OOPJ
Assignment-2

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

   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)).

```
boolean status = true;
String str = Boolean.toString(status);
System.out.println(str);
```

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)).

```
String strStatus = "true";
Boolean bool = Boolean.parseBoolean(strStatus);
System.out.println(bool);
```

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").

```
String strStatus = “1”;
Boolean bool = Boolean.parseBoolean(strStatus);
System.out.println(bool);
```

Output

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        Syntax error on tokens, delete these tokens
```

```
String strStatus = "true";
Boolean bool = Boolean.parseBoolean(strStatus);
System.out.println(bool);
```

Output: true

Note: “True” or “true” will return true.

Any other input (like “1”, ”0”, “yes”, “no”, “false”) will return 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)).

```
boolean status = true;
Boolean newStatus = Boolean.valueOf(status);
System.out.println(newStatus);
```

Output: 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)).

```
String strStatus = "true";
Boolean bool = Boolean.valueOf(strStatus);
System.out.println(bool);
```

Output: true

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

2. Working with java.lang.Byte

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

System.out.println(Byte.BYTES);

Output:- 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).

System.***out***.println(Byte.***MIN_VALUE***); —> Output: -127

System.***out***.println(Byte.***MAX_VALUE***); —> Output: 128

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)).

```
byte num = 51;
String n = Byte.toString(num);
System.out.println(n);
```

Output: 51

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)).

```
String strNumber = "108";
byte num = Byte.parseByte(strNumber);
System.out.println(num);
```

Output: 108

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).

```
String strNumber = "Ab12Cd3";
byte num = Byte.parseByte(strNumber);
System.out.println(num);
```

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3"
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
```

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)).

```
byte num = 21;
Byte n = Byte.valueOf(num);
System.out.println(n);
```

Output: 21

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)).

```
String strNumber = "47";
Byte num = Byte.valueOf(strNumber);
System.out.println(num);
```

Output: 47

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

```
byte a = 4;
int b = a;
System.out.println(b);
```

Output: 4

3. Working with java.lang.Short

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

System.out.println(Short.BYTES);

Output:- 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).

```
System.out.println(Short.MIN_VALUE);
System.out.println(Short.MAX_VALUE);
```

Output:

-32768
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)).

```
short num = 22251;
String n = Short.toString(num);
System.out.println(n);
```

Output: 22251

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)).

```
String strNumber = "10821";
short num = Short.parseShort(strNumber);
System.out.println(num);
```

Output: 10821

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).

```
String strNumber = "Ab12Cd3";
short num = Short.parseShort(strNumber);
System.out.println(num);
```

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3"
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
```

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)).

```
short num = 4521;
Short n = Short.valueOf(num);
System.out.println(n);
```

Output: 4521

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)).

```
String strNumber = "14730";
Short num = Short.valueOf(strNumber);
System.out.println(num);
```

Output: 14730

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

```
byte a = 4;
int b = a;
System.out.println(b);
```

Output: 4

4. Working with java.lang.Integer

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

System.out.println(Integer.BYTES);

Output:- 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).

```
System.out.println(Integer.MIN_VALUE);
System.out.println(Integer.MAX_VALUE);
```

Output:

-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)).

```
int num = 22251;
String n = Integer.toString(num);
System.out.println(n);
```

Output: 22251

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)).

```
String strNumber = "10821";
int num = Integer.parseInt(strNumber);
System.out.println(num);
```

Output: 10821

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).

```
String strNumber = "Ab12Cd3";
int num = Integer.parseInteger(strNumber);
System.out.println(num);
```

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3"
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
```

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)).

```
int num = 21;
Integer n = Integer.valueOf(num);
System.out.println(n);
```

Output: 21

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)).

```
String strNumber = "14730";
Integer num = Integer.valueOf(strNumber);
System.out.println(num);
```

Output: 14730

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)).

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter n1: ");
int n1 = sc.nextInt();
System.out.println("Enter n2: ");
int n2 = sc.nextInt();
int Sum = Integer.sum(n1, n2);
System.out.println("Sum = " + Sum);
sc.close();
```

Output:

```
Enter n1:
10
Enter n2:
20
Sum = 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)).

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter n1: ");
int n1 = sc.nextInt();
System.out.print("Enter n2: ");
int n2 = sc.nextInt();
int min = Integer.min(n1, n2);
System.out.println("Min = " + min);
int max = Integer.max(n1, n2);
System.out.println("Max = " + max);
sc.close();
```

Output:

```
Enter n1: 10
Enter n2: 20
Min = 10
Max = 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)).

```
int n = 7;
String bin = Integer.toBinaryString(n);
System.out.println("Binary = " + bin);
String oct = Integer.toOctalString(n);
System.out.println("Octal = " + oct);
String hex = Integer.toHexString(n);
System.out.println("Hexadecimal = " + hex);
```

Output:

```
Binary = 111
Octal = 7
Hexadecimal = 7
```

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

5. Working with java.lang.Long

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

System.out.println(Long.BYTES);

Output: 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).

```
System.out.println(Long.MIN_VALUE);
System.out.println(Long.MAX_VALUE);
```

Output:

```
-9223372036854775808
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)).

```
long num = 22251;
String n = Long.toString(num);
System.out.println(n);
```

Output: 22251

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)).

```
String strNumber = "10821";
long num = Long.parseLong(strNumber);
System.out.println(num);
```

Output: 10821

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).

```
String strNumber = "Ab12Cd3";
long num = Long.parseLong(strNumber);
System.out.println(num);
```

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3"
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
```

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)).

```
long num = 21;
Long n = Long.valueOf(num);
System.out.println(n);
```

Output: 21

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)).

```
String strNumber = "14730";
Long num = Long.valueOf(strNumber);
System.out.println(num);
```

Output: 14730

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)).

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter n1: ");
long n1 = sc.nextLong();
```

```
System.out.print("Enter n2: ");
long n2 = sc.nextLong();
long Sum = Long.sum(n1, n2);
System.out.println("Sum = " + Sum);
sc.close();
```

Output:

```
Enter n1: 1123
Enter n2: 9845
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)).

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter n1: ");
long n1 = sc.nextLong();
System.out.print("Enter n2: ");
long n2 = sc.nextLong();
long min = Long.min(n1, n2);
System.out.println("Min = " + min);
long max = Long.max(n1, n2);
System.out.println("Max = " + max);
sc.close();
```

Output:

```
Enter n1: 1122
Enter n2: 5566
Min = 1122
Max = 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)).

```
long n = 7;
String bin = Long.toBinaryString(n);
System.out.println("Binary = " + bin);
String oct = Long.toOctalString(n);
System.out.println("Octal = " + oct);
String hex = Long.toHexString(n);
System.out.println("Hexadecimal = " + hex);
```

Output:

```
Binary = 111
Octal = 7
Hexadecimal = 7
```

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

6. Working with java.lang.Float

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

System.out.println(Float.BYTES);

Output: 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).

```
System.out.println(Float.MIN_VALUE);
System.out.println(Float.MAX_VALUE);
```

Output:

```
1.4E-45
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)).

```
float num = 22251;
String n = Float.toString(num);
System.out.println(n);
```

Output: 22251.0

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)).

```
String strNumber = "10821";
float num = Float.parseFloat(strNumber);
System.out.println(num);
```

Output: 10821.0

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).

```
String strNumber = "Ab12Cd3";
float  num = Float.parseFloat(strNumber);
System.out.println(num);
```

Output:

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "Ab12Cd3"
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
```

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)).

```
float num = 21;
Float n = Float.valueOf(num);
System.out.println(n);
```

Output: 21.0

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)).

```
String strNumber = "14730";
Float num = Float.valueOf(strNumber);
System.out.println(num);
```

Output: 14730.0

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)).

```
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter n1: ");
```

```
        float n1 = sc.nextFloat();
        System.out.print("Enter n2: ");
        float n2 = sc.nextFloat();
        float Sum = Float.sum(n1, n2);
        System.out.println("Sum = " + Sum);
        sc.close();
```

Output:

```
Enter n1: 112.3
Enter n2: 984.5
Sum = 1096.8
```