

ASSIGNMENT NO.2

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)`).

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named `Demo.java`. The code defines a `public class Demo` with a `main` method. Inside the `main` method, a `boolean` variable `status` is assigned `true`, and a `String` variable `strStatus` is assigned the result of `Boolean.toString(status)`. The output window shows the command `javac Demo.java` and the output `true`.

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         boolean status = true;
5         String strStatus = Boolean.toString(status);
6         System.out.println(strStatus);
7     }
8 }
```

- 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)`).

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named `Demo.java`. The code defines a `public class Demo` with a `main` method. Inside the `main` method, a `String` variable `strStatus` is assigned the value "true", and a `boolean` variable `status` is assigned the result of `Boolean.parseBoolean(strStatus)`. The output window shows the command `javac Demo.java` and the output `true`.

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         String strStatus = "true";
5         boolean status = Boolean.parseBoolean(strStatus);
6         System.out.println(status);
7     }
8 }
```

- 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").

ASSIGNMENT NO.2

A screenshot of a Java IDE interface titled "Assignment 2". The left pane shows a file named "Demo.java" with the following code:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         String strStatus1 = "1";
6         String strStatus0 = "0";
7         boolean status1 = Boolean.parseBoolean(strStatus1);
8         boolean status0 = Boolean.parseBoolean(strStatus0);
9         System.out.println(status1);
10        System.out.println(status0);
11    }
12 }
```

The right pane shows the terminal output:

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
false
false
PS D:\Module 2\Day 2\Assignment 2>
```

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)`).

A screenshot of a Java IDE interface titled "Assignment 2". The left pane shows a file named "Demo.java" with the following code:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         boolean status = true;
6         boolean boolStatus = Boolean.valueOf(status);
7         System.out.println(boolStatus);
8     }
8 }
```

The right pane shows the terminal output:

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
true
PS D:\Module 2\Day 2\Assignment 2>
```

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)`).

A screenshot of a Java IDE interface titled "Assignment 2". The left pane shows a file named "Demo.java" with the following code:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         String strStatus = "true";
6         boolean boolStatus = Boolean.valueOf(strStatus);
7         System.out.println(boolStatus);
8     }
8 }
```

The right pane shows the terminal output:

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
true
PS D:\Module 2\Day 2\Assignment 2>
```

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

A screenshot of a Java IDE interface titled "Assignment 2". The left pane shows a file named "Demo.java" with the following code:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         boolean boolStatus = true;
6         int num = boolStatus ? 1 : 0;
7         System.out.println(num);
8
9         boolean Status = false;
10        int n = Status ? 1 : 0;
11        System.out.println(n);
12    }
12 }
```

The right pane shows the terminal output:

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
1
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
0
PS D:\Module 2\Day 2\Assignment 2>
```

ASSIGNMENT NO.2

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`).

Demo.java

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         byte b = Byte.BYTES;
5         System.out.println(b);
6     }
7 }
```

Assignment 2

PS D:\Module 2\Day 2\Assignment > javac Demo.java
PS D:\Module 2\Day 2\Assignment > java Demo
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`).

Demo.java

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         byte minValue = Byte.MIN_VALUE;
5         byte maxValue = Byte.MAX_VALUE;
6
7         System.out.println("Minimum value of byte is :" + minValue);
8         System.out.println("Maximum value of byte is :" + maxValue);
9     }
10 }
```

Assignment 2

PS D:\Module 2\Day 2\Assignment > javac Demo.java
PS D:\Module 2\Day 2\Assignment > java Demo
Minimum value of byte is :-128
Maximum value of byte is :127
PS D:\Module 2\Day 2\Assignment >

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

Demo.java

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         byte num = 6;
5         String strnum = Byte.toString(num);
6
7         System.out.println(strnum);
8     }
9 }
```

Assignment 2

PS D:\Module 2\Day 2\Assignment > javac Demo.java
PS D:\Module 2\Day 2\Assignment > java Demo
6
PS D:\Module 2\Day 2\Assignment >

- 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)`).

ASSIGNMENT NO.2

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         String strnum = "50";
6         byte b = Byte.parseByte(strnum);
7         System.out.println(b);
8     }
9 }
```

The terminal window on the right shows the command `javac Demo.java` being run, followed by the output `PS D:\Module 2\Day 2\Assignment 2> java Demo` and the result `50`.

- 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)`).

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         byte number = 3;
6         byte n = Byte.valueOf(number);
7         System.out.println(n);
8     }
9 }
```

The terminal window on the right shows the command `javac Demo.java` being run, followed by the output `PS D:\Module 2\Day 2\Assignment 2> java Demo` and the result `3`.

- 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)`).

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         String strNumber = "100";
6         Byte bytNum = Byte.valueOf(strNumber);
7         System.out.println(bytNum);
8     }
9 }
```

The terminal window on the right shows the command `javac Demo.java` being run, followed by the output `PS D:\Module 2\Day 2\Assignment 2> java Demo` and the result `100`.

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

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         byte num = 25;
6         int n = Integer.valueOf(num);
7         double d = Double.valueOf(num);
8
9         System.out.println(n);
10        System.out.println(d);
11    }
12 }
```

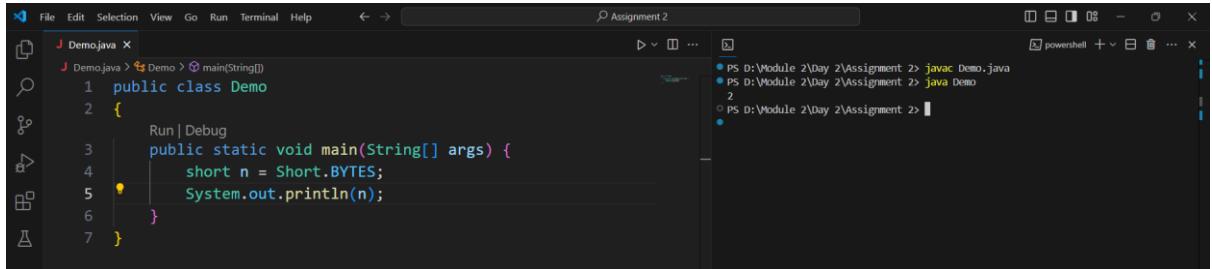
The terminal window on the right shows the command `javac Demo.java` being run, followed by the output `PS D:\Module 2\Day 2\Assignment 2> java Demo` and the result `25`.

ASSIGNMENT NO.2

3. Working with `java.lang.Short`

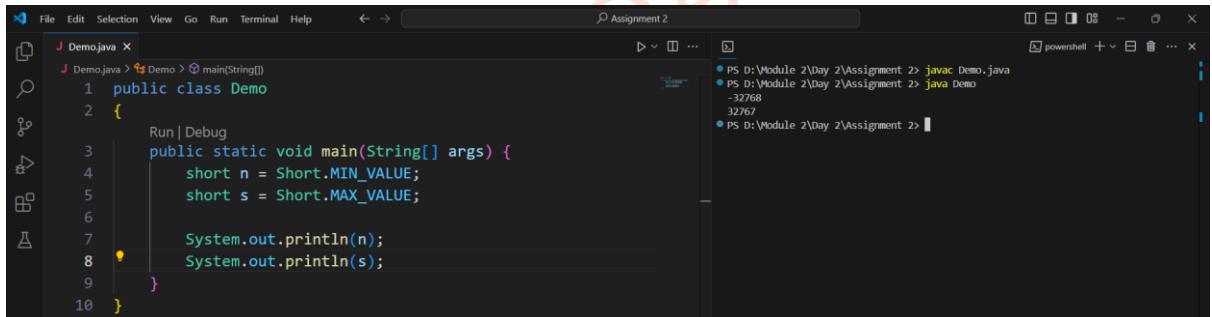
a. Explore the [Java API documentation for `java.lang.Short`](#) and observe its modifiers and super types.

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



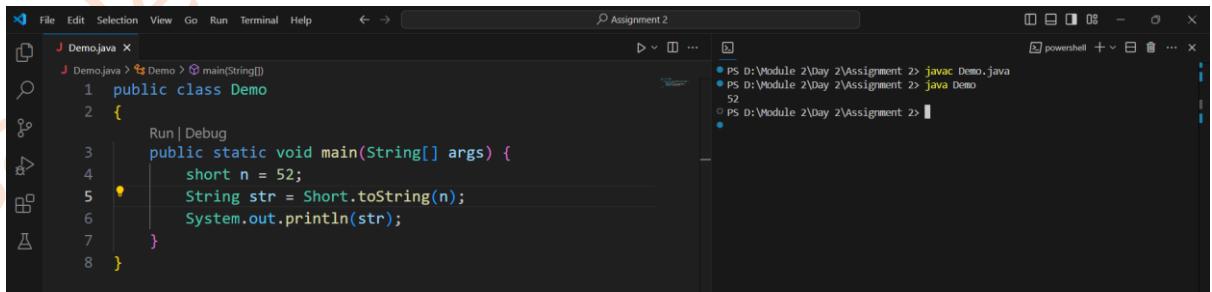
```
File Edit Selection View Go Run Terminal Help < > Assignment 2
Demo.java X
J Demo.java > Demo > main(String[])
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         short n = Short.BYTES;
6         System.out.println(n);
7     }
}
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
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`).



```
File Edit Selection View Go Run Terminal Help < > Assignment 2
Demo.java X
J Demo.java > Demo > main(String[])
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         short n = Short.MIN_VALUE;
6         short s = Short.MAX_VALUE;
7         System.out.println(n);
8         System.out.println(s);
9     }
10
}
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
-32768
32767
PS D:\Module 2\Day 2\Assignment 2>
```

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)`).



```
File Edit Selection View Go Run Terminal Help < > Assignment 2
Demo.java X
J Demo.java > Demo > main(String[])
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         short n = 52;
6         String str = Short.toString(n);
7         System.out.println(str);
8     }
}
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
52
PS D:\Module 2\Day 2\Assignment 2>
```

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)`).

ASSIGNMENT NO.2

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named Demo.java. The code contains a public class Demo with a main method that prints the value 50. The terminal window shows the command 'java Demo' being run and the output '50'.

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         String str = "50";
5         short s = Short.parseShort(str);
6         System.out.println(s);
7     }
8 }
```

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
50
PS D:\Module 2\Day 2\Assignment 2>
```

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)).

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named Demo.java. The code contains a public class Demo with a main method that converts the string "20" to a short value and prints it. The terminal window shows the command 'java Demo' being run and the output '20'.

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         short st = 20;
5         short s = Short.valueOf(st);
6         System.out.println(s);
7     }
8 }
```

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
20
PS D:\Module 2\Day 2\Assignment 2>
```

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)).

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named Demo.java. The code contains a public class Demo with a main method that converts the string "20000" to a short value and prints it. The terminal window shows the command 'java Demo' being run and the output '20000'.

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         String st = "20000";
5         short s = Short.valueOf(st);
6         System.out.println(s);
7     }
8 }
```

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
20000
PS D:\Module 2\Day 2\Assignment 2>
```

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

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named Demo.java. The code contains a public class Demo with a main method that converts a short value to an int and then to a double, printing each. The terminal window shows the command 'java Demo' being run and the output '3000', '3000', and '3000.0' respectively.

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         short s = 3000;
5         int n = Short.valueOf(s);
6         double d = Short.valueOf(s);
7         System.out.println(n);
8         System.out.println(d);
9     }
10 }
```

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
3000
3000
3000.0
PS D:\Module 2\Day 2\Assignment 2>
```

ASSIGNMENT NO.2

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`).

Demo.java

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         int s = Integer.BYTES;
5         System.out.println(s);
6     }
7 }
```

Assignment 2

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
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`).

Demo.java

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         int s = Integer.MIN_VALUE;
5         int v = Integer.MAX_VALUE;
6         System.out.println(s);
7         System.out.println(v);
8     }
9 }
```

Assignment 2

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
-2147483648
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)`).

Demo.java

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         int s = 500;
5         String ss = Integer.toString(s);
6         System.out.println(ss);
7     }
8 }
```

Assignment 2

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
500
```

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)`).

ASSIGNMENT NO.2

Demo.java

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         String num = "5200";
5         int ss = Integer.parseInt(num);
6         System.out.println(ss);
7     }
8 }
```

Assignment 2

PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
5200
PS D:\Module 2\Day 2\Assignment 2>

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)`).

Demo.java

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         int num = 10000;
5         int ss = Integer.valueOf(num);
6         System.out.println(ss);
7     }
8 }
```

Assignment 2

PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
10000
PS D:\Module 2\Day 2\Assignment 2>

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)`).

Demo.java

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         String num = "10000";
5         int ss = Integer.valueOf(num);
6         System.out.println(ss);
7     }
8 }
```

Assignment 2

PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
10000
PS D:\Module 2\Day 2\Assignment 2>

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)`).

Demo.java

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         int n1 = 10;
5         int n2 = 20;
6         int sum = Integer.sum(n1, n2);
7         System.out.println("Sum of n1 & n2 is : " + sum);
8     }
9 }
10 }
```

Assignment 2

PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
Sum of n1 & n2 is : 30
PS D:\Module 2\Day 2\Assignment 2>

ASSIGNMENT NO.2

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)`).

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named `Demo.java`. The code defines a `Demo` class with a `main` method. Inside the `main` method, two integers `n1` and `n2` are assigned values 40 and 20 respectively. Then, `Integer.min` and `Integer.max` methods are used to find the minimum and maximum values, which are then printed to the console. The terminal output shows the execution of the code and the resulting output.

```
public class Demo {
    public static void main(String[] args) {
        int n1 = 40;
        int n2 = 20;
        int min = Integer.min(n1, n2);
        int max = Integer.max(n1, n2);
        System.out.println("Minimun of n1 & n2 is : " + min);
        System.out.println("Minimun of n1 & n2 is : " + max);
    }
}
```

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
Minimum of n1 & n2 is : 20
Minimum of n1 & n2 is : 40
PS D:\Module 2\Day 2\Assignment 2>
```

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)`).

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named `Demo.java`. The code defines a `Demo` class with a `main` method. Inside the `main` method, an integer `n` is assigned the value 7. Then, `Integer.toBinaryString`, `Integer.toOctalString`, and `Integer.toHexString` methods are used to convert `n` to binary, octal, and hexadecimal strings respectively, which are then printed to the console. The terminal output shows the execution of the code and the resulting output.

```
public class Demo {
    public static void main(String[] args) {
        int n = 7;
        String s1 = Integer.toBinaryString(n);
        String s2 = Integer.toOctalString(n);
        String s3 = Integer.toHexString(n);

        System.out.println("Binary of 7 is : " + s1);
        System.out.println("Octal of 7 is : " + s2);
        System.out.println("Hexadecimal of 7 is : " + s3);
    }
}
```

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
Binary of 7 is : 111
Octal of 7 is : 7
Hexadecimal of 7 is : 7
PS D:\Module 2\Day 2\Assignment 2>
```

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

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named `Demo.java`. The code defines a `Demo` class with a `main` method. Inside the `main` method, an integer `n` is assigned the value 7. A `double` variable `d` is then assigned the value of `n` using the `Double.valueOf` method. Finally, the value of `d` is printed to the console. The terminal output shows the execution of the code and the resulting output.

```
public class Demo {
    public static void main(String[] args) {
        int n = 7;
        double d = Double.valueOf(n);
        System.out.println(d);
    }
}
```

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
7.0
PS D:\Module 2\Day 2\Assignment 2>
```

ASSIGNMENT NO.2

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`).

```
File Edit Selection View Go Run Terminal Help < > Assignment 2 powershell + ... x

J Demo.java x
J Demo.java > ⚡ Demo > main(String[])
1 public class Demo
2 {
3     Run | Debug
4         public static void main(String[] args) {
5             long d = Long.BYTES;
6             System.out.println(d);
7         }
8     }

PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
8
PS D:\Module 2\Day 2\Assignment 2> java Demo
```

- 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).

The screenshot shows a Java code editor with the following code:

```
public class Demo {
    public static void main(String[] args) {
        long d = Long.MIN_VALUE;
        long s = Long.MAX_VALUE;

        System.out.println(d);
        System.out.println(s);
    }
}
```

The code includes a breakpoint at line 9. The status bar at the bottom indicates the current line number as 9.

- 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)`).

The screenshot shows the Visual Studio Code interface. The left sidebar contains icons for file operations like Open, Save, Find, and Run. The main editor area displays a Java file named `Demo.java` with the following code:

```
public class Demo {  
    public static void main(String[] args) {  
        long s = 520000023;  
        String d = Long.toString(s);  
  
        System.out.println(d);  
    }  
}
```

The status bar at the bottom indicates the file is saved. The terminal tab is open, showing command history:

- PS D:\Module 2\Day 2\Assignment 2 > `javac Demo.java`
- PS D:\Module 2\Day 2\Assignment 2 > `java Demo`
- PS D:\Module 2\Day 2\Assignment 2 > `520000023`
- PS D:\Module 2\Day 2\Assignment 2 > `java Demo`
- PS D:\Module 2\Day 2\Assignment 2 > `520000023`

- 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)`).

ASSIGNMENT NO.2

The screenshot shows a Java development environment. The code editor displays the following Java code:

```
public class Demo {
    public static void main(String[] args) {
        String d = "205554";
        long l = Long.parseLong(d);
        System.out.println(l);
    }
}
```

The terminal window on the right shows the command `javac Demo.java` followed by the output `205554`.

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)`).

The screenshot shows a Java development environment. The code editor displays the following Java code:

```
public class Demo {
    public static void main(String[] args) {
        long l1 = 772015863;
        long l = Long.valueOf(l1);
        System.out.println(l);
    }
}
```

The terminal window on the right shows the command `javac Demo.java` followed by the output `772015863`.

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)`).

The screenshot shows a Java development environment. The code editor displays the following Java code:

```
public class Demo {
    public static void main(String[] args) {
        String l1 = "772015863";
        long l = Long.valueOf(l1);
        System.out.println(l);
    }
}
```

The terminal window on the right shows the command `javac Demo.java` followed by the output `772015863`.

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)`).

ASSIGNMENT NO.2

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5
6         long l1 = 1123;
7         long l2 = 9845;
8         long sum = Long.sum(l1, l2);
9
10        System.out.println(sum);
11    }
12}
```

The terminal window on the right shows the command `javac Demo.java` was run and completed successfully with exit code 0. It also shows the command `java Demo` was run, outputting the value 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)`).

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5
6         long l1 = 1122;
7         long l2 = 5566;
8         long min = Long.min(l1, l2);
9         long max = Long.max(l2, l1);
10
11        System.out.println("Minimum value : " + min);
12        System.out.println("Maximum value : " + max);
13    }
14}
```

The terminal window on the right shows the command `javac Demo.java` was run and completed successfully with exit code 0. It then shows the command `java Demo` was run, outputting the minimum value 1122 and the maximum value 5566.

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

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5
6         long l = 7;
7         String b = Long.toBinaryString(l);
8         String o = Long.toOctalString(l);
9         String h = Long.toHexString(l);
10
11        System.out.println("Binary value : " + b);
12        System.out.println("Octal value : " + o);
13        System.out.println("Hex value : " + h);
14    }
15}
```

The terminal window on the right shows the command `javac Demo.java` was run and completed successfully with exit code 0. It then shows the command `java Demo` was run, outputting the binary value 111, the octal value 7, and the hex value 7.

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

ASSIGNMENT NO.2

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         long l = 1234567890;
6         int n = (int)l;
7         System.out.println(n);
8     }
9 }
10 }
```

The terminal window on the right shows the command `javac Demo.java` was run and the output `1234567890`.

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`).

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         float f = Float.BYTES;
6         System.out.println(f);
7     }
8 }
9 }
```

The terminal window on the right shows the command `javac Demo.java` was run and the output `4.0`.

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`).

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         float f = Float.MIN_VALUE;
6         float g = Float.MAX_VALUE;
7
8         System.out.println(f);
9         System.out.println(g);
10    }
11 }
```

The terminal window on the right shows the command `javac Demo.java` was run and the output `1.4E-45` and `3.402823E38`.

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)`).

ASSIGNMENT NO.2

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         float f1 = 7.23f;
5         String f = Float.toString(f1);
6         System.out.println(f);
7     }
8 }
9 }
```

The terminal window on the right shows the command `javac Demo.java` was run and the output `7.23`.

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)`).

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         String f1 = "7.23f";
5         float f = Float.parseFloat(f1);
6         System.out.println(f);
7     }
8 }
9 }
```

The terminal window on the right shows the command `javac Demo.java` was run and the output `7.23`.

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)`).

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         float c = 42.036f;
5         float f = Float.valueOf(c);
6         System.out.println(f);
7     }
8 }
9 }
```

The terminal window on the right shows the command `javac Demo.java` was run and the output `42.036`.

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)`).

ASSIGNMENT NO.2

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named Demo.java. The code prints the string "42.036f" to the console. The terminal window shows the command "java Demo" and the output "42.036".

```
public class Demo {
    public static void main(String[] args) {
        String c = "42.036f";
        float f = Float.valueOf(c);
        System.out.println(f);
    }
}
```

- 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)`).

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named Demo.java. The code adds two float variables (112.3f and 984.5f) using the `Float.sum()` method and prints the result. The terminal window shows the command "java Demo" and the output "1096.8".

```
public class Demo {
    public static void main(String[] args) {
        float c = 112.3f;
        float d = 984.5f;
        float f = Float.sum(c, d);
        System.out.println(f);
    }
}
```

- 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)`).

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named Demo.java. The code finds the minimum and maximum of two float variables (112.2f and 556.6f) using the `Float.min()` and `Float.max()` methods and prints the results. The terminal window shows the command "java Demo" and the output "Minimum : 112.2 Maximum : 556.6".

```
public class Demo {
    public static void main(String[] args) {
        float c = 112.2f;
        float d = 556.6f;
        float min = Float.min(c, d);
        float max = Float.max(c, d);
        System.out.println("Minimum : " + min);
        System.out.println("Maximum : " + max);
    }
}
```

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

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named Demo.java. The code declares a float variable c with the value -25.0f, calculates its absolute value using `Math.abs()`, and then finds the square root using `Math.sqrt()`. The terminal window shows the command "java Demo" and the output "square root of -25.0f : 5.0".

```
public class Demo {
    public static void main(String[] args) {
        float c = -25.0f;
        double sqrt = Math.sqrt(Math.abs(c));
        System.out.println("square root of -25.0f : " + sqrt);
    }
}
```

ASSIGNMENT NO.2

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

The screenshot shows a Java development environment. The code editor displays the following Java code:

```
1 public class Demo {
2     public static void main(String[] args) {
3         float c = 0.0f;
4         float d = 0.0f;
5         float f = c / d;
6         System.out.println(f);
7     }
8 }
```

The terminal window on the right shows the command `javac Demo.java` followed by the output `NaN`.

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

The screenshot shows a Java development environment. The code editor displays the following Java code:

```
1 public class Demo {
2     public static void main(String[] args) {
3         float c = 8.50f;
4         int f = (int)c;
5         System.out.println(f);
6     }
7 }
```

The terminal window on the right shows the command `javac Demo.java` followed by the output `8`.

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`).

The screenshot shows a Java development environment. The code editor displays the following Java code:

```
1 public class Demo {
2     public static void main(String[] args) {
3         Double c = (double) Double.BYTES;
4         System.out.println(c);
5     }
6 }
```

The terminal window on the right shows the command `javac Demo.java` followed by the output `8.0`.

- 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`).

ASSIGNMENT NO.2

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named `Demo.java`. The code defines a `Demo` class with a `main` method. The `main` method prints the minimum and maximum values of `Double` to the console. The terminal window shows the command `java Demo` being run, and the output is `1.797693134862315E308`.

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         Double c = Double.MIN_VALUE;
5         Double d = Double.MAX_VALUE;
6
7         System.out.println(c);
8         System.out.println(d);
9     }
10 }
11 
```

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)`).

A screenshot of a Java IDE (IntelliJ IDEA) showing the same `Demo.java` file. In this version, a local variable `c` is declared as `Double c = 235.25;`. The `toString` method is used to convert `c` to a `String` and print it to the console. The terminal output shows the value `235.25`.

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         Double c = 235.25;
5         String d = Double.toString(c);
6
7         System.out.println(d);
8     }
9 }
10 
```

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)`).

A screenshot of a Java IDE (IntelliJ IDEA) showing the same `Demo.java` file. A local variable `c` is declared as a `String` with the value `"235.25"`. The `parseDouble` method is used to convert `c` to a `double` and print it to the console. The terminal output shows the value `235.25`.

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         String c = "235.25";
5         Double d = Double.parseDouble(c);
6
7         System.out.println(d);
8     }
9 }
10 
```

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)`).

ASSIGNMENT NO.2

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         Double c = 235.25;
6         Double d = Double.valueOf(c);
7
8         System.out.println(d);
9     }
10
11 }
```

The terminal window on the right shows the output of running the code:

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
235.25
PS D:\Module 2\Day 2\Assignment 2>
```

- 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)`).

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         String c = "235.25";
6         Double d = Double.valueOf(c);
7
8         System.out.println(d);
9     }
10
11 }
```

The terminal window on the right shows the output of running the code:

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
235.25
PS D:\Module 2\Day 2\Assignment 2>
```

- 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)`).

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         Double c = 112.3;
6         Double d = 984.5;
7         Double sum = Double.sum(c, d);
8
9         System.out.println(sum);
10
11 }
```

The terminal window on the right shows the output of running the code:

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
1096.8
PS D:\Module 2\Day 2\Assignment 2>
```

- 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)`).

A screenshot of a Java development environment. The code editor shows a file named `Demo.java` with the following content:

```
1 public class Demo
2 {
3     Run | Debug
4     public static void main(String[] args) {
5         Double c = 112.2;
6         Double d = 556.6;
7         Double min = Double.min(c, d);
8         Double max = Double.max(c, d);
9
10        System.out.println(min);
11        System.out.println(max);
12
13    }
14 }
```

The terminal window on the right shows the output of running the code:

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
112.2
556.6
PS D:\Module 2\Day 2\Assignment 2>
```

ASSIGNMENT NO.2

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

The screenshot shows a Java code editor with a file named `Demo.java`. The code contains a `public class Demo` with a `main` method. Inside the `main` method, a double variable `c` is assigned the value `-25.0`, and another double variable `d` is assigned the result of `Math.sqrt(Math.abs(c))`. Finally, `d` is printed to the console using `System.out.println(d)`. The terminal window shows the command `java Demo` being run, and the output is `5.0`.

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         Double c = -25.0;
5         Double d = Math.sqrt(Math.abs(c));
6         System.out.println(d);
7     }
8 }
```

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

The screenshot shows a Java code editor with a file named `Demo.java`. The code contains a `public class Demo` with a `main` method. Inside, two double variables `c` and `d` are both assigned the value `0.0`. Then, `v` is assigned the result of `c / d`. Finally, `v` is printed to the console using `System.out.println(v)`. The terminal window shows the command `java Demo` being run, and the output is `NaN`.

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         Double c = 0.0;
5         Double d = 0.0;
6         Double v = c / d;
7         System.out.println(v);
8     }
9 }
```

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

The screenshot shows a Java code editor with a file named `Demo.java`. The code contains a `public class Demo` with a `main` method. Inside, a double variable `c` is assigned the value `25.36`. An integer variable `v` is assigned the value of `c` using the cast operator `(int)c`. Finally, `v` is printed to the console using `System.out.println(v)`. The terminal window shows the command `java Demo` being run, and the output is `25`.

```
1 public class Demo
2 {
3     public static void main(String[] args) {
4         double c = 25.36;
5         int v = (int) c;
6         System.out.println(v);
7     }
8 }
```

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()`).

ASSIGNMENT NO.2

The screenshot shows a Java code editor with a file named Demo.java. The code demonstrates various ways to convert primitive types to strings:

```
public class Demo {
    public static void main(String[] args) {
        byte b = 100;
        short s = 20;
        int i = 3333;
        long l = 5566L;
        float f = 8.97f;
        double d = 19.5693;
        char c = 'N';
        boolean bool = true;

        String byteToString1 = Byte.toString(b);
        String shortToString1 = Short.toString(s);
        String intToString1 = Integer.toString(i);
        String longToString1 = Long.toString(l);
        String floatToString1 = Float.toString(f);
        String doubleToString1 = Double.toString(d);
        String charToString1 = Character.toString(c);
        String booleanToString1 = Boolean.toString(bool);

        String byteToString2 = String.valueOf(b);
        String shortToString2 = String.valueOf(s);
        String intToString2 = String.valueOf(i);
        String longToString2 = String.valueOf(l);
        String floatToString2 = String.valueOf(f);
        String doubleToString2 = String.valueOf(d);
        String charToString2 = String.valueOf(c);
    }
}
```

To the right of the code, the terminal window shows the output of running the program:

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
byte: 100
short: 20
int: 3333
long: 5566
float: 8.97
double: 19.5693
char: N
boolean: true

byte: 100
short: 20
int: 3333
long: 5566
float: 8.97
double: 19.5693
char: N
boolean: true

```

This screenshot shows the same Java code as above, but with additional System.out.println statements to print each converted string:

```
public class Demo {
    public static void main(String[] args) {
        String floatToString2 = String.valueOf(f);
        String doubleToString2 = String.valueOf(d);
        String charToString2 = String.valueOf(c);
        String booleanToString2 = String.valueOf(bool);

        System.out.println("byte: " + byteToString1);
        System.out.println("short: " + shortToString1);
        System.out.println("int: " + intToString1);
        System.out.println("long: " + longToString1);
        System.out.println("float: " + floatToString1);
        System.out.println("double: " + doubleToString1);
        System.out.println("char: " + charToString1);
        System.out.println("boolean: " + booleanToString1);
        System.out.println();
        System.out.println("byte: " + byteToString2);
        System.out.println("short: " + shortToString2);
        System.out.println("int: " + intToString2);
        System.out.println("long: " + longToString2);
        System.out.println("float: " + floatToString2);
        System.out.println("double: " + doubleToString2);
        System.out.println("char: " + charToString2);
        System.out.println("boolean: " + booleanToString2);
    }
}
```

The terminal output is identical to the one in the previous screenshot.

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).

ASSIGNMENT NO.2

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named Demo.java. The code prints various primitive data types: byte, short, int, long, float, double, char, and boolean. The output window shows the results of running the program.

```
public class Demo {
    public static void main(String[] args) {
        byte byteVar = Byte.BYTES;
        short shortVar=Short.BYTES;
        int intVar=Integer.BYTES;
        long longVar=Long.BYTES;
        float floatVar=Float.BYTES;
        double doubleVar=Double.BYTES;
        char charVar=Character.BYTES;
        boolean booleanVar=Boolean.FALSE;

        System.out.println("byte: " + byteVar);
        System.out.println("short: " + shortVar);
        System.out.println("int: " + intVar);
        System.out.println("long: " + longVar);
        System.out.println("float: " + floatVar);
        System.out.println("double: " + doubleVar);
        System.out.println("char: " + charVar );
        System.out.println("boolean: " + booleanVar);
    }
}
```

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
byte: 1
short: 2
int: 4
long: 8
float: 4.0
double: 8.0
char: @
boolean: false
```

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).

A screenshot of a Java IDE (IntelliJ IDEA) showing a file named Demo.java. The program reads two integers from the user and a choice of arithmetic operator (+, -, *, /). It then performs the operation and prints the result. The output window shows several runs of the program with different inputs.

```
public class Demo {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter num1 : ");
        int num1 = sc.nextInt();
        System.out.print("Enter num2 : ");
        int num2 = sc.nextInt();
        System.out.print("Enter your choice (+, -, *, /) : ");
        char choice = sc.next().charAt(index:0);
        switch (choice) {
            case '+':
                int sum = num1 + num2;
                System.out.println("Result of " + num1 + " + " + num2 + " : " + sum);
                break;
            case '-':
                int sub = num1 - num2;
                System.out.println("Result of " + num1 + " - " + num2 + " : " + sub);
                break;
            case '*':
                int multi = num1 * num2;
                System.out.println("Result of " + num1 + " * " + num2 + " : " + multi);
                break;
            case '/':
                int div = num1 / num2;
                System.out.println("Result of " + num1 + " / " + num2 + " : " + div);
                break;
            default:
                System.out.println("Invalid Input.....!");
                break;
        }
        sc.close();
    }
}
```

```
PS D:\Module 2\Day 2\Assignment 2> javac Demo.java
PS D:\Module 2\Day 2\Assignment 2> java Demo
Enter num1 : 20
Enter num2 : 40
Enter your choice (+, -, *, /) : +
Result of 20 + 40 : 60
PS D:\Module 2\Day 2\Assignment 2> java Demo
Enter num1 : 30
Enter num2 : 50
Enter your choice (+, -, *, /) : -
Result of 30 - 50 : -20
PS D:\Module 2\Day 2\Assignment 2> java Demo
Enter num1 : 30
Enter num2 : 10
Enter your choice (+, -, *, /) : *
Result of 30 * 10 : 300
PS D:\Module 2\Day 2\Assignment 2> java Demo
Enter num1 : 60
Enter num2 : 20
Enter your choice (+, -, *, /) : /
Result of 60 / 20 : 3
PS D:\Module 2\Day 2\Assignment 2>
```