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` 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)`).

```
package Assignment.Assignment2;



public class test {

   public static void main(String[] args) {

      boolean status = true;

      String s = Boolean.toString(status);



      System.out.println(s);

   }

}
```

Output:-

true

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

Code:

```
public class P3 {

        public static void main(String[] args) {

String strStatus = "true";
boolean status = Boolean.parseBoolean(strStatus);
System.out.println(status);
        }

}
```

Output:

true

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

Code:

```
public class P3 {

        public static void main(String[] args) {

String strStatus = "1";
boolean status = Boolean.parseBoolean(strStatus);
System.out.println(status);
        }

}
```

```
Output:
 false
```

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

Code:

```
public class p5 {

        public static void main(String[] args) {
                boolean status = true;
                boolean wrapperStatus =Boolean.valueOf(status);
                System.out.println("the boolean value is : " +wrapperStatus );
        }
}
```

Output:

```
the boolean value is : true
```

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

Code:

```
public class p6 {

	public static void main(String[] args) {
		String strStatus = "true";
		boolean wrapperStatus = Boolean.valueOf(strStatus);
		System.out.println("the boolean value is: " +wrapperStatus);

	}

}
```

Output:

```
the boolean value is: true
```

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

Code:

```
public class p7 {

	public static void main(String[] args) {
		// Boolean to int
		 boolean boolValue1 = true;
		 int intValue = boolValue1 ? 1 : 0;
		 System.out.println("Boolean to int: " + intValue);

		 // Boolean to double
		 boolean boolValue2 = false;
		 double doubleValue = boolValue2 ? 1.0 : 0.0;
		 System.out.println("Boolean to double: " + doubleValue);

		 // int to boolean
		 int intInput = 7;
		 boolean boolFromInt = intInput != 0;
		 System.out.println("Int to boolean: " + boolFromInt);

		 // double to boolean
		 double doubleInput = 3.14;
		 boolean boolFromDouble = doubleInput != 0.0;
		 System.out.println("Double to boolean: " + boolFromDouble);
	}

}
```

Output:

```
Boolean to int: 1
Boolean to double: 0.0
Int to boolean: true
Double to boolean: true
```

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

a. Explore the [Java API documentation for `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`).

code:

```java
public class Q2 {

	public static void main(String[] args) {
		int numberOfBytes = Byte.BYTES;
		System.out.println("byte value is:"+numberOfBytes);

	}

}
```

Output:

```
byte value is:1
```

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`).

Code:

```java
	public class Q3 {

		public static void main(String[] args) {
		    // Get the minimum and maximum values of byte
		    byte minValue = Byte.MIN_VALUE;
		    byte maxValue = Byte.MAX_VALUE;

		    // Print the minimum and maximum values
		    System.out.println("Minimum value of byte: " +
	minValue);
		    System.out.println("Maximum value of byte: " +
	maxValue);
		}

	}
```

Output:

```
	Minimum value of byte: -128
```

```
Maximum value of byte: 127
```

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)`).

Code:

```
public class Q4 {

          public static void main(String[] args) {
              // Declare a method-local variable of type byte and assign a
value
              byte number = 42;

              // Convert the byte variable to a String using
Byte.toString(byte)
              String numberAsString = Byte.toString(number);

              // Print the result
              System.out.println("The byte value as a String is: " +
numberAsString);
          }

}
```

Output:

```
The byte value as a String is: 42
```

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)`).

Code:

```
public class Q8 {
       public static void main(String [] args) {
              String strNumber = "60";
              byte number = Byte.parseByte(strNumber);
              System.out.println("the string value as a byte is: "+number);
       }

}
```

```
Output:
the string value as a byte is: 60
```

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`).

Code:

```
public class Q9 {
public static void main(String []args)
throws NumberFormatException{
        String strNumber = "Ab12Cd3";
        byte number =Byte.parseByte(strNumber);
        System.out.println("the string value as a byte is: " +number);
}
}
```

Output:

```
Exception in thread "main" java.lang.NumberFormatException: For input
string: "Ab12Cd3"
        at
java.base/java.lang.NumberFormatException.forInputString(NumberFormatExcept
ion.java:67)
        at java.base/java.lang.Integer.parseInt(Integer.java:662)
        at java.base/java.lang.Byte.parseByte(Byte.java:195)
        at java.base/java.lang.Byte.parseByte(Byte.java:221)
        at Q9.main(Q9.java:6)
```

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)`).

Code:

```
public class Q10 {
public static void main(String[] args)
{
        byte number =20;
        Byte byteWrapper = Byte.valueOf(number);
        System.out.println("Byte wrapper object is: "+byteWrapper);
}
}
```

```
Output:
Byte wrapper object is: 20
```

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)`).

Code:

```
public class Q11 {

        public static void main(String[] args) {
                String strNumber = "90";
                Byte byteWrapper = Byte.valueOf(strNumber);
                System.out.println("the string value is: "+byteWrapper);

        }
```

```
}
```

Output:

```
the string value is: 85
```

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

Code:

```
public static void main(String[] args) {
                int intValue = 10;
                short shortValue = 10;
                long longValue = 10L;
                float floatValue = 10.0f;
                double doubleValue = 10.0;
                char charValue = 'A';

                // Convert int to byte
                byte byteFromInt = (byte) intValue;
                System.out.println("Int to byte: " + byteFromInt); //
Output: Int to byte: 10

                // Convert short to byte
                byte byteFromShort = (byte) shortValue;
                System.out.println("Short to byte: " + byteFromShort);
// Output: Short to byte: 10

                // Convert long to byte
                byte byteFromLong = (byte) longValue;
                System.out.println("Long to byte: " + byteFromLong);
// Output: Long to byte: 10

                // Convert float to byte
                byte byteFromFloat = (byte) floatValue;
                System.out.println("Float to byte: " + byteFromFloat);
// Output: Float to byte: 10

                // Convert double to byte
                byte byteFromDouble = (byte) doubleValue;
                System.out.println("Double to byte: " +
byteFromDouble); // Output: Double to byte: 10

                // Convert char to byte
                byte byteFromChar = (byte) charValue;
                System.out.println("Char to byte: " + byteFromChar);
// Output: Char to byte: 65
            }
        }
```

Output:

```
Int to byte: 10
Short to byte: 10
Long to byte: 10
Float to byte: 10
Double to byte: 10
Char to byte: 65
```

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

a. Explore the Java API documentation for `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`).

code:

```
public class c2 {

	public static void main(String[] args) {
		int bytesUsed = Short.BYTES;
		System.out.println("the number of bytes used to represent a
short value is: " +bytesUsed);
	}

}
```

Output:

```
the number of bytes used to represent a short value is: 2
```

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

Code:

```
public class c3 {

	public static void main(String[] args) {
	short minValue = Short.MIN_VALUE;
	short maxValue = Short.MAX_VALUE;

	 System.out.println("the minimum value of short is: "+minValue);
	 System.out.println("the maximum value of short is: "+maxValue);
	}

}
```

Output:

```
the minimum value of short is: -32768
the maximum value of short is: 32767
```

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

Code:

```
public class c4 {

	public static void main(String[] args) {
		short number =12345;
		String numberAsString = Short.toString(number);
		System.out.println("the short value as a String is:
"+numberAsString);

	}

}
```

Output:

```
the short value as a String is: 12345
```

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)`).

Code:

```
public class c5 {

	public static void main(String[] args) {
		String strNumber= "5678";
		short number = Short.parseShort(strNumber);
		System.out.println("the short value is:" +number);

	}

}
```

Output:

```
the short value is:5678
```

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`).

Code:

```
	public class c6 {

		public static void main(String[] args) {
			String strNumber = "Ap23d1";
			try
			{
				short number = Short.parseShort(strNumber);
				System.out.println("the short value is: "
	+number);
			}
```

```
                    catch(NumberFormatException e){
                          System.out.println("NumberFormatException: "
        +e.getMessage());
                    }
              }

        }
```

Output:

```
NumberFormatException: For input string: "Ap23d1"
```

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)`).

Code:

```
public class c7 {

        public static void main(String[] args) {
              short number =6789;
              Short wrapperNumber = Short.valueOf(number);
              System.out.println(" the short object is: "+wrapperNumber);
        }

}
```

```
Output:
the short object is: 6789
```

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)`).

Code:

```
public class c8 {

        public static void main(String[] args) {


                        String strNumber = "123";
                        Short numberWrapper = Short.valueOf(strNumber);

                        System.out.println("Wrapper Short value: " +
numberWrapper);
                  }

        }
Output:
Wrapper Short value: 123
```

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

Code:

```
public class c9 {

	public static void main(String[] args) {
		int a=32;
		byte b= 45;
		float f=78.04f;
		double d=66488.68d;
		long l = 8925256;
		char c ='A';

		System.out.println(" int to short: " +(short)a);
		System.out.println(" byte to short: "
+Short.valueOf(b));
		System.out.println(" float to short: " +(short)f);
		System.out.println(" double to short: " +(short)d);
		System.out.println(" long to short: " +(short)l);
		System.out.println(" char to short: " +(short)c);
	}

}
```

Output:

```
int to short: 32
 byte to short: 45
 float to short: 78
 double to short: 952
 long to short: 12360
 char to short: 65
```

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

a. Explore the [Java API documentation for `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`).

Code:

```
public class Q4B {

	public static void main(String[] args) {

		Integer a =127;
```

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

        }

}
Output:
Integer size 4
```

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`).

Code:

```
        public class Q4C {

            public static void main(String[] args) {
                System.out.println("Integer Min Value:
"+Integer.MIN_VALUE);
                System.out.println("Integer Max Value:
"+Integer.MAX_VALUE);
            }
        }
        Output:
        Integer Min Value: -2147483648
        Integer Max Value: 2147483647
```

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

Code:

```
public class Q4D {

        public static void main(String[] args) {
             int b =567;
                System.out.println("Integer to string via
boxing:"+Integer.toString(b));

        }

}
Output:
Integer to string via boxing:567
```

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)`).

Code:

```
        public class Q4E {
        public static void main(String[] args)
        {
            String strStatus="676765";
```

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

        }
        }
```

Output:

```
String to int via Unboxing :676765
```

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`).

Code:

```
public class Q4F {

        public static void main(String[] args) {

            String numberStr = "12345";
          int number = Integer.parseInt(numberStr);
          System.out.println("The number is: " + number);


        }

}
```

```
Output:
The number is: 12345
```

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)`).

Code:

```
        public class Q4G {

            public static void main(String[] args) {
                int a = 12345;

                System.out.println("Integer value
        is:"+Integer.valueOf(a));

            }

        }
Output:
Integer value is:12345
```

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

Code:

```
public class Q4H {

	public static void main(String[] args) {

		String strNumber="7556";
		System.out.println("String to
int:"+Integer.valueOf(strNumber));

	}

}
Output:
String to int:7556
```

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)`).
Code:

```
public class Q4I {

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

	}
	Output:
	addition : 30
```

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

Code:

```
public class Q4J {

	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));
	}

}
```

Output:

```
Min value :10
Max value :20
```

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

Code:

```
public class Q4K {
	public static void main(String[] args) {
		int a = 7;
		System.out.println("Binary
:"+Integer.toBinaryString(a));
		System.out.println("Octal :"+Integer.toOctalString(a));
		System.out.println("Hex :"+Integer.toHexString(a));
	}

}
```

Output:

```
Binary :111
Octal :7
Hex :7
```

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

Code:

```
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));
```

Output:

```
Int 165
Integer to byte via narrowing :-91
```

```
¥
float 165.0
Double 165.0
Long 165
```

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

a. Explore the [Java API documentation for `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`).

CODE:

```
public class Q5B {

	public static void main(String[] args) {

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

}
```

OUTPUT:

```
Long size : 8
```

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

Code:

```
public class Q5C {

	public static void main(String[] args) {
		System.out.println("Min range of Long :"+Long.MIN_VALUE);
		System.out.println("Min range of Long :"+Long.MAX_VALUE);


	}

}
```

Output:

```
Min range of Long :-9223372036854775808
Min range of Long :9223372036854775807
```

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

Code:

```
 public class Q5D {

public static void main(String[] args) {
        long a=67453 ;
              System.out.println("Long to string via Long.toString(a));

       }

}
```

Output:

```
Long to string via boxing:67453
```

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)`).

CODE:

```
public class Q5E {

       public static void main(String[] args) {
              String strStatus="567432";
              System.out.println("String to long
:"+Long.parseLong(strStatus));


       }

}
```

```
Output:
String to long :567432
```

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`).

Code:

```
public class Q5F {

       public static void main(String[] args) {
              String strNumber="Ab12Cd3";
              System.out.println(" string  "+Long.parseLong(strNumber));
```

```
        }

}
```

Output:

```
Exception in thread "main" java.lang.NumberFormatException: For input
string: "Ab12Cd3"
        at
java.base/java.lang.NumberFormatException.forInputString(NumberFormatExcept
ion.java:67)
        at java.base/java.lang.Long.parseLong(Long.java:709)
        at java.base/java.lang.Long.parseLong(Long.java:832)
        at Q5F.main(Q5F.java:6)
```

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)`).

Code:

```
public class Q5G {

        public static void main(String[] args) {
                long c = 345384987;
                System.out.println("value is: "+Long.valueOf(c));


        }

}
```

Output:

```
value is: 345384987
```

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)`).

Code:

```
public class Q5H {

        public static void main(String[] args) {

                        String stftatus1="735224";
                        System.out.println("String is
:"+Long.valueOf(stftatus1));


        }
```

```
}
```

Output:

```
String is :735224
```

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)`).

Code:

```
public class Q5I {

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


	}

}
```

Output:

```
Sum :10968
```

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

Code:

```
public class Q5J {

	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));


	}

}
```

Output:

```
	Min value :1123
	Max value :9845
```

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)`).

Code:

```
public class Q5K {

	public static void main(String[] args) {
		long a = 2341;
		System.out.println("Binary :"+Long.toBinaryString(a));
		System.out.println("Octal :"+Long.toOctalString(a));
		System.out.println("Hex :"+Long.toHexString(a));

	}

}
```

Output:

```
Binary :100100100101
Octal :4445
Hex :925
```

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

Code:

```
public class Q5L {

	public static void main(String[] args) {
		long c=2351356;
		byte b=(byte)c;
		System.out.println("Int: "+Long.valueOf(c));
		System.out.println("long to byte: "+Byte.valueOf(b));
		System.out.println("float: "+Float.valueOf(c));
		System.out.println("Double "+Double.valueOf(c));
		System.out.println("Long "+Long.valueOf(c));

	}

}
```

Output:

```
Int: 2351356
long to byte: -4
float: 2351356.0
Double 2351356.0
Long 2351356
```

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

a. Explore the Java API documentation for `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`).

code:

```
public class Q6B {

	public static void main(String[] args) {
		System.out.println("Float size "+Float.BYTES);

	}

}
```

Output:

```
Float size 4
```

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

Code:

```
public class Q6C {

	public static void main(String[] args) {
		System.out.println("Min range of Float :"+Float.MIN_VALUE);
		System.out.println("Min range of Float :"+Float.MAX_VALUE);


	}

}
Output:
Min range of Float :1.4E-45
Min range of Float :3.4028235E38
```

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

CODE:

```
public class Q6D {

	public static void main(String[] args) {
		float b = 5238.02f;
		System.out.println("Floats to string :"+Float.toString(b));

	}

}
```

Output:

```
Floats to string boxinf:5238.02
```

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)`).

Code:

```
public class Q6E {

	public static void main(String[] args) {
		String strStatus="1274488788967.235";
		System.out.println("String to Float
:"+Float.parseFloat(strStatus));


	}

}
```

Output:

```
String to Float :1.2744888E12
```

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`).

Code:

```
public class Q6F {

	public static void main(String[] args) {
		String strNumber="Ab12Cd3";
		System.out.println(" string into Float:
"+Float.parseFloat(strNumber));


	}

}
```

Output:

```
Exception in thread "main" java.lang.NumberFormatException: For input
string: "Ab12Cd3"
	at
java.base/jdk.internal.math.FloatingDecimal.readJavaFormatString(FloatingDe
cimal.java:2054)
	at
java.base/jdk.internal.math.FloatingDecimal.parseFloat(FloatingDecimal.java
:122)
```

```
        at java.base/java.lang.Float.parseFloat(Float.java:556)
        at Q6F.main(Q6F.java:6)
```

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)`).

Code:

```
public class Q6G {

        public static void main(String[] args) {
                String stftatus1="6435.02f";
                System.out.println("String to Float
:"+Float.valueOf(stftatus1));


        }

}
```

```
Output:
String to Float  :6435.02
```

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)`).

Code:

```
public class Q6H {

        public static void main(String[] args) {
                String strNumber = "456652";
                System.out.println("String to float: "
+Float.valueOf(strNumber));
        }

}
```

Output:

```
String to float: 456652.0
```

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

```
public class Q6I {

        public static void main(String[] args) {
                float num1 = 224.4f;
```

```
            float num2 = 555.5f;
            System.out.println("sum of two float number :"+Float.sum(num1,
num2));

      }

}
Output:
sum of two float number :779.9
```

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)`).

Code:

```
public class Q6J {

      public static void main(String[] args) {
            float num1 = 121.5f;
            float num2 = 555.5f;
            System.out.println("Min value of two float number
"+Float.min(num1, num2));
            System.out.println("Max value of two float number
"+Float.max(num1, num2));

      }

}
Output:
Min value of two float number 121.5
Max value of two float number 555.5
```

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

Code:

```
public class Q6K {

      public static void main(String[] args) {
            float num=-25.0f;
            System.out.println("Sqaure root of negative number
:"+Math.sqrt(num));


      }

}
```

Output:

```
Sqaure root of negative number :NaN
```

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

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

Code:

```
public class Q6L {

	public static void main(String[] args) {
		float num1=-0.0f,num2=0.0f;
		System.out.println("Dividing the two number :"+(num1/num2));


	}

}
```

Output:

```
Dividing the two number :NaN
```

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

a. Explore the [Java API documentation for `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`).

code:

```
public class Q7B {

	public static void main(String[] args) {
		System.out.println("Double size :"+Double.BYTES);

	}

}
```

output:

```
Double size :8
```

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

Code:

```
public class Q7C {

	public static void main(String[] args) {
		System.out.println("Min range of Double :"+Double.MIN_VALUE);
		System.out.println("Min range of Double :"+Double.MAX_VALUE);

	}

}
```

Output:

```
Min range of Double :4.9E-324
Min range of Double :1.7976931348623157E308
```

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)`).

Code:

```
public class Q7D {

	public static void main(String[] args) {
		double b = 12344.11d;
		System.out.println("Double to string :"+Double.toString(b));


	}

}
```

Output:

```
Double to string :12344.11
```

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

Code:

```
public class Q7E {

	public static void main(String[] args) {
		String strStatus="23144";
		System.out.println("String to Double
:"+Double.parseDouble(strStatus));

	}

}
Output:
String to Double :23144.0
```

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`).

Code:

```
public class Q7F {

	public static void main(String[] args) {
		String strNumber="Ab12Cd3";
		System.out.println("string to Double
"+Double.parseDouble(strNumber));

	}

}
```

Output:

```
Exception in thread "main" java.lang.NumberFormatException: For input
string: "Ab12Cd3"
	at
java.base/jdk.internal.math.FloatingDecimal.readJavaFormatString(FloatingDe
cimal.java:2054)
	at
java.base/jdk.internal.math.FloatingDecimal.parseDouble(FloatingDecimal.jav
a:110)
	at java.base/java.lang.Double.parseDouble(Double.java:792)
	at Q7F.main(Q7F.java:6)
```

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)`).

Code:

```
public class Q7G {

	public static void main(String[] args) {
		double c = 1235.03d;
		System.out.println("Coverting doublr DT to Double wrapper
class :"+Double.valueOf(c));


	}

}
```

Output:

```
Coverting doublr DT to Double wrapper class :1235.03
```

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)`).

Code:

```
public class Q7H {

	public static void main(String[] args) {
		String strStatus="12731214.33d";
		System.out.println("String to Double
:"+Double.parseDouble(strStatus));

	}

}
```

Output:

```
String to Double :1.273121433E7
```

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)`).

Code:

```
public class Q7I {

	public static void main(String[] args) {
		double num1 = 112.3;
		double num2 = 984.5;
		System.out.println("Adition of two double number
"+Double.sum(num1, num2));

	}

}
```

Output:

```
Adition of two double number 1096.8
```

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

Code:

```
public class Q7J {

	public static void main(String[] args) {
		double num1 = 112.3;
		double num2 = 984.5;
```

```
            System.out.println("Min value of two double number
"+Double.min(num1, num2));
            System.out.println("Max value of two double number
"+Double.max(num1, num2));

      }

}
```

Output:

```
Min value of two double number 112.3
Max value of two double number 984.5
```

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

Code:

```
      public class Q7K {

            public static void main(String[] args) {
                  double num = -25.0;
                  System.out.println("square root of negative numbre:
      "+Math.sqrt(num));

            }

      }
```

Output:

```
square root of negative numbre: NaN
```

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

Code:

```
      public class Q7L {

            public static void main(String[] args) {
                  double num4=-0.0,num5=0.0;
                  System.out.println("Dividing the two number
      :"+(num5/num4));

            }

      }
```

Output:

```
Dividing the two number :NaN
```

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

Code:

```
public class Q7M {

	public static void main(String[] args) {
		double d =1245722321.034d;
		//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);

	}
}
```

Output:

```
Double to byte via narrowing :-47
Double to Char via narrowing :娶
Double to short via narrowing :14033
Double to int via narrowing  :1245722321
Double to float via narrowing :1.2457224E9
Double to long via narrowing :1245722321
```

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()`).

Code:

```
public class Q8A {

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

```java
            int intValue = 42;
        double doubleValue = 3.14;
        char charValue = 'A';
        long longValue = 123456789L;
        float floatValue = 1.23f;
        short shortValue = 123;
        byte byteValue = 127;

        String intString1 = Integer.toString(intValue);
        String doubleString1 = Double.toString(doubleValue);
        String charString1 = Character.toString(charValue);
        String longString1 = Long.toString(longValue);
        String floatString1 = Float.toString(floatValue);
        String shortString1 = Short.toString(shortValue);
        String byteString1 = Byte.toString(byteValue);

        String intString2 = String.valueOf(intValue);
        String doubleString2 = String.valueOf(doubleValue);
        String charString2 = String.valueOf(charValue);
        String longString2 = String.valueOf(longValue);
        String floatString2 = String.valueOf(floatValue);
        String shortString2 = String.valueOf(shortValue);
        String byteString2 = String.valueOf(byteValue);


        System.out.println("int: " + intString1);
        System.out.println("double: " + doubleString1);
        System.out.println("char: " + charString1);
        System.out.println("long: " + longString1);
        System.out.println("float: " + floatString1);
        System.out.println("short: " + shortString1);
        System.out.println("byte: " + byteString1);

        System.out.println("int: " + intString2);
        System.out.println("double: " + doubleString2);
        System.out.println("char: " + charString2);
        System.out.println("long: " + longString2);
        System.out.println("float: " + floatString2);
        System.out.println("short: " + shortString2);
        System.out.println("byte: " + byteString2);
      }

}
```

Output:

```
int: 42
double: 3.14
char: A
long: 123456789
float: 1.23
short: 123
byte: 127
int: 42
double: 3.14
char: A
long: 123456789
float: 1.23
```

```
short: 123
byte: 127
```

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).

Code:

```
public class DefaultValue {
static boolean a;
static char c;
static byte b;
static short s;
static int i;
static long l;
static float f ;
static double d;

	public static void main(String[] args) {
		System.out.println("Default value of boolean: " +a);
		System.out.println("Default value of char: " +c);
		System.out.println("Default value of byte: " +b);
		System.out.println("Default value of short: " +s);
		System.out.println("Default value of int: " +i);
		System.out.println("Default value of long: " +l);
		System.out.println("Default value of float: " +f);
		System.out.println("Default value of double: " +d);

	}

}
```

Output:

```
Default value of boolean: false
Default value of char:
Default value of byte: 0
Default value of short: 0
Default value of int: 0
Default value of long: 0
Default value of float: 0.0
Default value of double: 0.0
```

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).

Code:

```
import java.util.Scanner;
public class ArithmaticOperation {

       public static void main(String[] args) {

           char operator;
           int number1, number2, result;

           // create an object of Scanner class
           Scanner input = new Scanner(System.in);

           // ask users to enter operator
           System.out.println("Choose an operator: +, -, *, or /");
           operator = input.next().charAt(0);

           // ask users to enter numbers
           System.out.println("Enter first number");
           number1 = input.nextInt();

           System.out.println("Enter second number");
           number2 = input.nextInt();

           switch (operator) {

             // performs addition between numbers
             case '+':
               result = number1 + number2;
               System.out.println(number1 + " + " + number2 + " = " +
result);
               break;

             // performs subtraction between numbers
             case '-':
               result = number1 - number2;
               System.out.println(number1 + " - " + number2 + " = " +
result);
               break;

             // performs multiplication between numbers
             case '*':
               result = number1 * number2;
               System.out.println(number1 + " * " + number2 + " = " +
result);
               break;

             // performs division between numbers
             case '/':
               result = number1 / number2;
               System.out.println(number1 + " / " + number2 + " = " +
result);
               break;

             default:
               System.out.println("Invalid operator!");
               break;
           }
```

```
            input.close();
        }



    }
```

Output:

```
 Enter first number
4
Enter second number
6
4 * 6 = 24
```