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 HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Boolean.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Boolean.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Boolean.html"java.lang.Boolean and observe its modifiers and super types.

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

-->

```
public class Main{

   public static void main(String [] args){

      boolean status = true;


      String statusString = Boolean.toString(status);


      System.out.println("The status as a String is:" + statusString);
```

```
   }
}
```

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 Mainn {
   public static void main(String [] args){
      String strStatus = "true";
      boolean status = Boolean.parseBoolean(strStatus);

      System.out.println("The status of boolean is : " + status);

   }
}
```

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

-->>

```
public class QuesD {
   public static void main(String [] args){
      String strStatus = "1";
      boolean status;
```

```
      if ("1".equals(strStatus)){
         status = true;
      }
      else if ("0".equals(strStatus)){
         status = false;
      }
      else {
         throw new IllegalArgumentException("Invalid Input: "
+strStatus);
         }
            System.out.println("The status as a boolean is :" +status);
   }


}
```

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 QuesE {
   public static void main(String[] args) {
      boolean status =true;
      Boolean statusWrapper = Boolean.valueOf(status);
      System.out.println("The status of Boolean is :"
```

```
+statusWrapper);
   }
}
```

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 QuesF {
   public static void main(String[] args) {
      String strStatus = "true";
      Boolean statusWrapper = Boolean.valueOf(strStatus);

      System.out.println("The status of Boolean is : "
+statusWrapper);
   }

}
```

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

2. Working with java.lang.Byte

a. Explore the Java API documentation for HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Byte.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Byte.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Byte.html"java.lang.Byte 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).

-->>

```
public class Datatype1 {

    public static void main(String[] args) {
            System.out.println("byte size is :" +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 Datatype2 {

public static void main(String[] args) {

   System.out.println("min size of byte is : " +Byte.MAX_VALUE);

   System.out.println("max size of byte is :" +Byte.MIN_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 Datatype3 {
        public static void main(String[] args) {
                byte number=4;
                String numberAsString = Byte.toString(number);
                System.out.println("string is : " +number);
   }
}
```

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 static void main(String[] args) {

String strnumber="6";
Byte AsString=Byte.parseByte(strnumber);
System.out.println("convert byte to string "  +strnumber);
}
```

```
}
```

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 Datatype11 {

    public static void main1(String[] args) {

            String strnumber ="Ab12Cd3" ;
            byte string=Byte.parseByte(strnumber);
        }
}
```

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 spublic class Datatype11 {
tatic void main2(String[] args) {
    byte number =100;
    System.out.println("byte number is"+Byte.valueOf(number));

}
```

```
}
```

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 Datatype11 {
public static void main(String[] args) {
        String strnumber="3";
        byte number=Byte.valueOf(strnumber);
        System.out.println("string is" +strnumber);
}
}
```

3. Working with java.lang.Short

a. Explore the Java API documentation for HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Short.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Short.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Short.html"java.lang.Short 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).

-->>

```
public class ShortBytesExample {

   public static void main(String[] args){

      System.out.println("Number of bytes used to represent a
short value: " + Short.BYTES);

   }

}
```

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

-->>

```
public class ShortMinMax{

   public static void main(String[] args) {

      System.out.println("Minimum value of short: " +
Short.MIN_VALUE);

      System.out.println("Maximum value 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 ShortToString{
```

```
    public static void main(String[] args) {
        short number = 12345;
        String strNumber = Short.toString(number);
        System.out.println("Short to String: " + strNumber);
    }
}
```

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 StringToShort {
    public static void main(String[] args) {
        String strNumber = "12345";
        short number = Short.parseShort(strNumber);
        System.out.println("String to Short: " + number);
    }
}
```

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

```
        String strNumber = "Ab12Cd3";
        try {
            short number = Short.parseShort(strNumber);
            System.out.println("Converted short: " + number);
        } catch (NumberFormatException e) {
            System.out.println("NumberFormatException: " +
e.getMessage());
        }
    }
}
```

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 ShortValue{
    public static void main(String[] args) {
        short number = 12345;
        Short shortWrapper = Short.valueOf(number);
        System.out.println("Short value of short: " + shortWrapper);
    }
}
```

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 StringToShortValueOfExample {
   public static void main(String[] args) {
      String strNumber = "12345";
      Short shortWrapper = Short.valueOf(strNumber);
      System.out.println("Short value of String: " + shortWrapper);
   }
}
```

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

4. Working with java.lang.Integer

a. Explore the Java API documentation for HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html"java.lang.Integer 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).

-->>

```
public class IntBytes {
   public static void main(String[] args) {
      int bytes = Integer.BYTES;
      System.out.println("Bytes used to represent int: " + 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 IntRange {
   public static void main(String[] args) {
      int minValue = Integer.MIN_VALUE;
      int maxValue = Integer.MAX_VALUE;
      System.out.println("Minimum int value: " + minValue);
      System.out.println("Maximum int value: " + maxValue);
   }
}
```

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 IntToString {
   public static void main(String[] args) {
      int number = 12345;
      String numberStr = Integer.toString(number);
      System.out.println("Integer as String: " + numberStr);
   }
}
```

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 StringToInt {
   public static void main(String[] args) {
      String strNumber = "6789";
      int number = Integer.parseInt(strNumber);
      System.out.println("String as Integer: " + number);
   }
}
```

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

-->>

```
public class InvalidStringToInt{
   public static void main(String[] args) {
      String strNumber = "Ab12Cd3";
      try {
         int number = Integer.parseInt(strNumber);
         System.out.println("String as Integer: " + number);
      } catch (NumberFormatException e) {
         System.out.println("Error: " + e.getMessage());
      }
   }
}
```

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 IntToInteger {

   public static void main(String[] args) {

      int number = 42;

      Integer integerObject = Integer.valueOf(number);

      System.out.println("Integer wrapper class object: " +
integerObject);

   }

}
```

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 StringToInteger {

   public static void main(String[] args) {

      String strNumber = "12345";

      Integer integerObject = Integer.valueOf(strNumber);

      System.out.println("String as Integer wrapper class object: " +
integerObject);

   }

}
```

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

Integer.sum(int, int)).

-->>

```
public class IntegerSum {
   public static void main(String[] args) {
      int a = 10;
      int b = 20;
      int sum = Integer.sum(a, b);
      System.out.println("Sum of 10 and 20: " + sum);
   }
}
```

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

-->>

```
public class IntegerMinMax {
   public static void main(String[] args) {
      int a = 10;
      int b = 20;
      int min = Integer.min(a, b);
      int max = Integer.max(a, b);
      System.out.println("Minimum of 10 and 20: " + min);
      System.out.println("Maximum of 10 and 20: " + max);
   }
}
```

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 IntToStringFormats {
   public static void main(String[] args) {
      int number = 7;
      String binary = Integer.toBinaryString(number);
      String octal = Integer.toOctalString(number);
      String hex = Integer.toHexString(number);

      System.out.println("Binary representation of 7: " + binary);
      System.out.println("Octal representation of 7: " + octal);
      System.out.println("Hexadecimal representation of 7: " +
hex);
   }
}
```

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

5. Working with java.lang.Long

a. Explore the Java API documentation for HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Long.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Long.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Long.htm

HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Long.html"java.lang.Long 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).

-->>

```
public class LongBytesTest {
   public static void main(String[] args) {
      int bytes = Long.BYTES;
      System.out.println("Bytes used to represent a 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 LongRange{
   public static void main(String[] args) {
      long minValue = Long.MIN_VALUE;
      long maxValue = Long.MAX_VALUE;
      System.out.println("Minimum long value: " + minValue);
      System.out.println("Maximum long value: " + maxValue);
   }
}
```

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 LongToString {
   public static void main(String[] args) {
      long number = 123456789L;
      String numberStr = Long.toString(number);
      System.out.println("Long as String: " + numberStr);
   }
}
```

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 StringToLong {
   public static void main(String[] args) {
      String strNumber = "9876543210";
      long number = Long.parseLong(strNumber);
      System.out.println("String as Long: " + number);
   }
}
```

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 InvalidStringToLong {
```

```
    public static void main(String[] args) {
        String strNumber = "Ab12Cd3";
        try {
            long number = Long.parseLong(strNumber);
            System.out.println("String as Long: " + number);
        } catch (NumberFormatException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

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 LongToLongWrapper{
    public static void main(String[] args) {
        long number = 42L;
        Long longObject = Long.valueOf(number);
        System.out.println("Long wrapper class object: " +
longObject);
    }
}
```

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 StringToLongWrapper{
   public static void main(String[] args) {
      String strNumber = "1234567890";
      Long longObject = Long.valueOf(strNumber);
      System.out.println("String as Long wrapper class object: " +
longObject);
   }
}
```

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)).

-->>

```
public class LongSum{
   public static void main(String[] args) {
      long a = 1123L;
      long b = 9845L;
      long sum = Long.sum(a, b);
      System.out.println("Sum of 1123 and 9845: " + sum);
   }
}
```

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 LongMinMax{
   public static void main(String[] args) {
      long a = 1122L;
      long b = 5566L;
      long min = Long.min(a, b);
      long max = Long.max(a, b);
      System.out.println("Minimum of 1122 and 5566: " + min);
      System.out.println("Maximum of 1122 and 5566: " + max);
   }
}
```

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 LongToStringFormats{
   public static void main(String[] args) {
      long number = 7L;
      String binary = Long.toBinaryString(number);
      String octal = Long.toOctalString(number);
      String hex = Long.toHexString(number);

      System.out.println("Binary representation of 7: " + binary);
      System.out.println("Octal representation of 7: " + octal);
      System.out.println("Hexadecimal representation of 7: " +
```

```
hex);
    }
}
```

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

6. Working with java.lang.Float

a. Explore the Java API documentation for HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Float.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Float.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Float.html"java.lang.Float and observe its modifiers and super types.

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

-->>

```
public class FloatBytesTest {
    public static void main(String[] args) {
        int bytes = Float.BYTES;
        System.out.println("Bytes used to represent a 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 FloatRangeTest {
```

```
    public static void main(String[] args) {
        float minValue = Float.MIN_VALUE;
        float maxValue = Float.MAX_VALUE;
        System.out.println("Minimum float value: " + minValue);
        System.out.println("Maximum float value: " + maxValue);
    }
}
```

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 FloatToStringTest {
    public static void main(String[] args) {
        float number = 12.34f;
        String numberStr = Float.toString(number);
        System.out.println("Float as String: " + numberStr);
    }
}
```

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 StringToFloatTest {
    public static void main(String[] args) {
        String strNumber = "56.78";
```

```
        float number = Float.parseFloat(strNumber);
        System.out.println("String as Float: " + number);
    }
}
```

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 InvalidStringToFloatTest {
    public static void main(String[] args) {
        String strNumber = "Ab12Cd3";
        try {
            float number = Float.parseFloat(strNumber);
            System.out.println("String as Float: " + number);
        } catch (NumberFormatException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

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

```
        float number = 45.67f;

        Float floatObject = Float.valueOf(number);

        System.out.println("Float wrapper class object: " +
floatObject);

    }

}
```

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)).

```
-->>

public class StringToFloatWrapperTest {

    public static void main(String[] args) {

        String strNumber = "89.01";

        Float floatObject = Float.valueOf(strNumber);

        System.out.println("String as Float wrapper class object: " +
floatObject);

    }

}
```

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)).

```
-->>

public class FloatSumTest {

    public static void main(String[] args) {

        float a = 112.3f;

        float b = 984.5f;
```

```
        float sum = Float.sum(a, b);
        System.out.println("Sum of 112.3 and 984.5: " + sum);
    }
}
```

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 FloatMinMaxTest {
    public static void main(String[] args) {
        float a = 112.2f;
        float b = 556.6f;
        float min = Float.min(a, b);
        float max = Float.max(a, b);
        System.out.println("Minimum of 112.2 and 556.6: " + min);
        System.out.println("Maximum of 112.2 and 556.6: " + max);
    }
}
```

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

-->>

```
public class FloatSqrtTest {
    public static void main(String[] args) {
        float value = -25.0f;
        double sqrt = Math.sqrt(value);  // Note: Math.sqrt() returns
```

a double

```
        System.out.println("Square root of -25.0: " + sqrt);
    }
}
```

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 FloatDivisionTest {
    public static void main(String[] args) {
        float a = 0.0f;
        float b = 0.0f;
        float result = a / b;
        System.out.println("Result of 0.0f divided by 0.0f: " + result);
        System.out.println("Is result NaN? " + Float.isNaN(result));
    }
}
```

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

7. Working with java.lang.Double

a. Explore the Java API documentation for HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Double.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Double.html" HYPERLINK "https://docs.oracle.com/javase/8/docs/api/java/lang/Double.html"java.lang.Double and observe its modifiers and super types.

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

-->>

```
public class DoubleBytesTest {

   public static void main(String[] args) {

      int bytes = Double.BYTES;

      System.out.println("Bytes used to represent a 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 DoubleRangeTest {

   public static void main(String[] args) {

      double minValue = Double.MIN_VALUE;

      double maxValue = Double.MAX_VALUE;

      System.out.println("Minimum double value: " + minValue);

      System.out.println("Maximum double value: " + maxValue);

   }

}
```

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 DoubleToStringTest {
   public static void main(String[] args) {
      double number = 1234.56;
      String numberStr = Double.toString(number);
      System.out.println("Double as String: " + numberStr);
   }
}
```

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 StringToDoubleTest {
   public static void main(String[] args) {
      String strNumber = "789.01";
      double number = Double.parseDouble(strNumber);
      System.out.println("String as Double: " + number);
   }
}
```

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

```
        String strNumber = "Ab12Cd3";
        try {
            double number = Double.parseDouble(strNumber);
            System.out.println("String as Double: " + number);
        } catch (NumberFormatException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

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 DoubleToDoubleWrapperTest {
    public static void main(String[] args) {
        double number = 456.78;
        Double doubleObject = Double.valueOf(number);
        System.out.println("Double wrapper class object: " +
doubleObject);
    }
}
```

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 StringToDoubleWrapperTest {
   public static void main(String[] args) {
      String strNumber = "1234.56";
      Double doubleObject = Double.valueOf(strNumber);
      System.out.println("String as Double wrapper class object: " +
doubleObject);
   }
}
```

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)).

-->>

```
public class DoubleSumTest {
   public static void main(String[] args) {
      double a = 112.3;
      double b = 984.5;
      double sum = Double.sum(a, b);
      System.out.println("Sum of 112.3 and 984.5: " + sum);
   }
}
```

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 DoubleMinMaxTest {
```

```
    public static void main(String[] args) {
        double a = 112.2;
        double b = 556.6;
        double min = Double.min(a, b);
        double max = Double.max(a, b);
        System.out.println("Minimum of 112.2 and 556.6: " + min);
        System.out.println("Maximum of 112.2 and 556.6: " + max);
    }
}
```

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

-->>

```
public class DoubleSqrtTest {
    public static void main(String[] args) {
        double value = -25.0;
        double sqrt = Math.sqrt(value);  // Note: Math.sqrt() returns NaN for negative inputs
        System.out.println("Square root of -25.0: " + sqrt);
    }
}
```

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 DoubleDivisionTest {
```

```
    public static void main(String[] args) {
        double a = 0.0;
        double b = 0.0;
        double result = a / b;
        System.out.println("Result of 0.0 divided by 0.0: " + result);
        System.out.println("Is result NaN? " + Double.isNaN(result));
        System.out.println("Is result Infinity? " +
Double.isInfinite(result));
    }
}
```

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

8. Conversion between Primitive Types and Strings

Initialize a variable of each primitive type with a user-defined value and convert it into String:

- First, use the toString method of the corresponding wrapper class. (e.g., Integer.toString()).
- Then, use the valueOf method of the String class. (e.g., String.valueOf()).

```
-->>
public class PrimitiveToStringConversion {
  public static void main(String[] args) {

    byte byteValue = 10;
    short shortValue = 100;
```

```
int intValue = 1000;
long longValue = 10000L;
float floatValue = 10.5f;
double doubleValue = 20.5;
char charValue = 'A';
boolean booleanValue = true;


String byteStringToString = Byte.toString(byteValue);
String shortStringToString = Short.toString(shortValue);
String intStringToString = Integer.toString(intValue);
String longStringToString = Long.toString(longValue);
String floatStringToString = Float.toString(floatValue);
String doubleStringToString = Double.toString(doubleValue);
String charStringToString = Character.toString(charValue);
String booleanStringToString = Boolean.toString(booleanValue);

String byteStringValueOf = String.valueOf(byteValue);
String shortStringValueOf = String.valueOf(shortValue);
String intStringValueOf = String.valueOf(intValue);
String longStringValueOf = String.valueOf(longValue);
String floatStringValueOf = String.valueOf(floatValue);
String doubleStringValueOf = String.valueOf(doubleValue);
```

```
String charStringValueOf = String.valueOf(charValue);
String booleanStringValueOf = String.valueOf(booleanValue);


System.out.println("Using toString method:");
System.out.println("Byte: " + byteStringToString);
System.out.println("Short: " + shortStringToString);
System.out.println("Int: " + intStringToString);
System.out.println("Long: " + longStringToString);
System.out.println("Float: " + floatStringToString);
System.out.println("Double: " + doubleStringToString);
System.out.println("Char: " + charStringToString);
System.out.println("Boolean: " + booleanStringToString);

System.out.println("\nUsing valueOf method:");
System.out.println("Byte: " + byteStringValueOf);
System.out.println("Short: " + shortStringValueOf);
System.out.println("Int: " + intStringValueOf);
System.out.println("Long: " + longStringValueOf);
System.out.println("Float: " + floatStringValueOf);
System.out.println("Double: " + doubleStringValueOf);
System.out.println("Char: " + charStringValueOf);
System.out.println("Boolean: " + booleanStringValueOf);
```

```
  }
}
```

9. Default Values of Primitive Types

Declare variables of each primitive type as fields of a class and check their default values. (Note: Default values depend on whether the variables are instance variables or static variables).

-->>

```
package com.org.program.demo;
    public class PrimitiveDefaults {
       byte instanceByte;
      short instanceShort;
      int instanceInt;
      long instanceLong;
      float instanceFloat;
      double instanceDouble;
      char instanceChar;
      boolean instanceBoolean;
      static byte staticByte;
      static short staticShort;
      static int staticInt;
      static long staticLong;
      static float staticFloat;
      static double staticDouble;
      static char staticChar;
```

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

PrimitiveDefaults defaults = new PrimitiveDefaults();


    System.out.println("Instance variable default values:");
    System.out.println("byte: " + defaults.instanceByte);
    System.out.println("short: " + defaults.instanceShort);
    System.out.println("int: " + defaults.instanceInt);
    System.out.println("long: " + defaults.instanceLong);
    System.out.println("float: " + defaults.instanceFloat);
    System.out.println("double: " + defaults.instanceDouble);
    System.out.println("char: [" + defaults.instanceChar + "]");
    System.out.println("boolean: " + defaults.instanceBoolean);


    System.out.println("\nStatic variable default values:");
    System.out.println("byte: " + staticByte);
    System.out.println("short: " + staticShort);
    System.out.println("int: " + staticInt);
    System.out.println("long: " + staticLong);
    System.out.println("float: " + staticFloat);
    System.out.println("double: " + staticDouble);
    System.out.println("char: [" + staticChar + "]");
```

```
        System.out.println("boolean: " + staticBoolean);
    }
}
```

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 com.org.program.demo;


public class LongDatatype {


    public static void main1(String[] args) {
        System.out.println("long conversion is"  +Long.BYTES);
        System.out.println("min value"+Long.MIN_VALUE);
        System.out.println("max value"+Long.MAX_VALUE);
    }
    public static void main2(String[] args) {
     long number=1234456;
  System.out.println("string is" +Long.toString(number));
    }
    public static void main3(String[] args) {
        String strnumber="abcd";
```

```
        System.out.println(Long.parseLong(strnumber));
}
public static void main4(String[] args) {
        String strnumber="Ab12Cd3" ;
        System.out.println(Long.parseLong(strnumber));
}
public static void main5(String[] args) {
        long num=5678;
        String num2="abcd";
        System.out.println(Long.valueOf(num));
        System.out.println(Long.valueOf(num2));
}
public static void main6(String[] args) {
        long num=100;
        long num1=200;
        System.out.println(Long.sum(num, num1));
}
public static void main7(String[] args) {
        long num1=1122;
        long num2=5566;
        System.out.println("min number"+Long.min(num1, num2));
        System.out.println("max number"+Long.max(num1, num2));
}
public static void main(String[] args) {
  long num=7;
```

```
  System.out.println(Long.toBinaryString(num));
  System.out.println(Long.toOctalString(num));
  System.out.println(Long.toHexString(num));

}
}
```