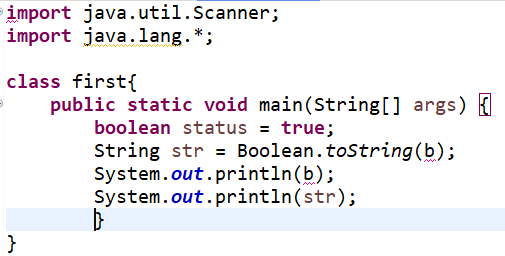
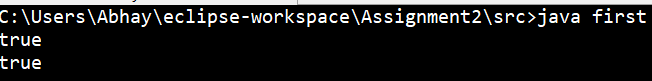
**ASSIGNMENT-2**

1. Working with java.lang.Boolean

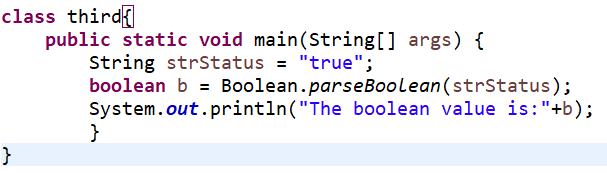
**a.** Explore the [Java API documentation for java.lang.Boolean](https://docs.oracle.com/javase/8/docs/api/java/lang/Boolean.html) and observe its modifiers and super types.

**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) .



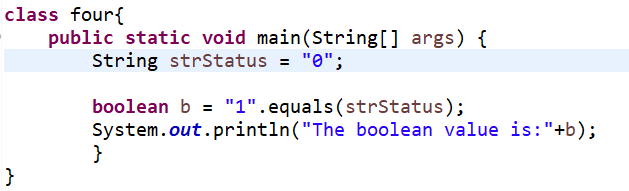


**c**. Declare a method-local variable strStatus of type String with the value "true" and convert it to a boolean using the parseBoolean method. (Hint: Use Boolean.parseBoolean(String)).



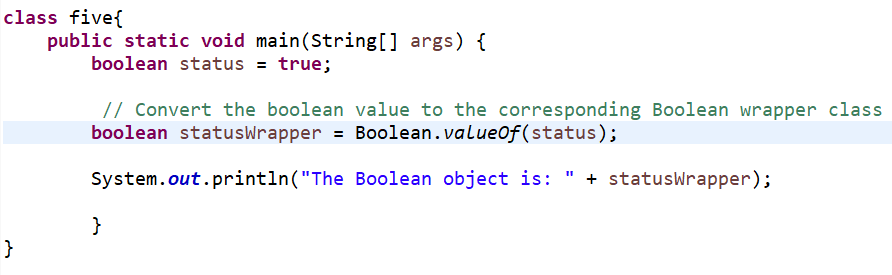
****

**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").

****

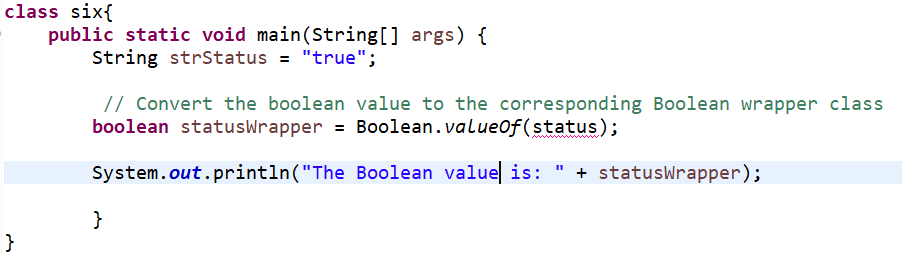
****

**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)).



****

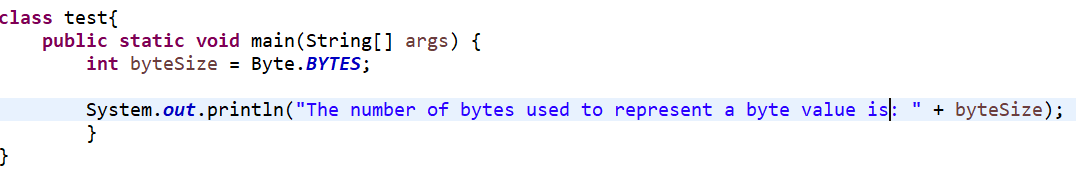
**f**. Declare a method-local variable strStatus of type String with the value "true" and convert it to the corresponding wrapper class using Boolean.valueOf(). (Hint: Use Boolean.valueOf(String)).



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

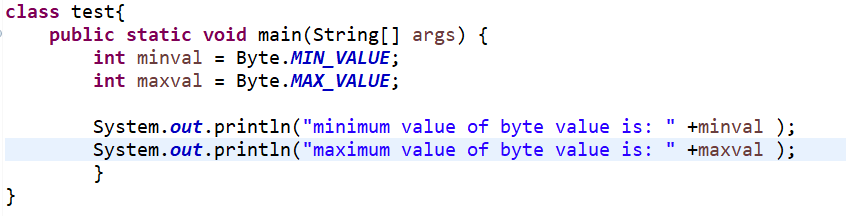
**a.** Explore the [Java API documentation for java.lang.Byte](https://docs.oracle.com/javase/8/docs/api/java/lang/Byte.html) and observe its modifiers and super types.

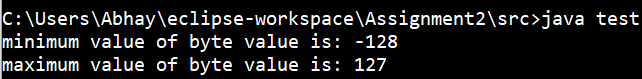
**b.** Write a program to test how many bytes are used to represent a byte value using the BYTES field. (Hint: Use Byte.BYTES).



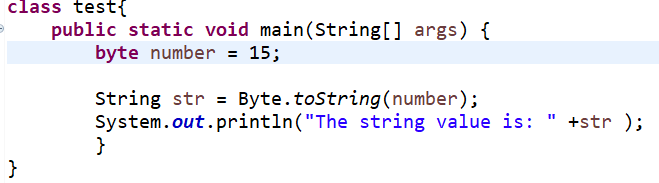
****

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).



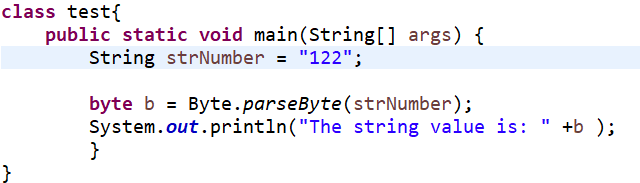
****

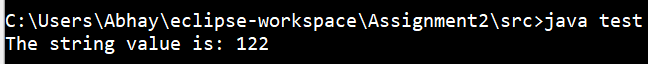
**d.** Declare a method-local variable number of type byte with some value and convert it to a String using the toString method. (Hint: Use Byte.toString(byte)).



****

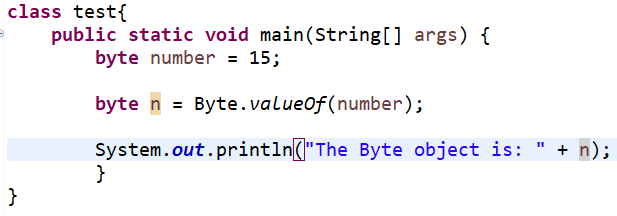
**e.** Declare a method-local variable strNumber of type String with some value and convert it to a byte value using the parseByte method. (Hint: Use Byte.parseByte(String)).



****

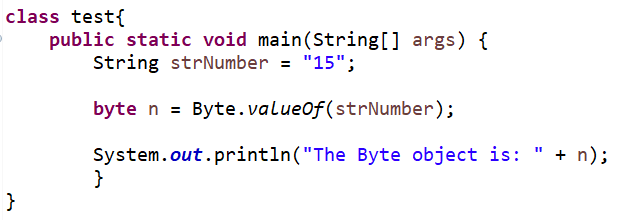
**f**. Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to a byte value. (Hint: parseByte method will throw a NumberFormatException).

**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)).

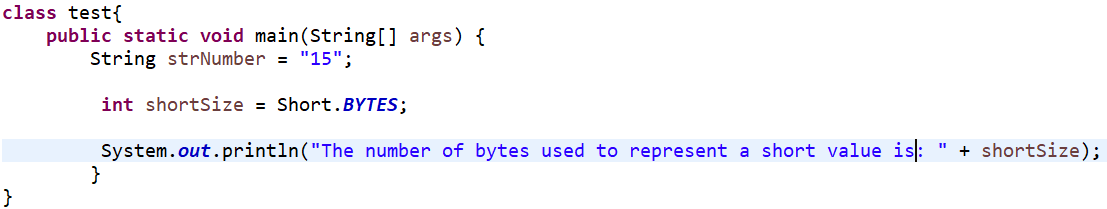




**3. Working with java.lang.Short**

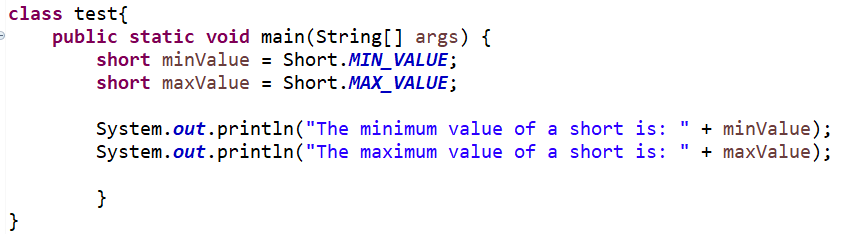
**a**. Explore the [Java API documentation for java.lang.Short](https://docs.oracle.com/javase/8/docs/api/java/lang/Short.html) and observe its modifiers and super types.

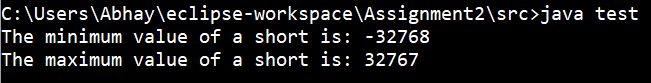
**b.** Write a program to test how many bytes are used to represent a short value using the BYTES field. (Hint: Use Short.BYTES).



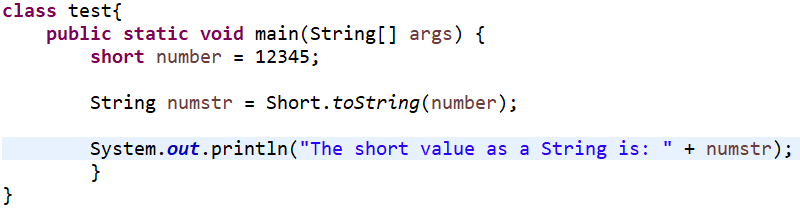


**c.** Write a program to find the minimum and maximum values of short using the MIN\_VALUE and MAX\_VALUE fields. (Hint: Use Short.MIN\_VALUE and Short.MAX\_VALUE).



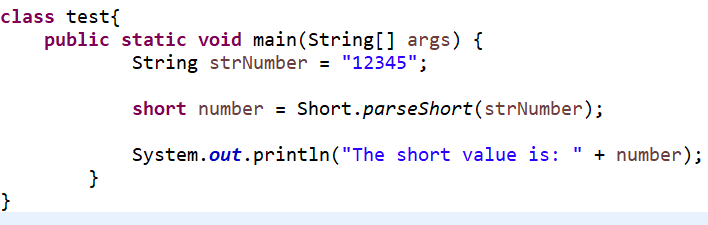


**d.** Declare a method-local variable number of type short with some value and convert it to a String using the toString method. (Hint: Use Short.toString(short)).





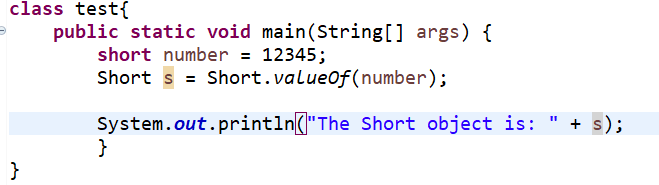
**e.** Declare a method-local variable strNumber of type String with some value and convert it to a short value using the parseShort method. (Hint: Use Short.parseShort(String)).





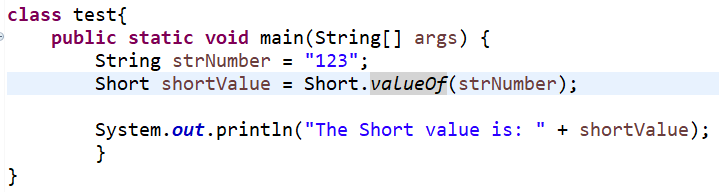
**f.** Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to a short value. (Hint: parseShort method will throw a NumberFormatException).

**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)).





**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)).

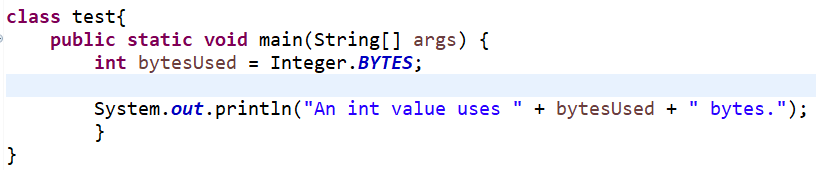




**4. Working with java.lang.Integer**

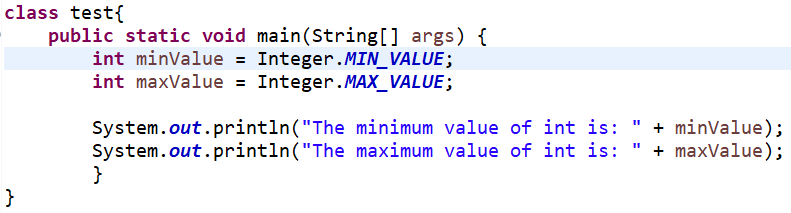
a. Explore the [Java API documentation for java.lang.Integer](https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html) and observe its modifiers and super types.

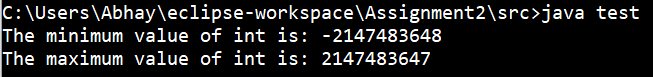
b. Write a program to test how many bytes are used to represent an int value using the BYTES field. (Hint: Use Integer.BYTES).



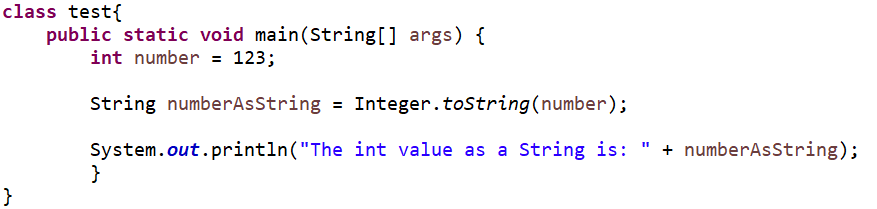


c. Write a program to find the minimum and maximum values of int using the MIN\_VALUE and MAX\_VALUE fields. (Hint: Use Integer.MIN\_VALUE and Integer.MAX\_VALUE).



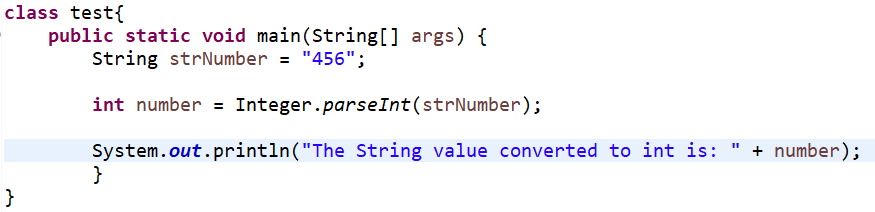


d. Declare a method-local variable number of type int with some value and convert it to a String using the toString method. (Hint: Use Integer.toString(int)).





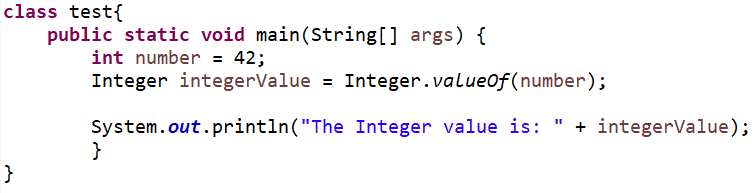
e. Declare a method-local variable strNumber of type String with some value and convert it to an int value using the parseInt method. (Hint: Use Integer.parseInt(String)).





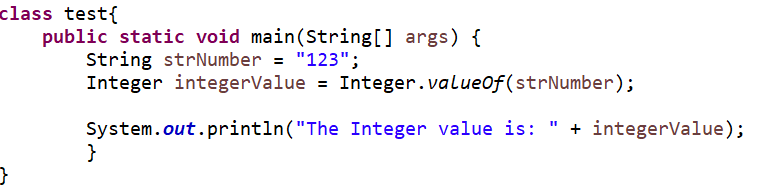
f. Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to an int value. (Hint: parseInt method will throw a NumberFormatException).

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)).



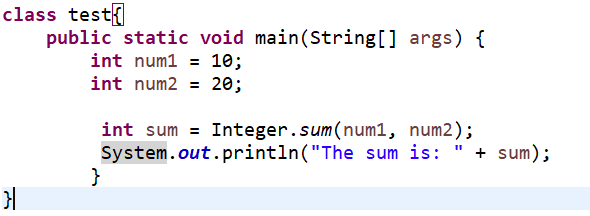


h. Declare a method-local variable strNumber of type String with some integer value and convert it to the corresponding wrapper class using Integer.valueOf(). (Hint: Use Integer.valueOf(String)).



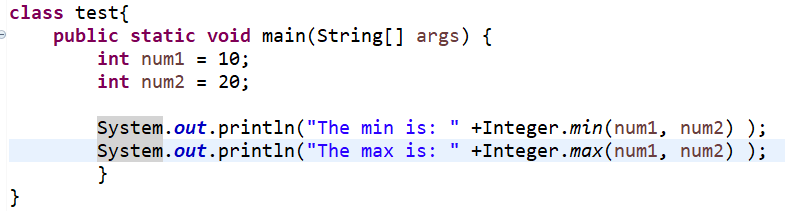


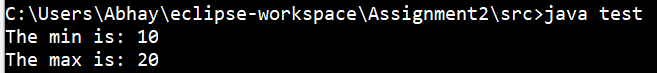
i. Declare two integer variables with values 10 and 20, and add them using a method from the Integer class. (Hint: Use Integer.sum(int, int)).



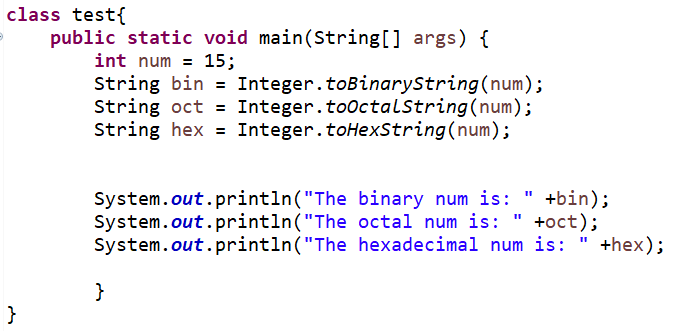


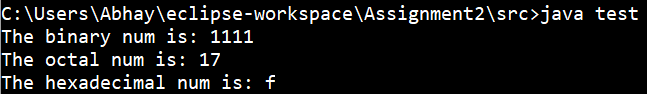
j. Declare two integer variables with values 10 and 20, and find the minimum and maximum values using the Integer class. (Hint: Use Integer.min(int, int) and Integer.max(int, int)).





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)).

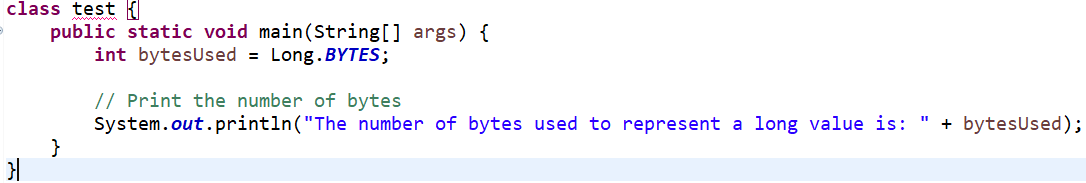




**5. Working with java.lang.Long**

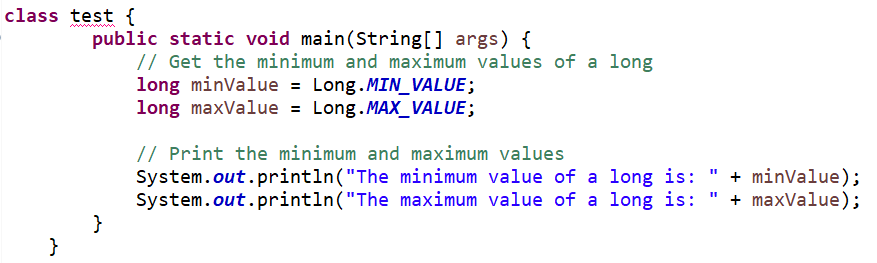
a. Explore the [Java API documentation for java.lang.Long](https://docs.oracle.com/javase/8/docs/api/java/lang/Long.html) and observe its modifiers and super types.

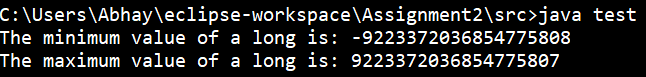
b. Write a program to test how many bytes are used to represent a long value using the BYTES field. (Hint: Use Long.BYTES).



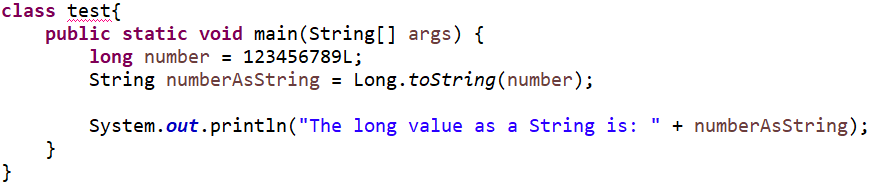


c. Write a program to find the minimum and maximum values of long using the MIN\_VALUE and MAX\_VALUE fields. (Hint: Use Long.MIN\_VALUE and Long.MAX\_VALUE).



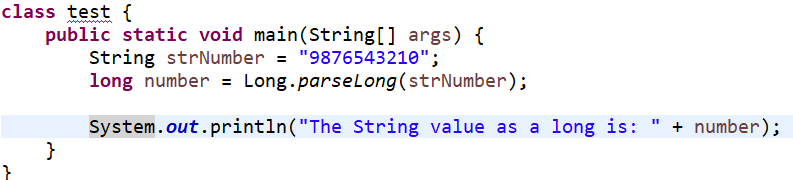


d. Declare a method-local variable number of type long with some value and convert it to a String using the toString method. (Hint: Use Long.toString(long)).



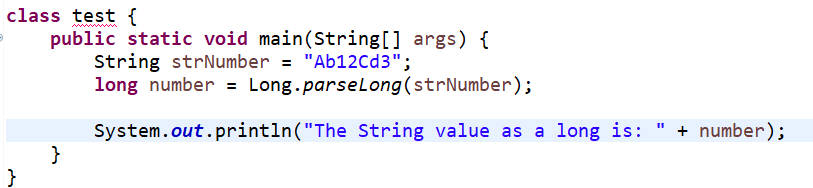


e. Declare a method-local variable strNumber of type String with some value and convert it to a long value using the parseLong method. (Hint: Use Long.parseLong(String)).

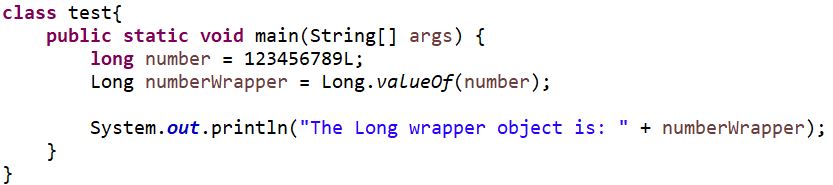




f. Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to a long value. (Hint: parseLong method will throw a NumberFormatException).

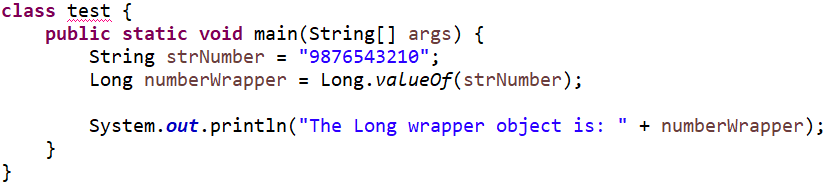


g. Declare a method-local variable number of type long with some value and convert it to the corresponding wrapper class using Long.valueOf(). (Hint: Use Long.valueOf(long)).



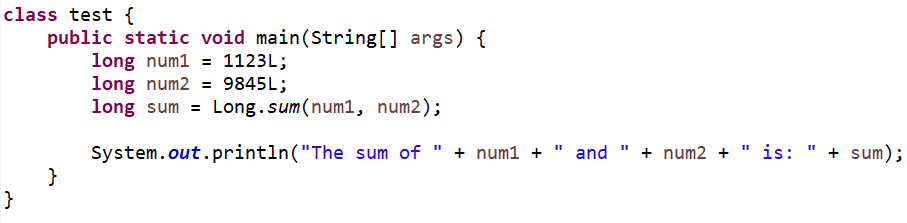


h. Declare a method-local variable strNumber of type String with some long value and convert it to the corresponding wrapper class using Long.valueOf(). (Hint: Use Long.valueOf(String)).

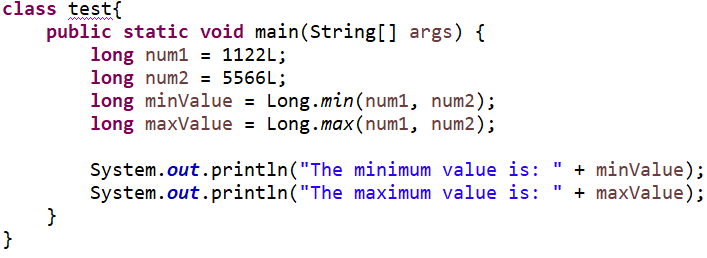


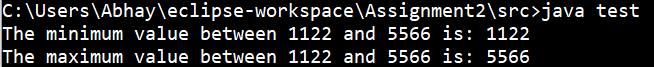


i. Declare two long variables with values 1123 and 9845, and add them using a method from the Long class. (Hint: Use Long.sum(long, long)).

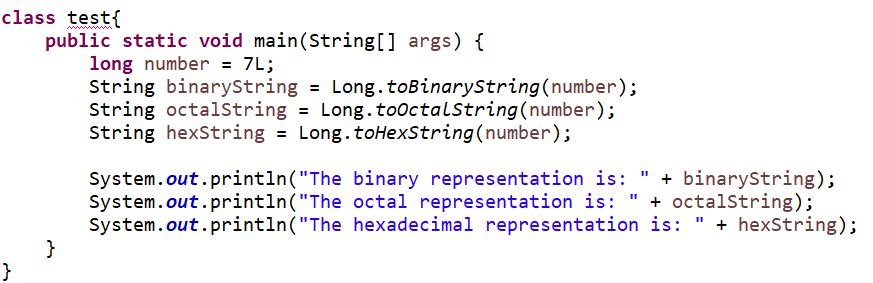
 

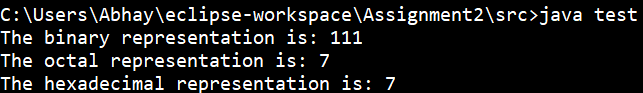
j. Declare two long variables with values 1122 and 5566, and find the minimum and maximum values using the Long class. (Hint: Use Long.min(long, long) and Long.max(long, long)).





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

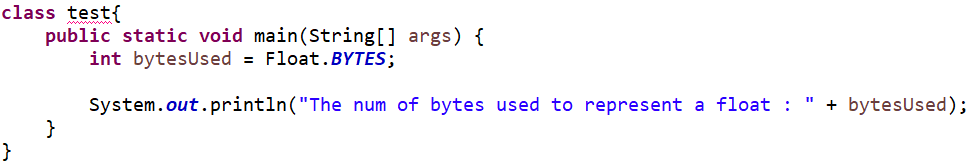




**6. Working with java.lang.Float**

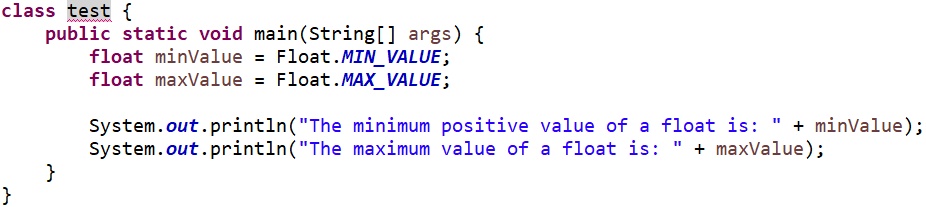
**a.** Explore the [Java API documentation for java.lang.Float](https://docs.oracle.com/javase/8/docs/api/java/lang/Float.html) and observe its modifiers and super types.

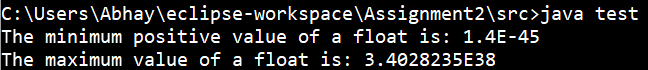
**b.** Write a program to test how many bytes are used to represent a float value using the BYTES field. (Hint: Use Float.BYTES).



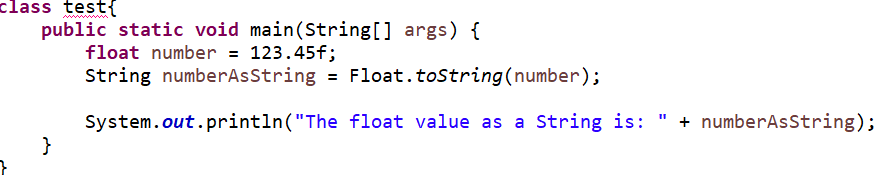


**c.** Write a program to find the minimum and maximum values of float using the MIN\_VALUE and MAX\_VALUE fields. (Hint: Use Float.MIN\_VALUE and Float.MAX\_VALUE).



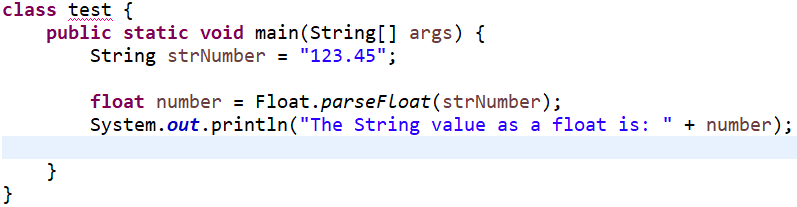


**d.** Declare a method-local variable number of type float with some value and convert it to a String using the toString method. (Hint: Use Float.toString(float)).



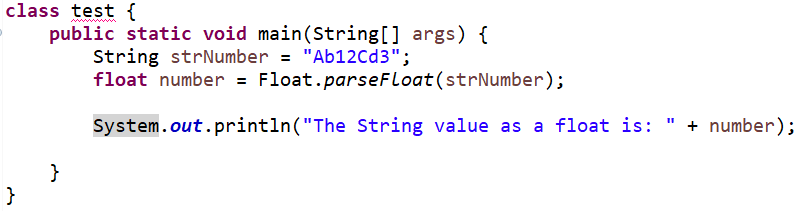


**e.** Declare a method-local variable strNumber of type String with some value and convert it to a float value using the parseFloat method. (Hint: Use Float.parseFloat(String)).

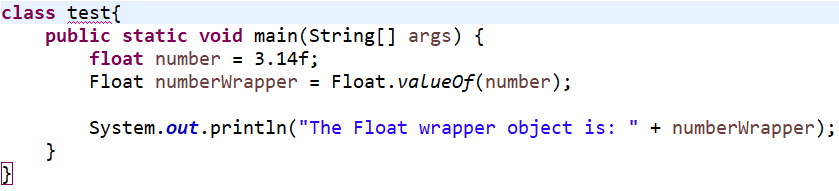




**f.** Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to a float value. (Hint: parseFloat method will throw a NumberFormatException).

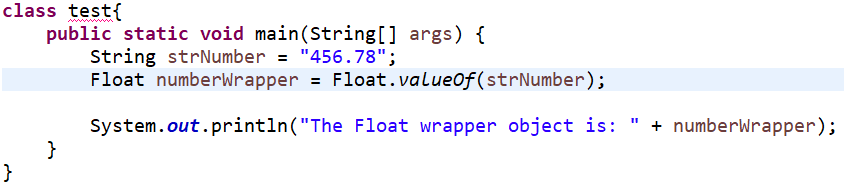


**g.** Declare a method-local variable number of type float with some value and convert it to the corresponding wrapper class using Float.valueOf(). (Hint: Use Float.valueOf(float)).



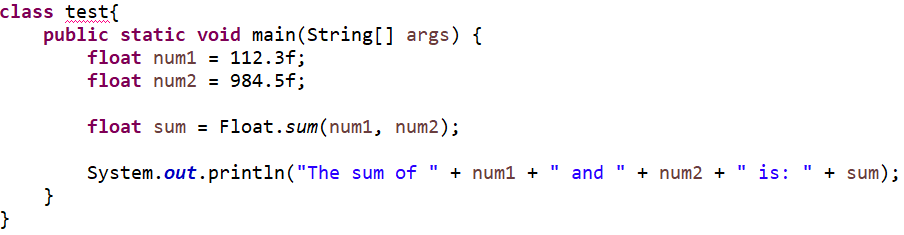


**h.** Declare a method-local variable strNumber of type String with some float value and convert it to the corresponding wrapper class using Float.valueOf(). (Hint: Use Float.valueOf(String)).



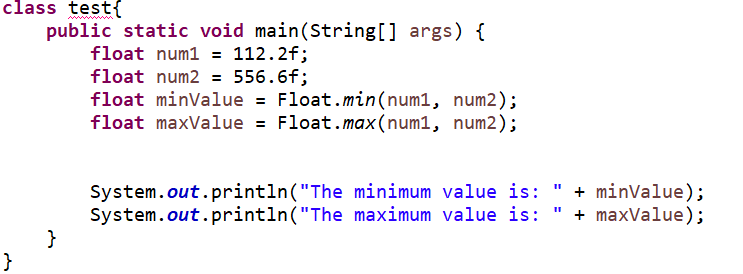


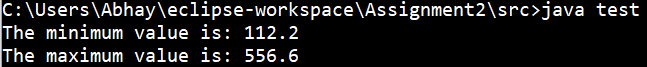
**i.** Declare two float variables with values 112.3 and 984.5, and add them using a method from the Float class. (Hint: Use Float.sum(float, float)).



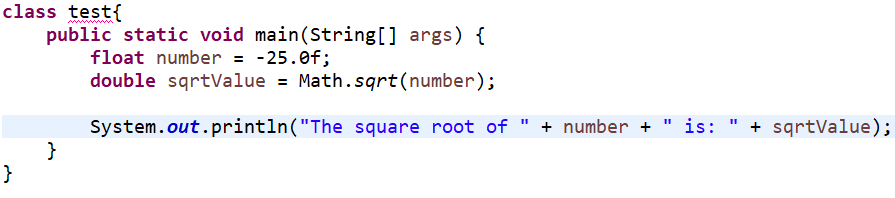


**j.** Declare two float variables with values 112.2 and 556.6, and find the minimum and maximum values using the Float class. (Hint: Use Float.min(float, float) and Float.max(float, float)).

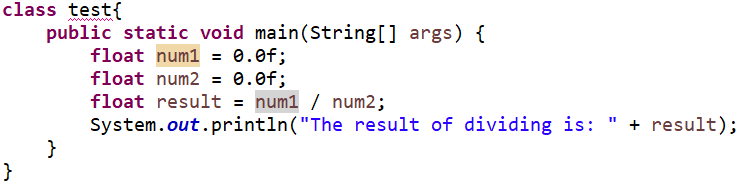




**k.** Declare a float variable with the value -25.0f. Find the square root of this value. (Hint: Use Math.sqrt() method).



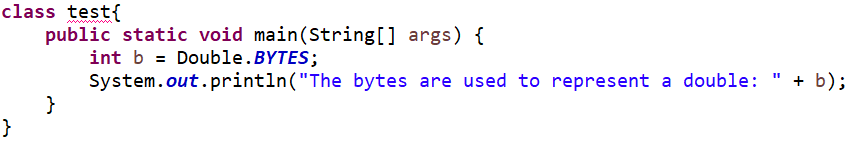
**l.** Declare two float variables with the same value, 0.0f, and divide them. (Hint: Observe the result and any special floating-point behavior).



**7. Working with java.lang.Double**

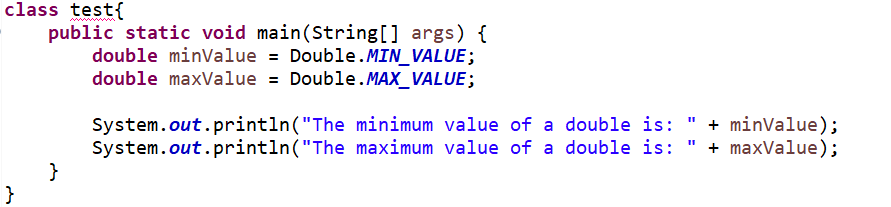
**a.** Explore the [Java API documentation for java.lang.Double](https://docs.oracle.com/javase/8/docs/api/java/lang/Double.html) and observe its modifiers and super types.

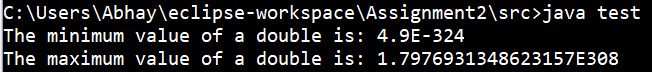
**b.** Write a program to test how many bytes are used to represent a double value using the BYTES field. (Hint: Use Double.BYTES).



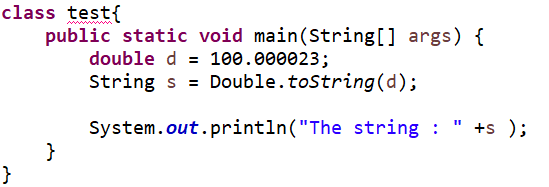


**c.** Write a program to find the minimum and maximum values of double using the MIN\_VALUE and MAX\_VALUE fields. (Hint: Use Double.MIN\_VALUE and Double.MAX\_VALUE).



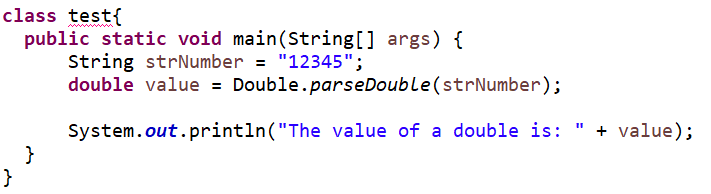


**d.** Declare a method-local variable number of type double with some value and convert it to a String using the toString method. (Hint: Use Double.toString(double)).



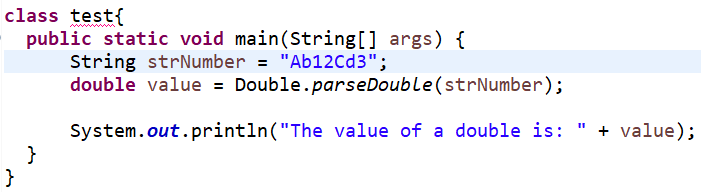


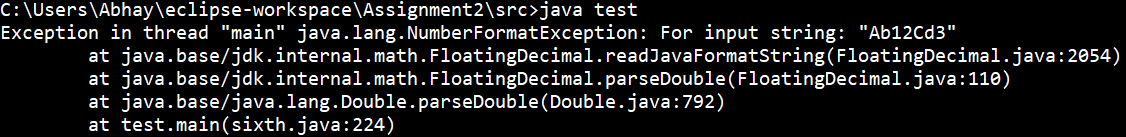
**e.** Declare a method-local variable strNumber of type String with some value and convert it to a double value using the parseDouble method. (Hint: Use Double.parseDouble(String)).



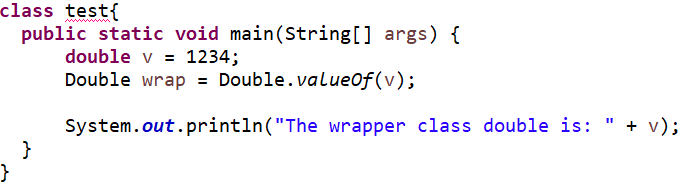


**f.** Declare a method-local variable strNumber of type String with the value "Ab12Cd3" and attempt to convert it to a double value. (Hint: parseDouble method will throw a NumberFormatException).



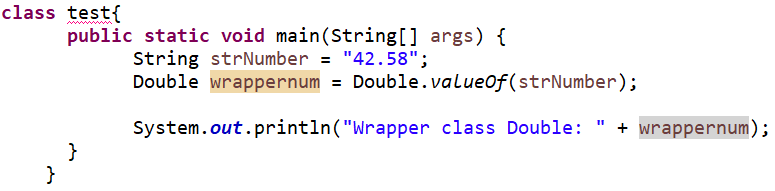


**g.** Declare a method-local variable number of type double with some value and convert it to the corresponding wrapper class using Double.valueOf(). (Hint: Use Double.valueOf(double)).



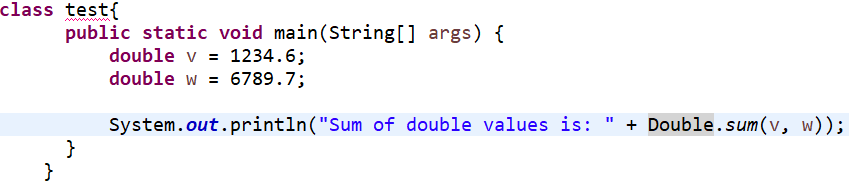


**h.** Declare a method-local variable strNumber of type String with some double value and convert it to the corresponding wrapper class using Double.valueOf(). (Hint: Use Double.valueOf(String)).



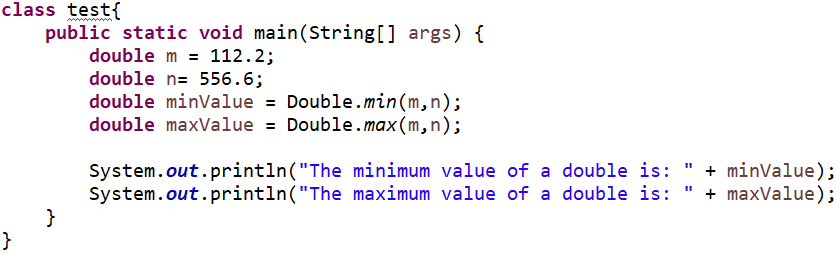


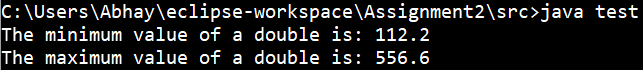
**i.** Declare two double variables with values 112.3 and 984.5, and add them using a method from the Double class. (Hint: Use Double.sum(double, double)).



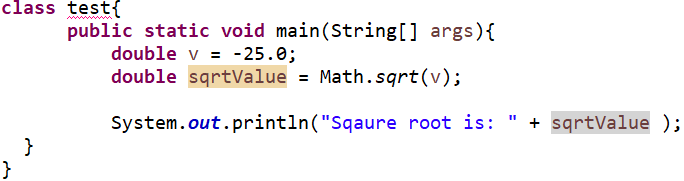


**j.** Declare two double variables with values 112.2 and 556.6, and find the minimum and maximum values using the Double class. (Hint: Use Double.min(double, double) and Double.max(double, double)).



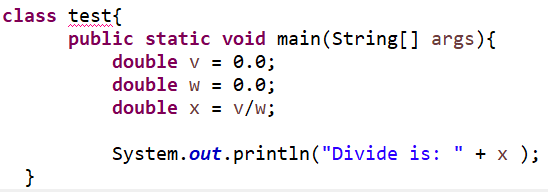


**k.** Declare a double variable with the value -25.0. Find the square root of this value. (Hint: Use Math.sqrt() method).





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

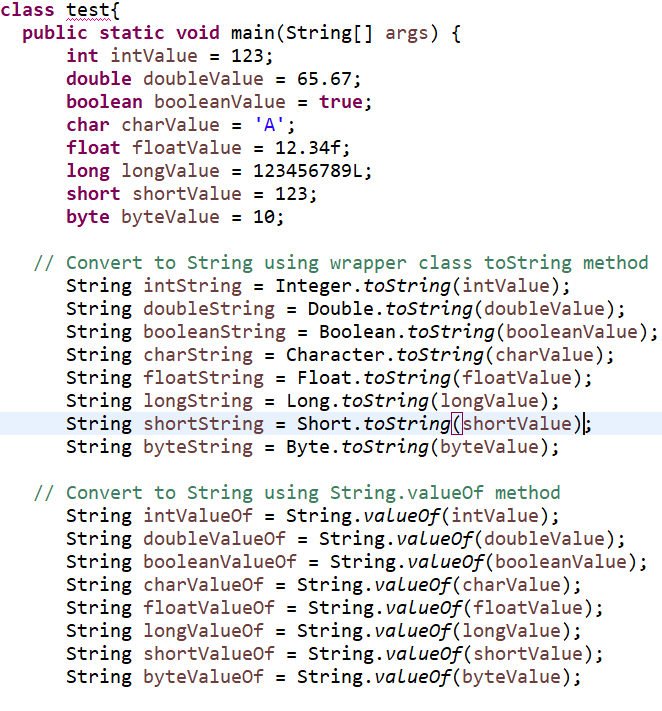


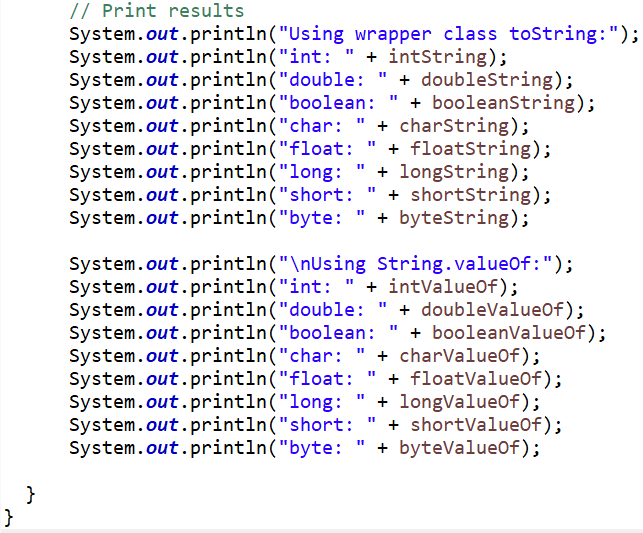


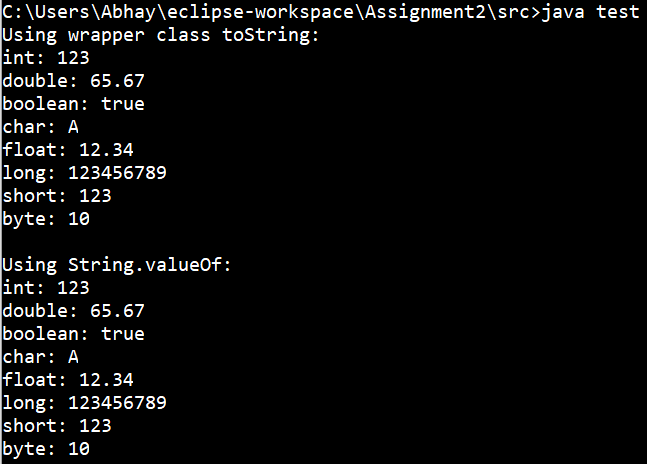
**8. Conversion between Primitive Types and Strings**

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).

Ans. In Java, if a char variable is declared but not explicitly initialized, it is assigned the default value '\u0000' or null.

**Null Character**: '\u0000' is the null character and has a Unicode value of 0.

