Assignment 2

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)).

```
public class BoolToString1 {

    public static void main(String[] args){
        boolean status=true;
        String strStatus=Boolean.toString(status) ;
        System.out.println("String of bool is:"+strStatus);
    }
}
```

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

```
public class StringToBool {
    public static void main(String[] args) {
        String strStatus=new String("true");
        Boolean strbool=Boolean.parseBoolean(strStatus);
        System.out.println("String to bool unboxing:"+strbool);
    }

}
```

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

work as expected with "1" or "0").

```
public class Bool0or1 {
    public static void main(String[] args) {
        String strStatus="1";
        Boolean bool1=strStatus.equals("0");
      // Boolean bool1=Boolean.parseBoolean(strStatus);
        System.out.println("String to bool: "+bool1);
    }
}
```

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

```
public class eBool {
    public static void main(String[] args) {
        boolean status=true;
        Boolean bool1=Boolean.valueOf(status);
        System.out.println("Value of bool: "+bool1);

    }

}
```

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

```
public class fBoolean {
    public static void main(String[] args) {
        //unboxing
        String strStatus="true";
        Boolean bool=Boolean.valueOf(strStatus);
        System.out.println("bool value:"+bool);
    }
```

```
}
```

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

```
public class gBool {
    public static void main(String[] args) {
        boolean status=false;
        Integer num=Boolean.hashCode(status);
        System.out.println("Integer:"+num);


    }

}
```

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).

```
public class bByte {

    public static void main(String[] args) {
        int bytes=Byte.BYTES; //1
        //char bytes=Byte.BYTES;  //smile emoji
       // float bytes=Byte.BYTES; //1.0
       //double bytes=Byte.BYTES;  //1.0
       //long bytes=Byte.BYTES;     //1
       // byte bytes=Byte.BYTES;   //1
```

```
         System.out.println("Byte used to represent:"+bytes);
    }
}
```

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

```
public class cByteMinMax {
    public static void main(String[] args) {
       // int minVal=Byte.MIN_VALUE;
       // System.err.println("Min value: "+minVal);  //-128

       int maxVal=Byte.MAX_VALUE;
       System.out.println("Max value: "+maxVal);   //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)).

```
public class dByteToString {
    public static void main(String[] args) {
        byte number=100;
        String num=Byte.toString(number);
       // String num =Integer.toString(number);

        System.out.println("Byte to String: "+num);

    }
}
```

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

```
public class eStringToByte {
    public static void main(String[] args) {
        String strNumber="120";
        Byte byVal=Byte.parseByte(strNumber);
        System.out.println("Byte value: "+byVal);

    }

}
```

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

public static byte parseByte(String s,

int radix)

string "Ab12Cd3" into a Byte, which will result in a **NumberFormatException** because the string contains non-numeric characters and cannot be converted to a Byte.

The Byte.parseByte(String) method expects a valid numeric string that falls within the range of a byte (-128 to 127). Non-numeric characters such as "Ab" and "Cd" make this string invalid for conversion.

```
public class fNumberException {
    public static void main(String[] args) {
        String strNumber="Ab12Cd3" ;
        Byte byteVal=Byte.parseByte(strNumber);
        System.out.println("Byte value: "+byteVal);
    }
}
```

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

```
public class gByte {
    public static void main(String[] args) {
        byte number=127;
        Byte num=Byte.valueOf(number);
        System.out.println("Byte : "+num);
        //String str=String.valueOf(number);
        //System.out.println("Byte : "+str);

    }
}
```

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

```
public class hByteString {
    public static void main(String[] args) {
        //unboxing
        String strNumber="100";

        Byte byt= Byte.valueOf(strNumber);
        System.out.println(byt);
        //Integer in=Integer.parseInt(strNumber);
        //System.out.println(in);

       /* char ch='A';
        Integer n=Integer.valueOf(ch);
        System.out.println(n);
        */
    }

}
```

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

```
public class iByteToPrimitive {
    public static void main(String[] args) {
        byte bytVal=100;
        Integer intVal=Integer.valueOf(bytVal);
        System.out.println(intVal);

       //String str=String.valueOf(bytVal);
       //System.out.println(str);



    }
}
```

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).

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

    Short sh=Short.BYTES;
    System.out.println(sh); //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).

```
public class cShortMinMax {
    public static void main(String[] args) {
        //Short sh=Short.MIN_VALUE;
        //System.out.println(sh);  //-32768

        Short sh=Short.MAX_VALUE;
        System.out.println(sh);   //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)).

```
public class dShortToString {
    public static void main(String[] args) {
        Short number=30712;
        String srt=Short.toString(number);
        System.out.println(srt);
    }

}
```

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

```
public class eStringToShort {
    public static void main(String[] args) {
        String strNumber="100000";
        Short str=Short.parseShort(strNumber);
        System.out.println(str);
    }
}
```

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

```
public class fNumberException {
    public static void main(String[] args) {
        String strNumber="Ab12Cd3";
        Short srt=Short.parseShort(strNumber);

        System.out.println(srt);
    }
}
```

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

```
public class gShortVal {
    public static void main(String[] args) {
        Short number=123;
        int num=number;
       // String str=String.valueOf(number);
        Integer inti=Integer.valueOf(num);
        System.out.println(inti);
    }
}
```

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

```
public class hShort {
    public static void main(String[] args) {
        String strNumber="123";
        Short str=Short.valueOf(strNumber);
        System.out.println(str);
```

```
    }

}
```

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

```
public class iShortToPrimitive {
    public static void main(String[] args) {
       // Short str=1;
       // byte byt = (byte) str;
        //byte byt=Byte.parseByte(str);
        //char ch=(char)str.str();
        //System.out.println(ch);

        Short shortValue = 65;
        // Short to byte using narrowing typecasting
        byte byteValue = (byte) (shortValue.shortValue());

        // Short to char using typecasting
        char charValue = (char) shortValue.shortValue();

        System.out.println("Byte value: " + byteValue);
        System.out.println("Char value: " + charValue);

    }
}
```

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).

```
public class bIntToByte {

    public static void main(String[] args) {
        byte byt=Integer.BYTES;
        System.out.println(byt); //4
    }
}
```

1. Write a program to find the minimum and maximum values of int using the MIN_VALUE and MAX_VALUE fields. (Hint: Use Integer.MIN_VALUE and Integer.MAX_VALUE).

```
public class cIntMinMax {
    public static void main(String[] args) {
        int n=Integer.MIN_VALUE;
        System.out.println(n);
        int m=Integer.MAX_VALUE;
        System.out.println(m);
    }
}
```

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

```
public class dIntToString {

    public static void main(String[] args) {
        int number=11;
        String str=Integer.toString(number);
        System.out.println(str);
    }
}
```

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

```
public class eStringToInt {
    public static void main(String[] args) {
        String strNumber="100000";
        Integer str=Integer.parseInt(strNumber);
        System.out.println(str);
    }

}
```

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

```
public class fException {
    public static void main(String[] args) {
        String strNumber="Ab12Cd3";
        Integer srt=Integer.parseInt(strNumber);

        System.out.println(srt);
    }
}
```

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

```
public class gInt {
    public static void main(String[] args) {
        Short number=123;
        int num=number;
       // String str=String.valueOf(number);
```

```
        Integer inti=Integer.valueOf(num);
        System.out.println(inti);
    }
}
```

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

```
public class hStringInt {
    public static void main(String[] args) {
        String strNumber="123";
        Integer str=Integer.valueOf(strNumber);
        System.out.println(str);
    }

}
```

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

```
public class iInteger {
    public static void main(String[] args) {
        int n=10;
        int m=20;
       int a= Integer.sum(10,20);
        System.out.println(a);
    }
}
```

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

```
public class jMinMaxValue {
    public static void main(String[] args) {
        int a=10;
        int b=20;
        int min=Integer.min(a, b);
        System.out.println("Min: "+min);

        int max=Integer.max(a, b);
        System.out.println("Max :"+max);


    }
}
```

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

```
public class kConvert {
    public static void main(String[] args) {
        int n=15;
        String a=Integer.toBinaryString(n);
        String b=Integer.toOctalString(n);
        String c=Integer.toHexString(n);

        System.out.println(a);
        System.out.println(b);
        System.out.println(c);

    }

}
```

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

```
public class lIntToPrimitive {
    public static void main(String[] args) {
        int n=10;
        double num=(double)n;
        System.out.println(num);
    }

}
```

5. Working with java.lang.Long

1. Explore the Java API documentation for java.lang.Long and observe its modifiers and super types.
2. Write a program to test how many bytes are used to represent a long value using the BYTES field. (Hint: Use Long.BYTES).

```
public class bLongByte {
    public static void main(String[] args) {
        byte byt=Long.BYTES;
        System.out.println(byt);
    }
}
```

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

```
public class cMinMax {
    public static void main(String[] args) {
        long n=Long.MIN_VALUE;
        System.out.println(n);
        long m=Long.MAX_VALUE;
```

```
        System.out.println(m);
    }
}
```

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

```
public class dLongToString {

    public static void main(String[] args) {
        long number=11;
        String str=Long.toString(number);
        System.out.println(str);
    }
}
```

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

```
public class eStringToLong {
    public static void main(String[] args) {
        String strNumber="100000";
        Long str=Long.parseLong(strNumber);
        System.out.println(str);
    }

}
```

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

```
public class fException {
    public static void main(String[] args) {
        String strNumber="Ab12Cd3";
        long srt=Long.parseLong(strNumber);

        System.out.println(srt);
    }
}
```

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

```
public class gInt {
    public static void main(String[] args) {
        Short number=123;
        int num=number;
       // String str=String.valueOf(number);
        Long inti=Long.valueOf(num);
        System.out.println(inti);
    }
}
```

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

```
public class hStringLong {
    public static void main(String[] args) {
        String strNumber="123";
        long str=Long.valueOf(strNumber);
        System.out.println(str);
    }
```

```
}
```

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

```
public class iInteger {
    public static void main(String[] args) {
        long n=1123;
        long m=9845;
       long a= Long.sum(n,m);
        System.out.println(a);
    }
}
```

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

```
public class jMinMaxValue {
    public static void main(String[] args) {
        long a=1122;
        long b=5566;
        long min=Long.min(a, b);
        System.out.println("Min: "+min);

        long max=Long.max(a, b);
        System.out.println("Max :"+max);


    }
}
```

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

Long.toBinaryString(long), Long.toOctalString(long), and Long.toHexString(long)).

```
public class kConvert {
    public static void main(String[] args) {
        long n=15;
        String a=Long.toBinaryString(n);
        String b=Long.toOctalString(n);
        String c=Long.toHexString(n);

        System.out.println(a);
        System.out.println(b);
        System.out.println(c);

    }

}
```

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

```
public class jPrimitive {

        public static void main(String[] args) {
            long n=10.1l;
            double num=(double)n;
            int num1=(int)n;
            System.out.println(num);
            System.out.println(num1);
        }


}
```

6. Working with java.lang.Float

1. Explore the Java API documentation for java.lang.Float and observe its modifiers and super types.
2. Write a program to test how many bytes are used to represent a float value using the BYTES field. (Hint: Use Float.BYTES).

```
public class bFloatByte {
    public static void main(String[] args) {
        byte byt=Float.BYTES;
        System.out.println(byt);
    }
}
```

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

```
public class cMinMax {
    public static void main(String[] args) {
        float n=Float.MIN_VALUE;
        System.out.println(n);
        float m=Float.MAX_VALUE;
        System.out.println(m);
    }
}
```

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

```
public class dFloatToString {

    public static void main(String[] args) {
        float number=11;
        String str=Float.toString(number);
        System.out.println(str);
```

```
    }
}
```

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

```
public class eStringToFloat {
    public static void main(String[] args) {
        String strNumber="100000";
        Float str=Float.parseFloat(strNumber);
        System.out.println(str);
    }

}
```

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

```
public class fException {
    public static void main(String[] args) {
        String strNumber="Ab12Cd3";
        Float srt=Float.parsefloat(strNumber);

        System.out.println(srt);
    }
}
```

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

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

```
        Short number=123;
        int num=number;
       // String str=String.valueOf(number);
        float inti=Float.valueOf(num);
        System.out.println(inti);
    }
}
```

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

```
public class hStringFloat {
    public static void main(String[] args) {
        String strNumber="123";
        float str=Float.valueOf(strNumber);
        System.out.println(str);
    }

}
```

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

```
public class iFloat {
    public static void main(String[] args) {
        float n=112.3;
        float m=984.5;
       float a= Float.sum(n,m);
        System.out.println(a);
    }
}
```

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

and Float.max(float, float)).

```
public class jMinMaxValue {
    public static void main(String[] args) {
        float a=1122;
        float b=5566;
        float min=Float.min(a, b);
        System.out.println("Min: "+min);

        float max=Float.max(a, b);
        System.out.println("Max :"+max);



    }
}
```

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

```
public class ksquroot {
    public static void main(String[] args) {
        float n=25.0f;
        double s=Math.sqrt(n);
        System.out.println(s);  //5.0
    }
}
```

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

```
public class iSameVar {
    public static void main(String[] args) {
        float a=2.0f;
        float b=2.0f;
        float c=a/b;
```

```
        System.out.println(c);
    }
}
```

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

```
public class jPrimitive {

        public static void main(String[] args) {
            float n=10.1f;
            double num=(double)n;
            int num1=(int)n;
            System.out.println(num);
            System.out.println(num1);
        }


}
```

7. Working with java.lang.Double

1. Explore the Java API documentation for java.lang.Double and observe its modifiers and super types.
2. Write a program to test how many bytes are used to represent a double value using the BYTES field. (Hint: Use Double.BYTES).

```
public class bDoubleByte {
    public static void main(String[] args) {
        byte byt=Double.BYTES;
        System.out.println(byt);
    }
}
```

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

```
public class cMinMax {
    public static void main(String[] args) {
        double n=Double.MIN_VALUE;
        System.out.println(n);
       double m=Double.MAX_VALUE;
        System.out.println(m);
    }
}
```

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

```
public class dDoubleToString {

    public static void main(String[] args) {
        double number=11.od;
        String str=Double.toString(number);
        System.out.println(str);
    }
}
```

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

```
public class eStringToDouble {
    public static void main(String[] args) {
        String strNumber="100000";
        Double str=Double.parseDouble(strNumber);
        System.out.println(str);
```

```
    }

}
```

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

```
public class fException {
    public static void main(String[] args) {
        String strNumber="Ab12Cd3";
        double srt=Double.parseDouble(strNumber);

        System.out.println(srt);
    }
}
```

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

```
public class gDouble {
    public static void main(String[] args) {
        Short number=123;
        int num=number;
       // String str=String.valueOf(number);
        double inti=Double.valueOf(num);
        System.out.println(inti);
    }
}
```

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

```
public class hStringDouble {
    public static void main(String[] args) {
        String strNumber="123";
        double str=Double.valueOf(strNumber);
        System.out.println(str);
    }

}
```

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

```
public class iDouble {
    public static void main(String[] args) {
        double n=112.3;
        double m=984.5;
       double a= Double.sum(n,m);
        System.out.println(a);
    }
}
```

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

```
public class jMinMaxValue {
    public static void main(String[] args) {
        double a=1122;
        double b=5566;
        double min=Double.min(a, b);
        System.out.println("Min: "+min);

        double max=Double.max(a, b);
        System.out.println("Max :"+max);
```

```
    }
}
```

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

```
public class ksquroot {
    public static void main(String[] args) {
        float n=25.0f;
        double s=Math.sqrt(n);
        System.out.println(s);  //5.0
    }
}
```

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

```
public class iSameVar {
    public static void main(String[] args) {
        double a=2.0f;
        float b=2.0f;
        float c=a/b;
        System.out.println(c);
    }
}
```

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

```
public class jPrimitive {

        public static void main(String[] args) {
            double n=10.1d;
            float num=(float)n;
```

```
            int num1=(int)n;
            System.out.println(num);
            System.out.println(num1);
        }


}
```

8. Conversion between Primitive Types and Strings

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

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

```
public class PrimitiveToString {
 public static void main(String[] args) {
    int n=100;
    String a=Integer.toString(n);
    System.out.println(a);
   // String s=String.valueOf(n);
   // System.out.println(s);

    Float m=10.0f;
   // String s=String.valueOf(m);
   // System.out.println(s);
    String s=Float.toString(m);
    System.out.println(s);

    char c='A';
    String ch=String.valueOf(c);
    //String s=Char.toString(c);
    System.out.println(s);
```

```
    byte b=127;
    String by=String.valueOf(b);
    //String s=Float.toString(b);
    //System.out.println(s);

 }

}
```

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).

```
public class DefaultValuePrimitive {

    boolean boolVal;
    byte byteVal;
    char charVal;
    short shVal;
    int intVal;
    float floatVal;
    double doubleVal;
    long longVal;

    public static void main(String[] args) {
        DefaultValuePrimitive dat=new DefaultValuePrimitive();

        System.out.println("Boolean:"+dat.boolVal);
        System.out.println("Byte:"+dat.byteVal);
        System.out.println("Char:' " + dat.charVal +"'");
        System.out.println("Short:"+dat.shVal);
        System.out.println("Integer:"+dat.intVal);
        System.out.println("Float:"+dat.floatVal);
        System.out.println("Double:"+dat.doubleVal);
```

```
        System.out.println("Long:"+dat.longVal);

    }

}
```

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).

```
public class ArithmOp {
    public static void main(String args[]) {
        double num1=Double.parseDouble(args[0]);
       char op = args[1].charAt(0);
        //int op=String.parseString(args[1]);
        double num2=Double.parseDouble(args[2]);

        switch (op) {
            case '+':
            System.out.println(num1+num2);
                break;
                case '-':
            System.out.println(num1-num2);
                break;
                case '*':
                System.out.println(num1 * num2);
                    break;
              case '/':
                System.out.println(num1 / num2);
                    break;

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

```
            }

        }

}
```