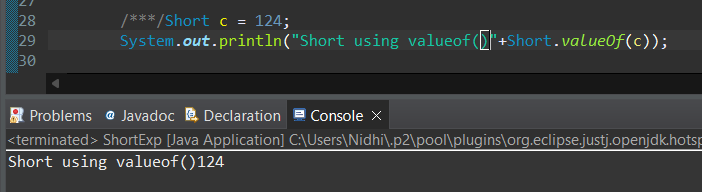
public static void main(String[] args) {

Short c = 124;

System.out.println("Short using valueof()"+Short.valueOf(c));

}

}



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

public class ShortExp {

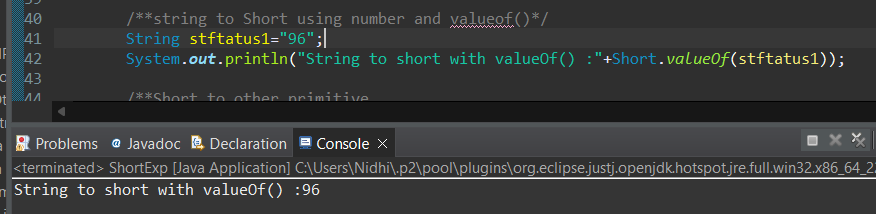
public static void main(String[] args) {

String stftatus1="96";

System.out.println("String to short with valueOf() :"+Short.valueOf(stftatus1));

}

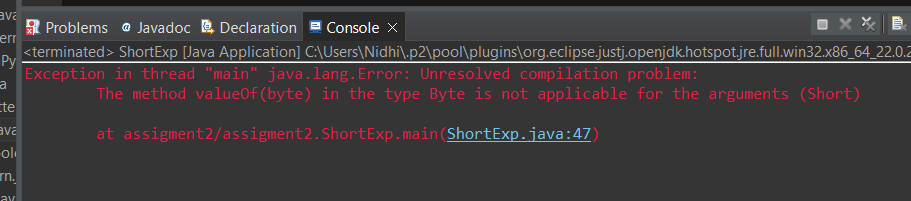
}

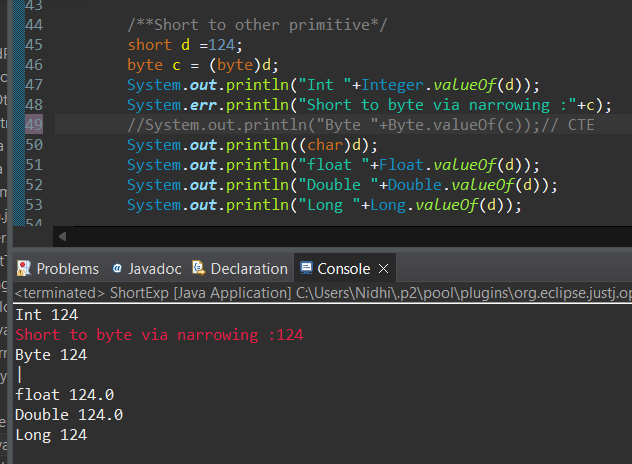


1. Experiment with converting a short value into other primitive types or vice versa and observe the results.

System.***out***.println("Byte "+Byte.*valueOf*(d)); giving error

short d =124;

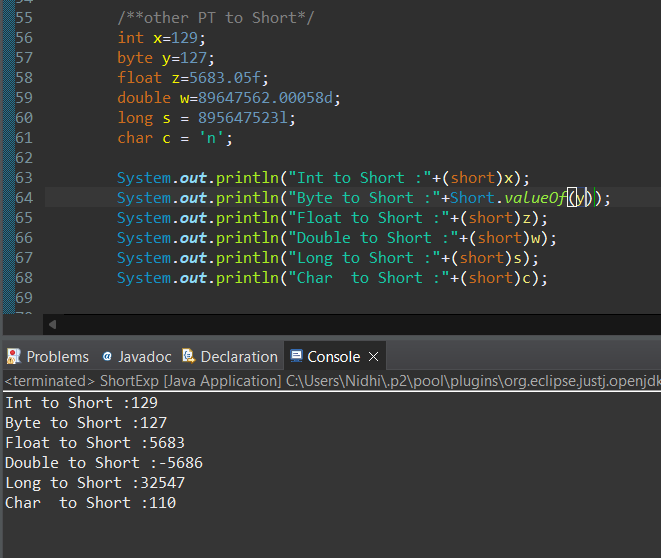
if we type cast then the casting pass both CT and RT



Other to Short DT

We can’t use valueOf() to concert higher DT to short DT hence mostly we user implicit casting narrowing .

For converting



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

**a.** Explore the [Java API documentation for java.lang.Integer](https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html) and observe its modifiers and super types.

public final class **Byte** extends [Number](https://docs.oracle.com/javase/8/docs/api/java/lang/Number.html)

**The access modifier for Boolean is public**

**While the non-access modifier is final that mean we can’t create child class for the Boolean class**

The Byte class super type is Number , and the Number class super type is Object

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

public class IntegerExp {

public static void main(String[] args) {

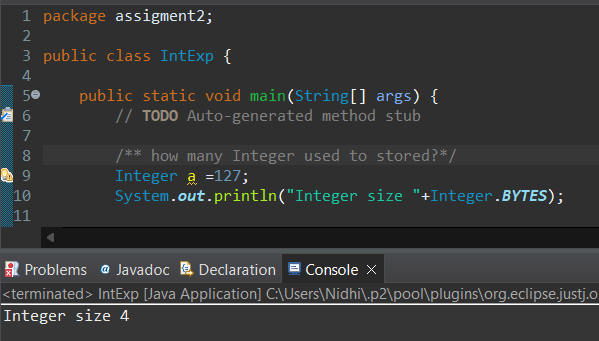
/\*\* how many Integer used to stored?\*/

Integer a =127;

System.out.println("Integer size "+Integer.BYTES);

}

}



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

public class IntegerExp {

public static void main(String[] args) {

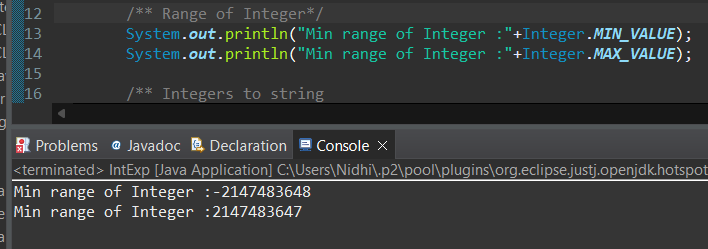
/\*\* Range of Integer\*/

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

System.out.println("Min range of Integer :"+Integer.MAX\_VALUE);

}

}



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

public class IntegerExp {

public static void main(String[] args) {

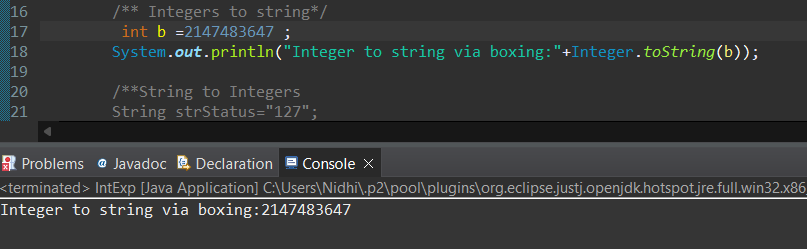
/\*\* Integers to string\*/

int b =2147483647 ;

System.out.println("Integer to string via boxing:"+Integer.toString(b));

}

}



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

public class IntegerExp {

public static void main(String[] args) {

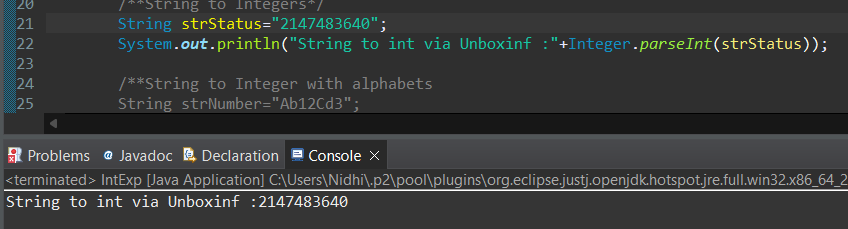
/\*\*String to Integers\*/

String strStatus="2147483640";

System.out.println("String to int via Unboxinf :"+Integer.parseInt(strStatus));

}

}



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

public class IntegerExp {

public static void main(String[] args) {

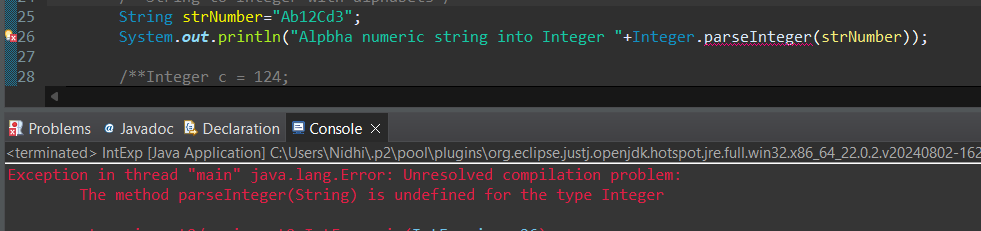
/\*\*String to Integer with alphabets\*/

String strNumber="Ab12Cd3";

System.out.println("Alpbha numeric string into Integer "+Integer.parseInteger(strNumber));

}

}



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

public class IntegerExp {

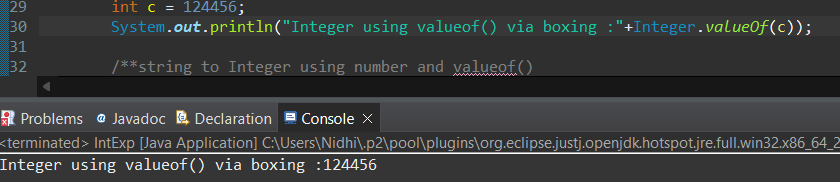
public static void main(String[] args) {

int c = 124456;

System.out.println("Integer using valueof() via boxing :"+Integer.valueOf(c));

}

}



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

public class IntegerExp {

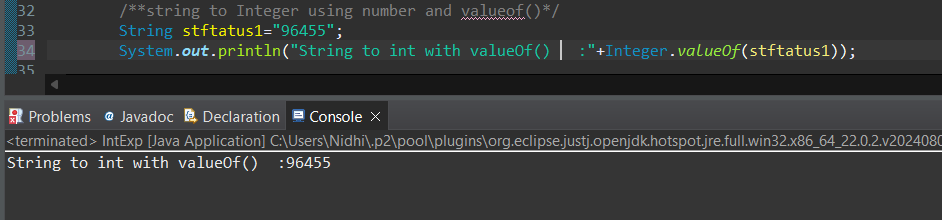
public static void main(String[] args) {

String stftatus1="96455";

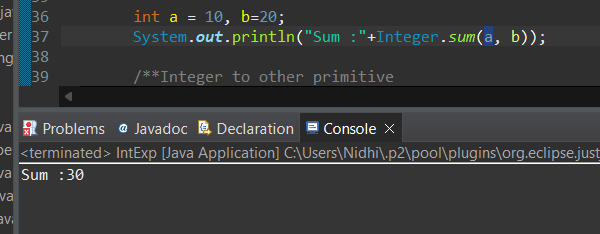
System.out.println("String to int with valueOf() :"+Integer.valueOf(stftatus1));

}

}



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)).



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

public class IntegerExp {

public static void main(String[] args) {

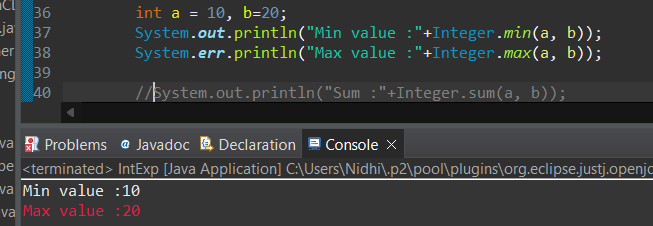
int a = 10, b=20;

System.out.println("Min value :"+Integer.min(a, b));

System.err.println("Max value :"+Integer.max(a, b));

}

}



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

public class IntegerExp {

public static void main(String[] args) {

int a = 10;

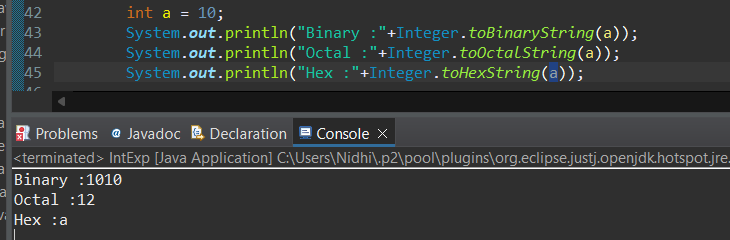
System.out.println("Binary :"+Integer.toBinaryString(a));

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

System.out.println("Hex :"+Integer.toHexString(a));

}

}



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

public class IntegerExp {

public static void main(String[] args) {

int d =124;

byte c = (byte)d;

System.out.println("Int "+Integer.valueOf(d));

System.err.println("Integer to byte via narrowing :"+c);

//System.out.println("Byte "+Byte.valueOf(c));// CTE

System.out.println((char)d);

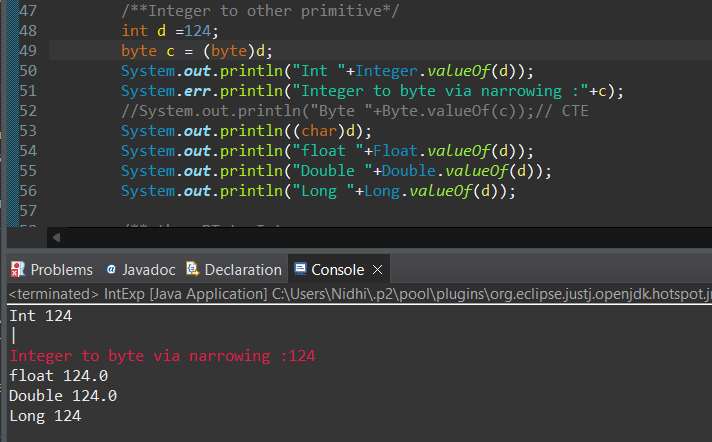
System.out.println("float "+Float.valueOf(d));

System.out.println("Double "+Double.valueOf(d));

System.out.println("Long "+Long.valueOf(d));

}

}



public class IntegerExp {

public static void main(String[] args) {

short x=129;

byte y=127;

float z=5683.05f;

double w=89647562.00058d;

long s = 895647523l;

char c = 'n';

System.out.println("Short to Integer :"+x);//widing

System.out.println("short to integer :"+Integer.valueOf(x));

System.out.println("Byte to Integer :"+Integer.valueOf(y));

System.out.println("Float to Integer :"+(int)z);//cant use value

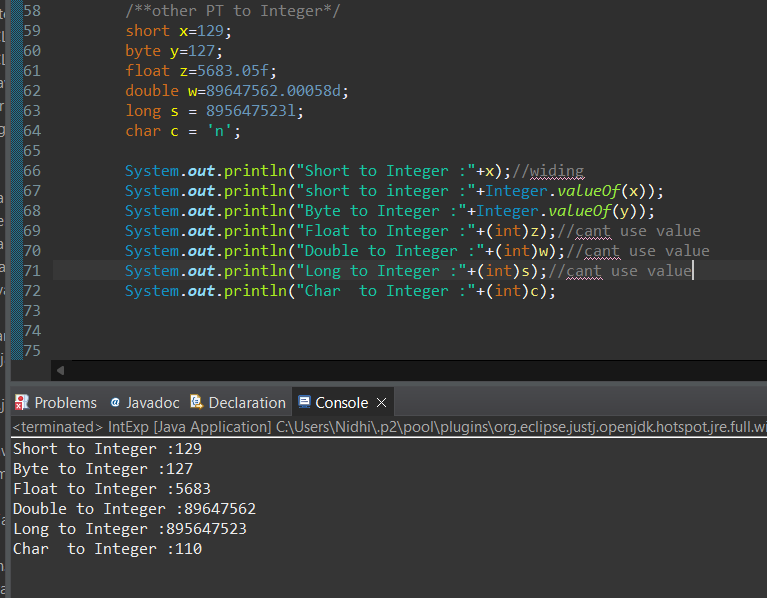
System.out.println("Double to Integer :"+(int)w);//cant use value

System.out.println("Long to Integer :"+(int)s);//cant use value

System.out.println("Char to Integer :"+(int)c);

}

}



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

**a.** Explore the [Java API documentation for java.lang.Long](https://docs.oracle.com/javase/8/docs/api/java/lang/Long.html) and observe its modifiers and super types.

public final class **Long** extends [Number](https://docs.oracle.com/javase/8/docs/api/java/lang/Number.html)

**The access modifier for Long is public**

**While the non-access modifier is final that mean we can’t create child class**

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

public class IntegerExp {

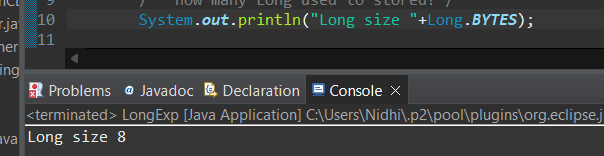
public static void main(String[] args) {

/\*\* how many Long used to stored?\*/

System.out.println("Long size "+Long.BYTES);

}

}



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

public class IntegerExp {

public static void main(String[] args) {

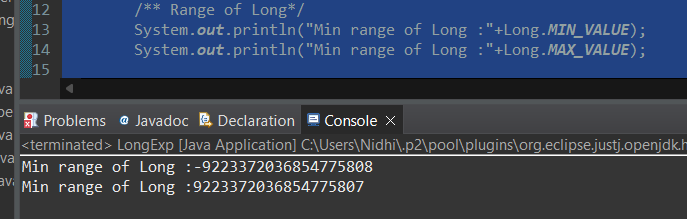
/\*\* Range of Long\*/

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

System.out.println("Min range of Long :"+Long.MAX\_VALUE);

}

}



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

public class IntegerExp {

public static void main(String[] args) {

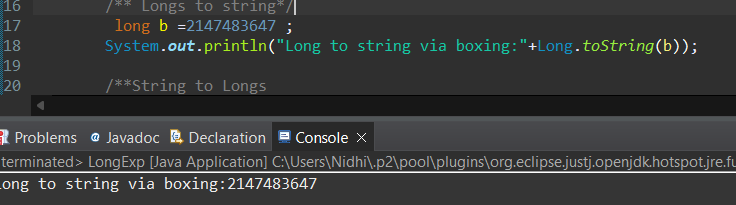
/\*\* Longs to string\*/

long b =2147483647 ;

System.out.println("Long to string via boxing:"+Long.toString(b));

}

}



**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)).

public class IntegerExp {

public static void main(String[] args) {

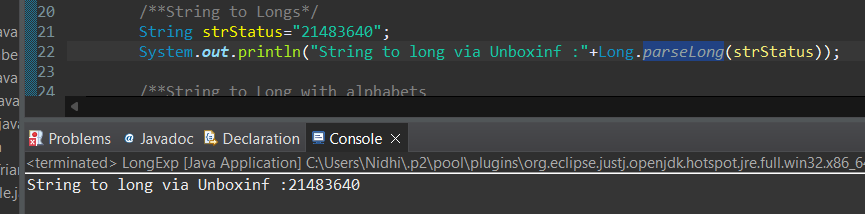
/\*\*String to Longs\*/

String strStatus="21483640";

System.out.println("String to long via Unboxinf :"+Long.parseLong(strStatus));

}

}



**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).

public class IntegerExp {

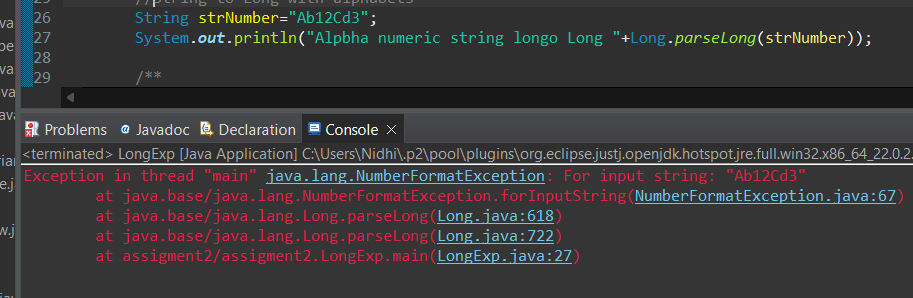
public static void main(String[] args) {

String strNumber="Ab12Cd3";

System.out.println("Alpbha numeric string longo Long "+Long.parseLong(strNumber));

}

}



**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)).

public class IntegerExp {

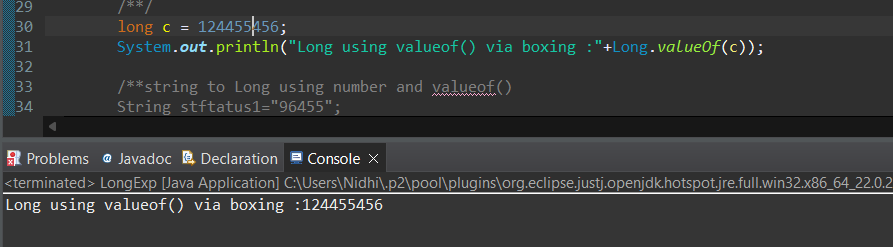
public static void main(String[] args) {

long c = 124455456;

System.out.println("Long using valueof() via boxing :"+Long.valueOf(c));

}

}



**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)).

public class IntegerExp {

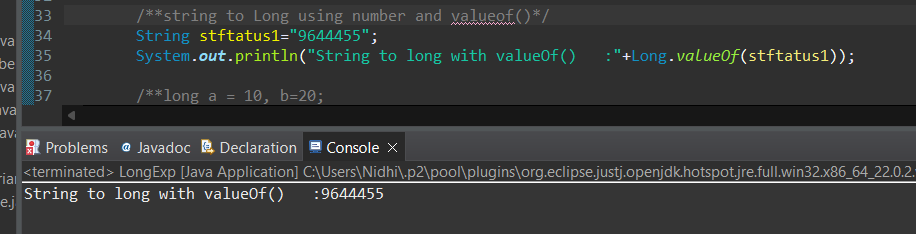
public static void main(String[] args) {

String stftatus1="9644455";

System.out.println("String to long with valueOf() :"+Long.valueOf(stftatus1));

}

}



1. 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)).

public class IntegerExp {

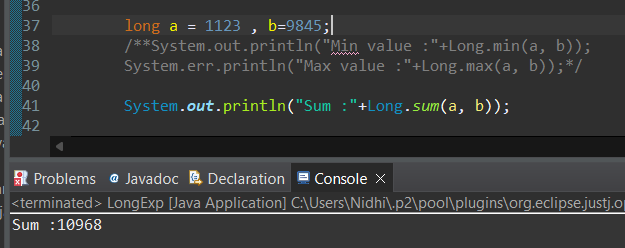
public static void main(String[] args) {

long a = 1123 , b=9845;

System.out.println("Sum :"+Long.sum(a, b));

}

}



**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)).

public class IntegerExp {

public static void main(String[] args) {

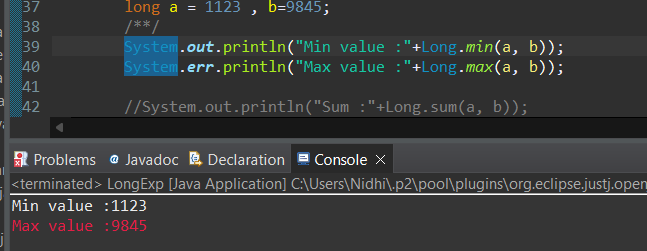
long a = 1123 , b=9845;

System.out.println("Min value :"+Long.min(a, b));

System.err.println("Max value :"+Long.max(a, b));

}

}



**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)).

public class IntegerExp {

public static void main(String[] args) {

long a = 14560;

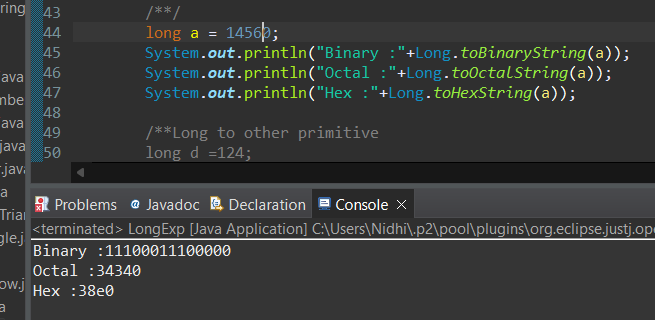
System.out.println("Binary :"+Long.toBinaryString(a));

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

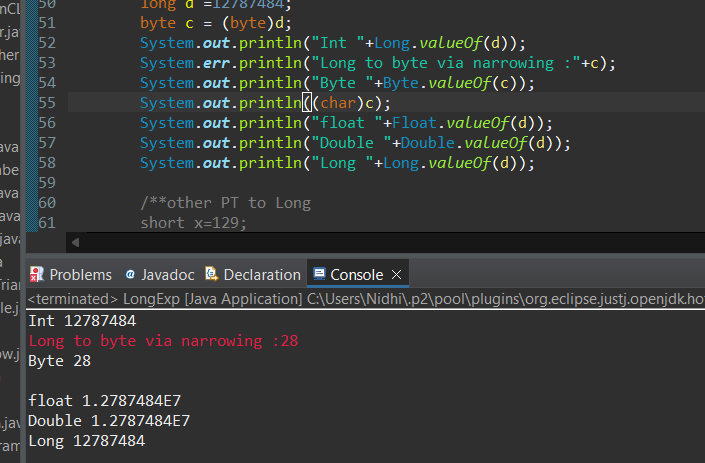
System.out.println("Hex :"+Long.toHexString(a));;

}

}



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



public class IntegerExp {

public static void main(String[] args) {

/\*\*other PT to Long\*/

short x=1294;

byte y=127;

float z=5683.05f;

double w=89647562.00058d;

long s = 895647523l;

char c = 'n';

System.out.println("Short to Long :"+x);//widing

System.out.println("short to longeger :"+Long.valueOf(x));

System.out.println("Byte to Long :"+Long.valueOf(y));

System.out.println("Float to Long :"+(long)z);

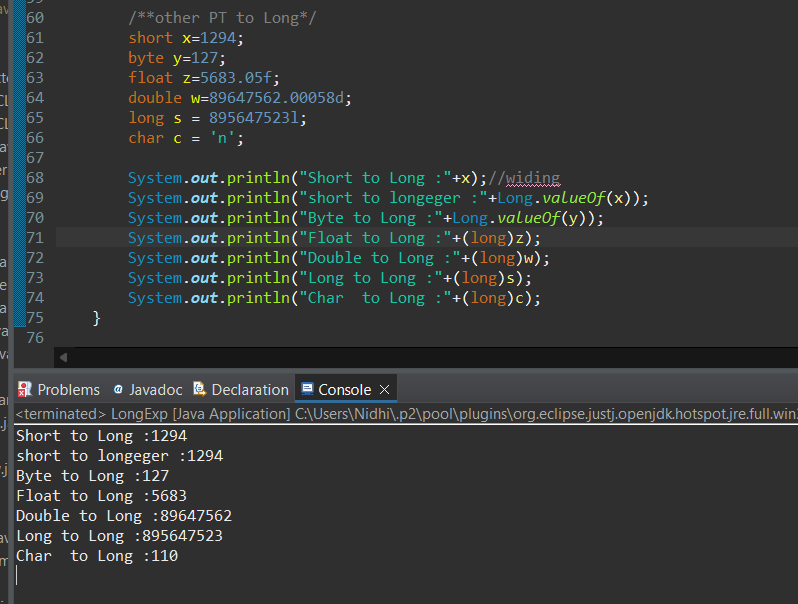
System.out.println("Double to Long :"+(long)w);

System.out.println("Long to Long :"+(long)s);

System.out.println("Char to Long :"+(long)c);

}

}



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

**a.** Explore the [Java API documentation for java.lang.Float](https://docs.oracle.com/javase/8/docs/api/java/lang/Float.html) and observe its modifiers and super types.

public final class **Float** extends [Number](https://docs.oracle.com/javase/8/docs/api/java/lang/Number.html)

The access modifier for Float is public and the non access modifier is static

The Float class is extended from Number, meanwhile the Number class is extended form Object, hence super class for Double class is Object

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

public class FloatExpr {

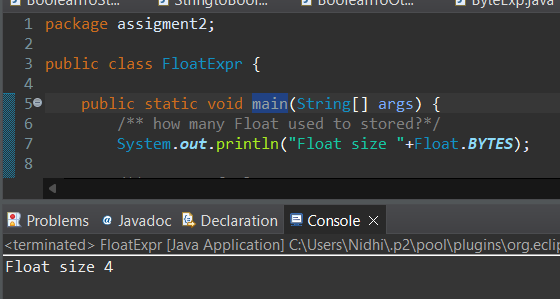
public static void main(String[] args) {

/\*\* how many Float used to stored?\*/

System.out.println("Float size "+Float.BYTES);

}

}



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

public class FloatExpr {

public static void main(String[] args) {

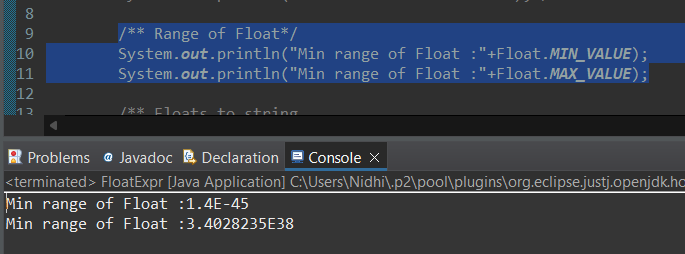
/\*\* Range of Float\*/

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

System.out.println("Min range of Float :"+Float.MAX\_VALUE);

}

}



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

public class FloatExpr {

public static void main(String[] args) {

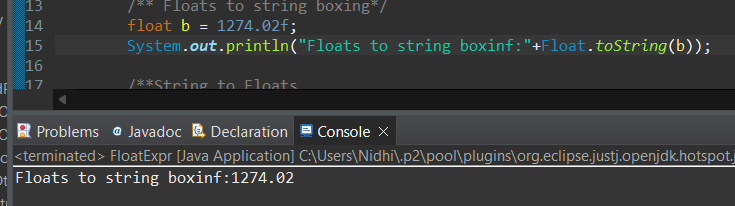
/\*\* Floats to string boxing\*/

float b = 1274.02f;

System.out.println("Floats to string boxinf:"+Float.toString(b));

}

}



**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)).

public class FloatExpr {

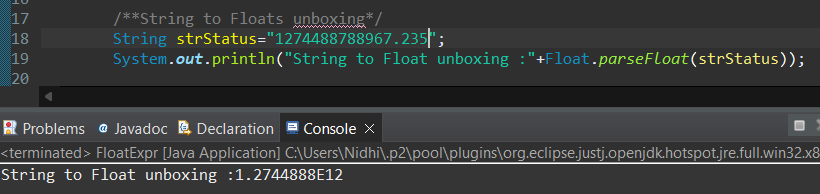
public static void main(String[] args) {

String strStatus="1274488788967.235";

System.out.println("String to Float unboxing :"+Float.parseFloat(strStatus));

}

}



**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).

public class FloatExpr {

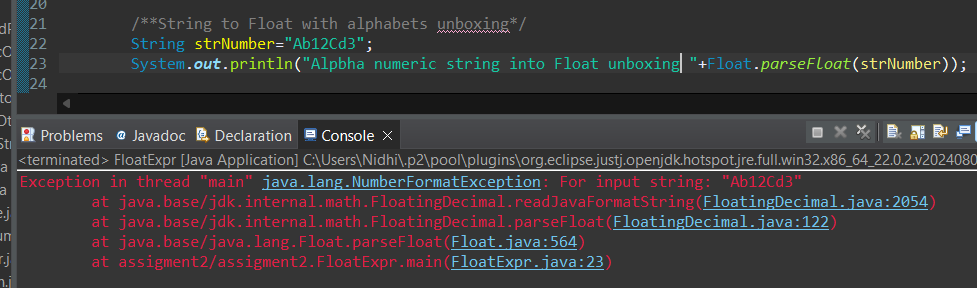
public static void main(String[] args) {

String strNumber="Ab12Cd3";

System.out.println("Alpbha numeric string into Float unboxing "+Float.parseFloat(strNumber));

}

}



**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)).

public class FloatExpr {

public static void main(String[] args) {

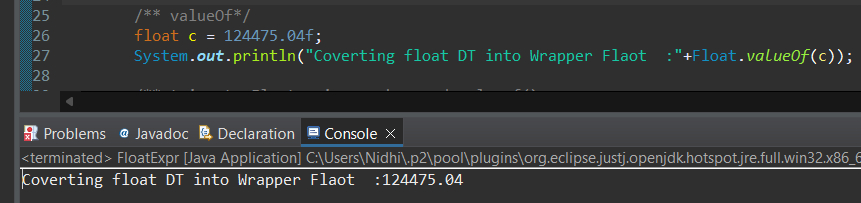
/\*\*string to Float using number and valueof()\*/

String stftatus1="9456.02f";

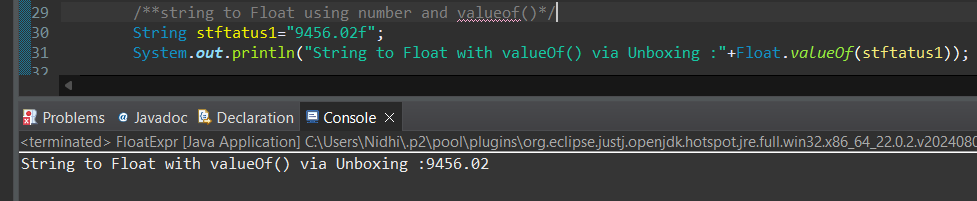
System.out.println("String to Float with valueOf() via Unboxing :"+Float.valueOf(stftatus1));

}

}



**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)).



1. 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)).

public class FloatExpr {

public static void main(String[] args) {

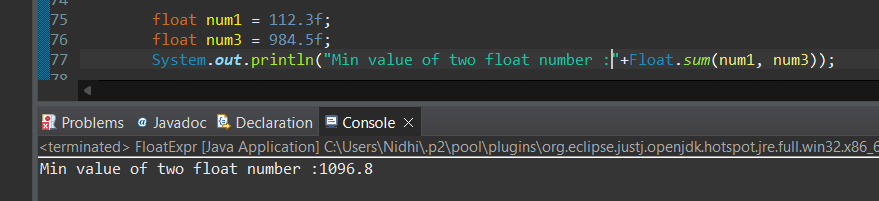
float num1 = 112.3f;

float num3 = 984.5f;

System.out.println("Min value of two float number :"+Float.sum(num1, num3));

}

}



**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)).

public class FloatExpr {

public static void main(String[] args) {

float num1 = 112.2f;

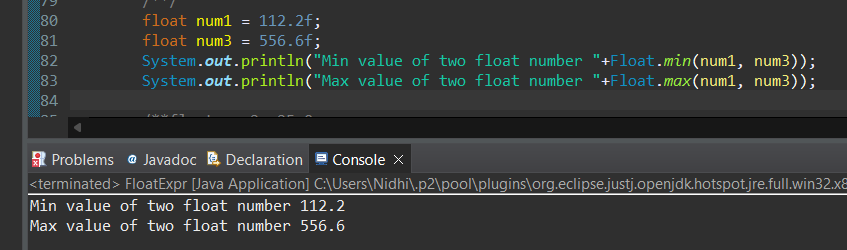
float num3 = 556.6f;

System.out.println("Min value of two float number "+Float.min(num1, num3));

System.out.println("Max value of two float number "+Float.max(num1, num3));

}

}



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

public class FloatExpr {

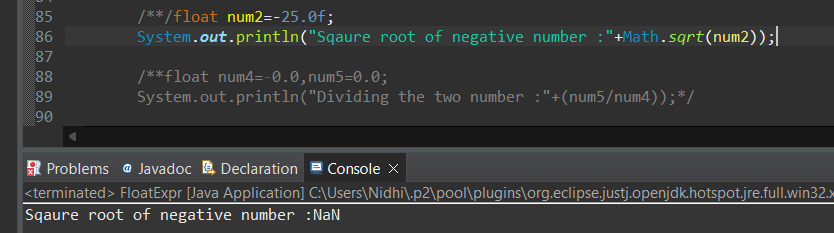
public static void main(String[] args) {

float num2=-25.0f;

System.out.println("Sqaure root of negative number :"+Math.sqrt(num2));

}

}



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

public class FloatExpr {

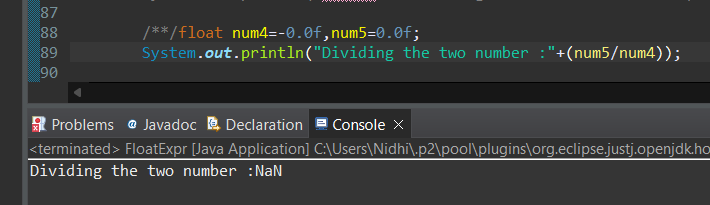
public static void main(String[] args) {

float num4=-0.0f,num5=0.0f;

System.out.println("Dividing the two number :"+(num5/num4));

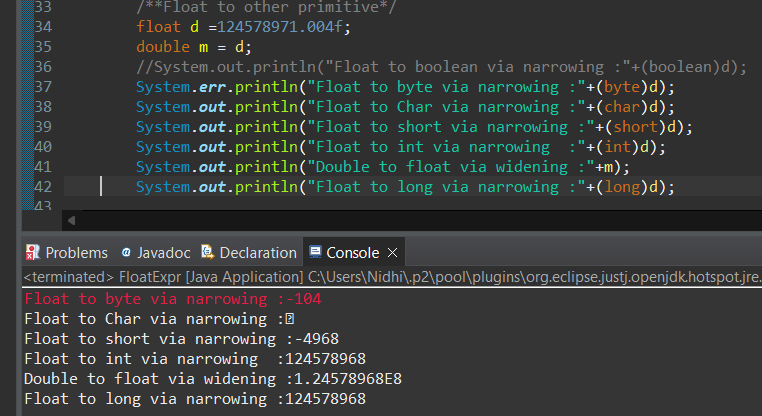
}

}



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

**Float to other dt via narrowing & Widening**



Other to float

public class FloatExpr {

public static void main(String[] args) {

/\*\*other PT to Float\*/

int x=129786;

byte y=127;

float z=5683.05f;

float w=89647562.00058f;

long s = 895647523l;

char c = 'n';

boolean b = true;

float x1 = x;

float y1=y;

float z1=z;

float w1 = w;

float s1 = s;

float c1 = c;

//float b1 = b;

System.out.println("Int to Float via Widening :"+x1);

System.out.println("Byte to Float via Widening :"+y1);

System.out.println("Float to Float via Widening :"+z1);

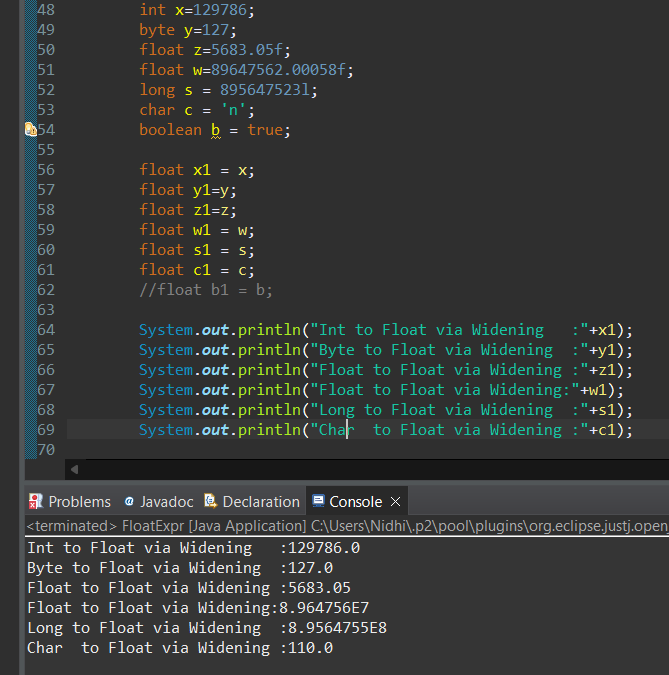
System.out.println("Float to Float via Widening:"+w1);

System.out.println("Long to Float via Widening :"+s1);

System.out.println("Char to Float via Widening :"+c1);

}

}



#### ****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.

public final class **Double** extends [Number](https://docs.oracle.com/javase/8/docs/api/java/lang/Number.html)

The access modifier for Double is public and the non access modifier is static

The Double class is extended from Number, meanwhile the Number class is extended form Object, hence super class for Double class is Object

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

public class DoubleExpr {

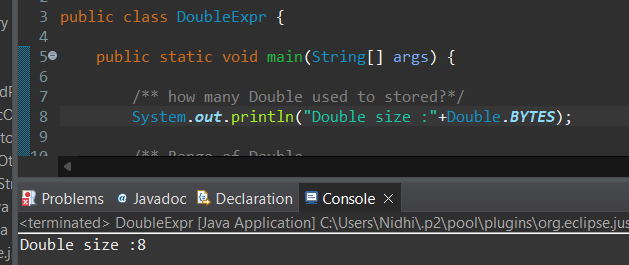
public static void main(String[] args) {

/\*\* how many Double used to stored?\*/

System.out.println("Double size :"+Double.BYTES);

}

}



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

public class DoubleExpr {

public static void main(String[] args) {

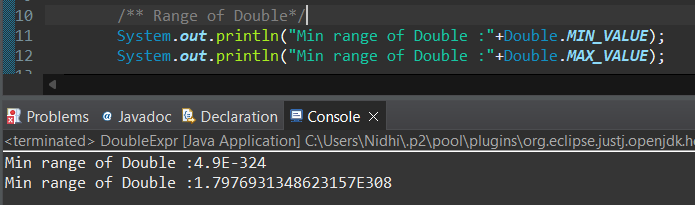
/\*\* Range of Double\*/

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

System.out.println("Min range of Double :"+Double.MAX\_VALUE);

}

}



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

public class DoubleExpr {

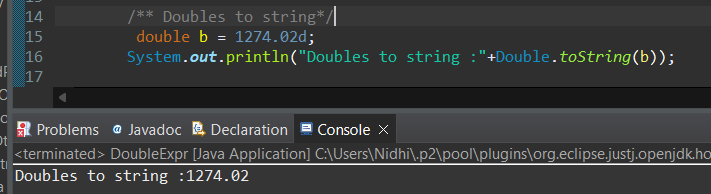
public static void main(String[] args) {

/\*\* Doubles to string\*/

double b = 1274.02d;

System.out.println("Doubles to string :"+Double.toString(b));

}

}

**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)).

public class DoubleExpr {

public static void main(String[] args) {

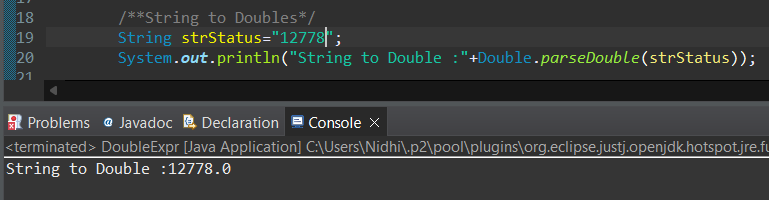
/\*\*String to Doubles\*/

String strStatus="12778";

System.out.println("String to Double :"+Double.parseDouble(strStatus));

}

}



**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).

public class DoubleExpr {

public static void main(String[] args) {

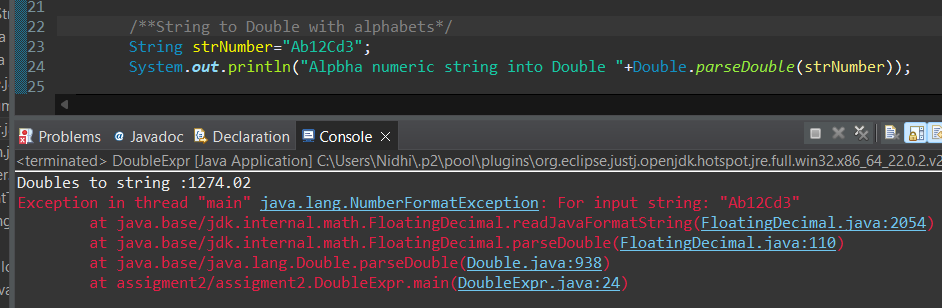
/\*\*String to Double with alphabets\*/

String strNumber="Ab12Cd3";

System.out.println("Alpbha numeric string into Double "+Double.parseDouble(strNumber));

}

}



**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)).

public class DoubleExpr {

public static void main(String[] args) {

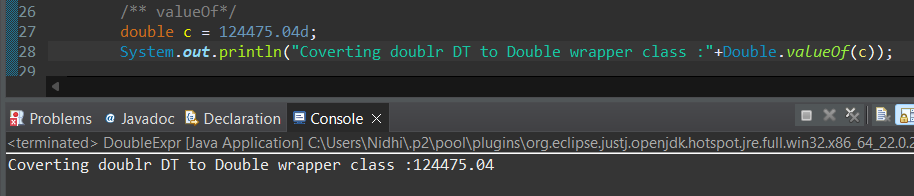
/\*\* valueOf\*/

double c = 124475.04d;

System.out.println("Coverting doublr DT to Double wrapper class :"+Double.valueOf(c));

}

}



**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)).

public class DoubleExpr {

public static void main(String[] args) {

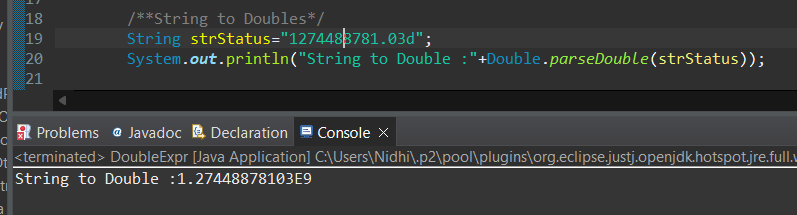
/\*\*String to Doubles\*/

String strStatus="1274488781.03d";

System.out.println("String to Double :"+Double.parseDouble(strStatus));

}

}



1. 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)).

public class DoubleExpr {

public static void main(String[] args) {

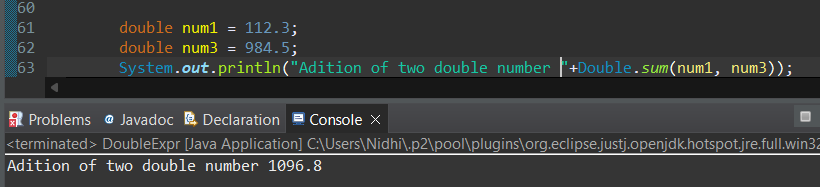
double num1 = 112.3;

double num3 = 984.5;

System.out.println("Adition of two double number "+Double.sum(num1, num3));

}

}



**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)).

public class DoubleExpr {

public static void main(String[] args) {

double num1 = 112.3;

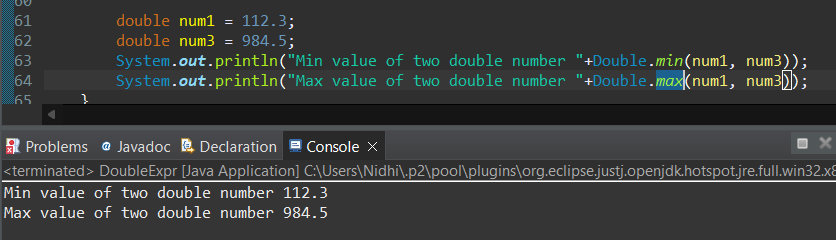
double num3 = 984.5;

System.out.println("Min value of two double number "+Double.min(num1, num3));

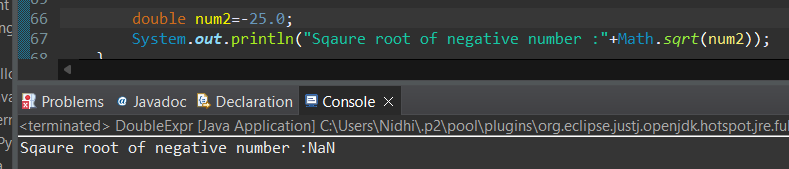
System.out.println("Max value of two double number "+Double.max(num1, num3));

}

}



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



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

public class DoubleExpr {

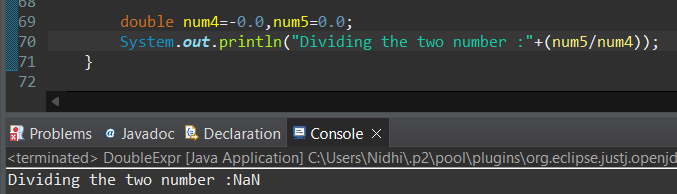
public static void main(String[] args) {

double num4=-0.0,num5=0.0;

System.out.println("Dividing the two number :"+(num5/num4));

}

}



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

**Double to other primitive DT**

**public class DoubleExpr {**

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

**double d =124578971.004d;**

**//System.out.println("Double to boolean via narrowing :"+(boolean)d);**

**System.err.println("Double to byte via narrowing :"+(byte)d);**

**System.out.println("Double to Char via narrowing :"+(char)d);**

**System.out.println("Double to short via narrowing :"+(short)d);**

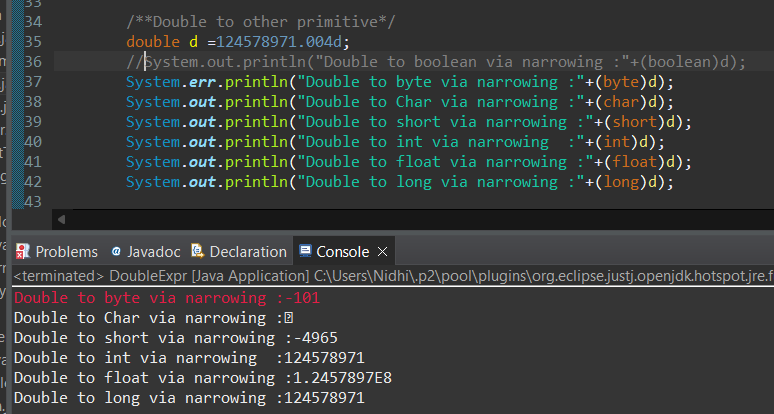
**System.out.println("Double to int via narrowing :"+(int)d);**

**System.out.println("Double to float via narrowing :"+(float)d);**

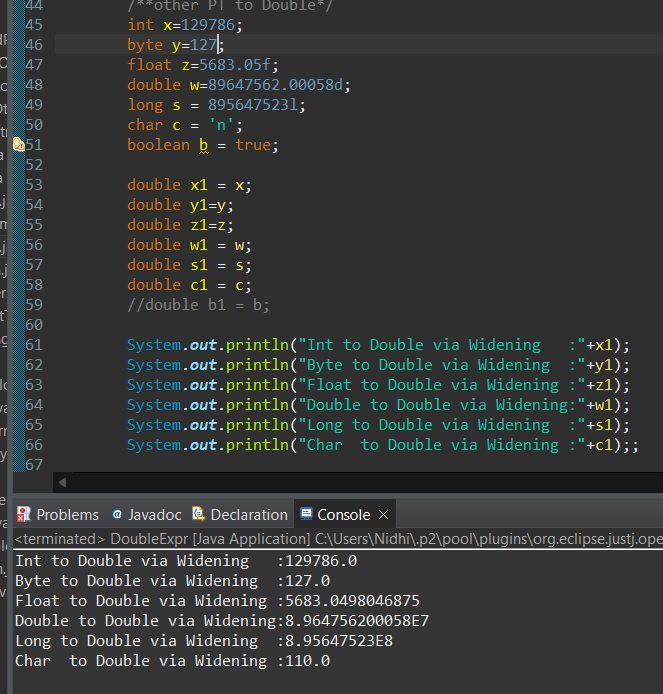
**System.out.println("Double to long via narrowing :"+(long)d);**

**}**

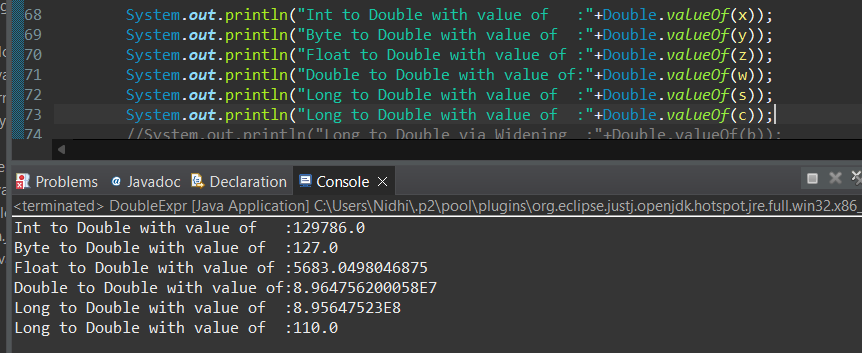
**}**



Other to Double DT



We can convert other DT to Double DT via ValuOf() too



#### ****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).

For static/class level field members

public class DefaultVlue {

static boolean *a*;

static byte *b*;

static short *c*;

static int *d*;

static char *e*;

static float *g*;

static double *h*;

static long *i*;

public static void main(String[] args) {

System.***out***.println("static Boolean default value :"+*a*);

System.***out***.println("static Byte default value :"+*b*);

System.***out***.println("static Short default value :"+*c*);

System.***out***.println("static Int default value :"+*d*);

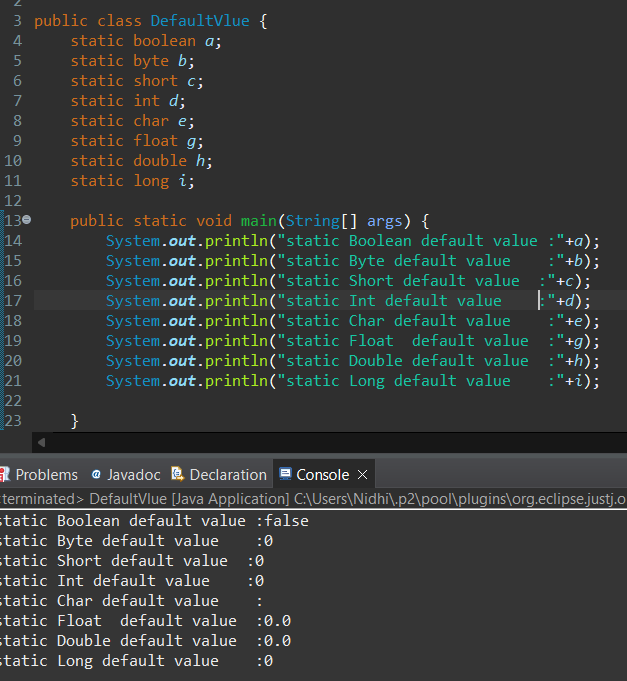
System.***out***.println("static Char default value :"+*e*);

System.***out***.println("static Float default value :"+*g*);

System.***out***.println("static Double default value :"+*h*);

System.***out***.println("static Long default value :"+*i*);

}



For Non static/Instance level FILED member DT

public class ByteExp {

boolean a;

byte b;

short c;

int d;

char e;

float g;

double h;

long i;

public static void main(String[] args) {

DefaultVlue dv = new DefaultVlue();

System.out.println("Non Static Boolean default value :"+dv.a);

System.out.println("Non Static Byte default value :"+dv.b);

System.out.println("Non Static Short default value :"+dv.c);

System.out.println("Non Static Int default value :"+dv.d);

System.out.println("Non Static Char default value :"+dv.e);

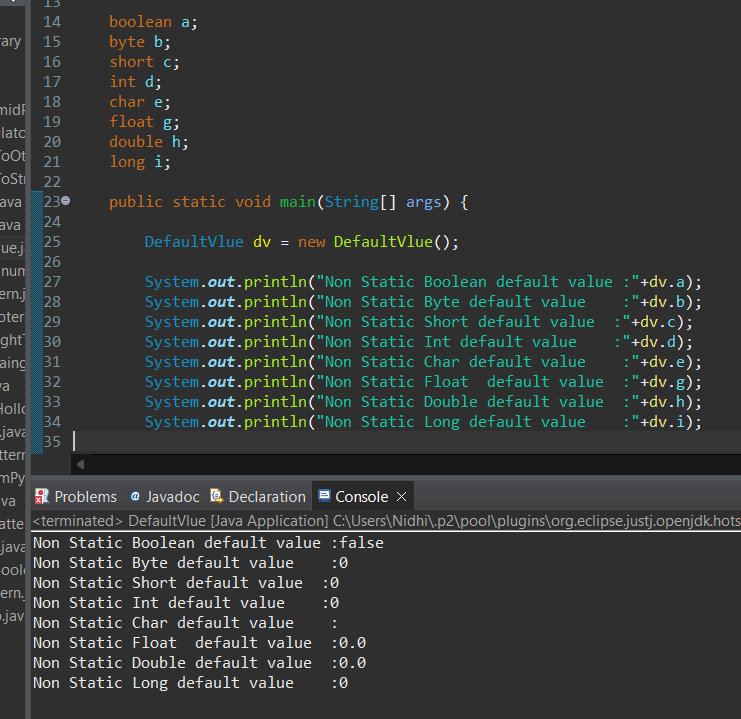
System.out.println("Non Static Float default value :"+dv.g);

System.out.println("Non Static Double default value :"+dv.h);

System.out.println("Non Static Long default value :"+dv.i);

}

}



#### ****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 assigment2;

public class ArithematicOnCLI {

public static void main(String[] args) {

String c = args[0];

int a = Integer.*parseInt*(args[2]);

int b = Integer.*parseInt*(args[3]);

switch (c){

case "+":

System.***out***.println("Addition of two numbers :"+ (a + b));

break;

case "-":

System.***out***.println("Subtraction of two numbers :"+ (a - b));

break;

case "\*":

System.***out***.println("Multiplication of two numbers :"+ a \* b);

break;

case "/":

System.***out***.println("Division of two numbers :"+ a / b);

break;

default :

System.***out***.println("Only enter + - \* / operator");

}

}

}