**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](https://docs.oracle.com/javase/8/docs/api/java/lang/Boolean.html) 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) ).

**Sol-**- boolean flag=true;

String s=String.valueOf(flag);

System.out.println(s);

**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)).

**Sol— String** strStatus **=”true”;**

boolean f1 = Boolean.parseBoolean(s);

System.out.println(f1);

**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 s="1",s1="0";

boolean f1 = Boolean.parseBoolean(s);

boolean f2 = Boolean.parseBoolean(s1);

System.out.println(f1);

System.out.println(f2);  
OUTPUT—False,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)).

**Sol-**- boolean status = true;

Boolean wrapperStatus = Boolean.valueOf(status);

System.out.println("Wrapper class Boolean value: " + wrapperStatus);

**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)).

**Sol-**- String status = "0";

Boolean wrapperStatus = Boolean.valueOf(status);

System.out.println("Wrapper class Boolean value: " + wrapperStatus);

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

**Sol-**-

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

**a.** Explore the [Java API documentation for java.lang.Byte](https://docs.oracle.com/javase/8/docs/api/java/lang/Byte.html) 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).

**sol-**-System.out.println("Bytes used to represent a byte value: " + Byte.BYTES);

**c.** Write a program to find the minimum and maximum values of byte using the MIN\_VALUE and MAX\_VALUE fields. (Hint: Use Byte.MIN\_VALUE and Byte.MAX\_VALUE).

**Sol-**-System.out.println("Byte MIN\_VALUE: " + Byte.MIN\_VALUE); System.out.println("Byte MAX\_VALUE: " + Byte.MAX\_VALUE);

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

Sol--byte number = 10; String byteString = Byte.toString(number); System.out.println("Byte to String: " + byteString);

**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)).

**Sol-**-String strNumber = "20"; byte byteFromString = Byte.parseByte(strNumber); System.out.println("String to Byte: " + byteFromString);

**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).

**Sol-**-try {

String invalidStr = "Ab12Cd3";

Byte.parseByte(invalidStr);

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

}

**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)).

**Sol-**-Byte wrapperByte = Byte.valueOf(number);

System.out.println("Wrapper class Byte value: " + wrapperByte);

**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)).

**Sol-**-String strByte = "30";

Byte wrapperByteFromString = Byte.valueOf(strByte);

System.out.println("Wrapper class Byte from String: " + wrapperByteFromString);

}

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

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

**a.** Explore the [Java API documentation for java.lang.Short](https://docs.oracle.com/javase/8/docs/api/java/lang/Short.html) 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).

**sol-**-System.out.println("Bytes used to represent a short value: " + Short.BYTES);

**c.** Write a program to find the minimum and maximum values of short using the MIN\_VALUE and MAX\_VALUE fields. (Hint: Use Short.MIN\_VALUE and Short.MAX\_VALUE).

**Sol-**-System.out.println("Short MIN\_VALUE: " + Short.MIN\_VALUE);

System.out.println("Short MAX\_VALUE: " + Short.MAX\_VALUE);

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

**Sol-**- short shortNumber = 100;

String shortString = Short.toString(shortNumber);

System.out.println("Short to String: " + shortString);

**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)).

**Sol-**-String strShortNumber = "200";

short shortFromString = Short.parseShort(strShortNumber);

System.out.println("String to Short: " + shortFromString);

**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).

**Sol-**- try {

String invalidStr = "Ab12Cd3";

Short.parseShort(invalidStr);

} catch (NumberFormatException e) {

System.out.println("NumberFormatException: " + e.getMessage());

}

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

**Sol-**-Short wrapperShort = Short.valueOf(shortNumber);

System.out.println("Wrapper class Short value: " + wrapperShort);

**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)).

**Sol-**-String strShort = "150";

Short wrapperShortFromString = Short.valueOf(strShort);

System.out.println("Wrapper class Short from String: " + wrapperShortFromString);

}

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

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

**a.** Explore the [Java API documentation for java.lang.Integer](https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html) 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).

**sol-**-System.out.println("Bytes used to represent an int value: " + Integer.BYTES);

**c.** Write a program to find the minimum and maximum values of int using the MIN\_VALUE and MAX\_VALUE fields. (Hint: Use Integer.MIN\_VALUE and Integer.MAX\_VALUE).

**Sol-**-System.out.println("Integer MIN\_VALUE: " + Integer.MIN\_VALUE);

System.out.println("Integer MAX\_VALUE: " + Integer.MAX\_VALUE);

**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)).  
sol--int intNumber = 500;

String intString = Integer.toString(intNumber);

System.out.println("Integer to String: " + intString);

**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)).

**Sol-**-String strIntNumber = "600";

int intFromString = Integer.parseInt(strIntNumber);

System.out.println("String to Integer: " + intFromString);

**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).

**Sol-**-try {

String invalidStr = "Ab12Cd3";

Integer.parseInt(invalidStr);

} catch (NumberFormatException e) {

System.out.println("NumberFormatException: " + e.getMessage());

}

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

**Sol-**- Integer wrapperInteger = Integer.valueOf(intNumber);

System.out.println("Wrapper class Integer value: " + wrapperInteger);

**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)).

**Sol-**-String strInteger = "700";

Integer wrapperIntegerFromString = Integer.valueOf(strInteger);

System.out.println("Wrapper class Integer from String: " + wrapperIntegerFromString);

**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)).

int a = 10;

int b = 20;

int sum = Integer.sum(a, b);

System.out.println("Sum of 10 and 20: " + sum);

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

**Sol-**- int min = Integer.min(a, b);

int max = Integer.max(a, b);

System.out.println("Minimum of 10 and 20: " + min);

System.out.println("Maximum of 10 and 20: " + max);

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

**Sol-**-int value = 7;

System.out.println("Binary representation: " + Integer.toBinaryString(value));

System.out.println("Octal representation: " + Integer.toOctalString(value));

System.out.println("Hexadecimal representation: " + Integer.toHexString(value));

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

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

**a.** Explore the [Java API documentation for java.lang.Long](https://docs.oracle.com/javase/8/docs/api/java/lang/Long.html) 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).

sol--System.out.println("Bytes used to represent a long value: " + Long.BYTES);

**c.** Write a program to find the minimum and maximum values of long using the MIN\_VALUE and MAX\_VALUE fields. (Hint: Use Long.MIN\_VALUE and Long.MAX\_VALUE).

**Sol-**- System.out.println("Long MIN\_VALUE: " + Long.MIN\_VALUE);

System.out.println("Long MAX\_VALUE: " + Long.MAX\_VALUE);

**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)).

**Sol-**- long longNumber = 1000L;

String longString = Long.toString(longNumber);

System.out.println("Long to String: " + longString);

**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)).

**Sol-**- String strLongNumber = "2000";

long longFromString = Long.parseLong(strLongNumber);

System.out.println("String to Long: " + longFromString);

**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).

**Sol-**- try {

String invalidStr = "Ab12Cd3";

Long.parseLong(invalidStr);

} catch (NumberFormatException e) {

System.out.println("NumberFormatException: " + e.getMessage());

}

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

**Sol-**- Long wrapperLong = Long.valueOf(longNumber);

System.out.println("Wrapper class Long value: " + wrapperLong);

**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)).

**Sol-**- String strLong = "3000";

Long wrapperLongFromString = Long.valueOf(strLong);

System.out.println("Wrapper class Long from String: " + wrapperLongFromString);

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)).

Sol--long l1 = 1123L;

long l2 = 9845L;

long longSum = Long.sum(l1, l2);

System.out.println("Sum of 1123 and 9845: " + longSum);

**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)).

**Sol-**- long minLong = Long.min(l1, l2);

long maxLong = Long.max(l1, l2);

System.out.println("Minimum of 1123 and 9845: " + minLong);

System.out.println("Maximum of 1123 and 9845: " + maxLong);

**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)).

**Sol-**-long longValue = 7L;

System.out.println("Binary representation: " + Long.toBinaryString(longValue));

System.out.println("Octal representation: " + Long.toOctalString(longValue));

System.out.println("Hexadecimal representation: " + Long.toHexString(longValue));

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

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

**a.** Explore the [Java API documentation for java.lang.Float](https://docs.oracle.com/javase/8/docs/api/java/lang/Float.html) 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).

**sol-**-System.out.println("Bytes used to represent a float value: " + Float.BYTES);

**c.** Write a program to find the minimum and maximum values of float using the MIN\_VALUE and MAX\_VALUE fields. (Hint: Use Float.MIN\_VALUE and Float.MAX\_VALUE).

**Sol-**-System.out.println("Float MIN\_VALUE: " + Float.MIN\_VALUE);

System.out.println("Float MAX\_VALUE: " + Float.MAX\_VALUE);

**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)).

**Sol-**- float floatNumber = 10.5f;

String floatString = Float.toString(floatNumber);

System.out.println("Float to String: " + floatString);

**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)).

**Sol-**-String strFloatNumber = "20.5";

float floatFromString = Float.parseFloat(strFloatNumber);

System.out.println("String to Float: " + floatFromString);

**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).

**Sol-**-try {

String invalidStr = "Ab12Cd3";

Float.parseFloat(invalidStr);

} catch (NumberFormatException e) {

System.out.println("NumberFormatException: " + e.getMessage());

}

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

**Sol-**- Float wrapperFloat = Float.valueOf(floatNumber);

System.out.println("Wrapper class Float value: " + wrapperFloat);

**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)).

**Sol-**- String strFloat = "30.5";

Float wrapperFloatFromString = Float.valueOf(strFloat);

System.out.println("Wrapper class Float from String: " + wrapperFloatFromString);

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)).

Sol-- float f1 = 112.3f;

float f2 = 984.5f;

float floatSum = Float.sum(f1, f2);

System.out.println("Sum of 112.3 and 984.5: " + floatSum);

**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)).

**Sol-**-float minFloat = Float.min(f1, f2);

float maxFloat = Float.max(f1, f2);

System.out.println("Minimum of 112.3 and 984.5: " + minFloat);

System.out.println("Maximum of 112.3 and 984.5: " + maxFloat);

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

**Sol-**- float negativeFloat = -25.0f;

double sqrtValue = Math.sqrt(negativeFloat);

System.out.println("Square root of -25.0f: " + sqrtValue);

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

**Sol-**- float zeroFloat1 = 0.0f;

float zeroFloat2 = 0.0f;

float floatDivision = zeroFloat1 / zeroFloat2;

System.out.println("0.0f / 0.0f: " + floatDivision);

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

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

**a.** Explore the [Java API documentation for java.lang.Double](https://docs.oracle.com/javase/8/docs/api/java/lang/Double.html) 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).

**sol-**-System.out.println("Bytes used to represent a double value: " + Double.BYTES);

**c.** Write a program to find the minimum and maximum values of double using the MIN\_VALUE and MAX\_VALUE fields. (Hint: Use Double.MIN\_VALUE and Double.MAX\_VALUE).

**Sol-**-System.out.println("Double MIN\_VALUE: " + Double.MIN\_VALUE);

System.out.println("Double MAX\_VALUE: " + Double.MAX\_VALUE);

**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)).

**Sol-**-double doubleNumber = 50.5;

String doubleString = Double.toString(doubleNumber);

System.out.println("Double to String: " + doubleString);

**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)).

**Sol- String strDoubleNumber = "60.5";**

**double doubleFromString = Double.parseDouble(strDoubleNumber);**

**System.out.println("String to Double: " + doubleFromString);**

**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).

**Sol-**-try {

String invalidStr = "Ab12Cd3";

Double.parseDouble(invalidStr);

} catch (NumberFormatException e) {

System.out.println("NumberFormatException: " + e.getMessage());

}

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

**Sol-**-Double wrapperDouble = Double.valueOf(doubleNumber);

System.out.println("Wrapper class Double value: " + wrapperDouble);

**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)).

**Sol-**- String strDouble = "70.5";

Double wrapperDoubleFromString = Double.valueOf(strDouble);

System.out.println("Wrapper class Double from String: " + wrapperDoubleFromString);

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)).

Sol--double d1 = 112.3;

double d2 = 984.5;

double doubleSum = Double.sum(d1, d2);

System.out.println("Sum of 112.3 and 984.5: " + doubleSum);

**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)).

**Sol-**- double minDouble = Double.min(d1, d2);

double maxDouble = Double.max(d1, d2);

System.out.println("Minimum of 112.3 and 984.5: " + minDouble);

System.out.println("Maximum of 112.3 and 984.5: " + maxDouble);

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

**Sol-**- double negativeDouble = -25.0;

double sqrtDouble = Math.sqrt(negativeDouble);

System.out.println("Square root of -25.0: " + sqrtDouble);

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

**Sol-**-double zeroDouble1 = 0.0;

double zeroDouble2 = 0.0;

double doubleDivision = zeroDouble1 / zeroDouble2;

System.out.println("0.0 / 0.0: " + doubleDivision);

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

#### ****8. Conversion between Primitive Types and Strings****

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

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

sol-- public static void primitiveToStringConversions() {

int intValue = 10;

float floatValue = 5.5f;

double doubleValue = 8.8;

boolean booleanValue = true;

char charValue = 'A';

System.out.println("Integer to String: " + Integer.toString(intValue));

System.out.println("Float to String: " + Float.toString(floatValue));

System.out.println("Double to String: " + Double.toString(doubleValue));

System.out.println("Boolean to String: " + Boolean.toString(booleanValue));

System.out.println("Character to String: " + Character.toString(charValue));

System.out.println("String.valueOf(int): " + String.valueOf(intValue));

System.out.println("String.valueOf(float): " + String.valueOf(floatValue));

System.out.println("String.valueOf(double): " + String.valueOf(doubleValue));

System.out.println("String.valueOf(boolean): " + String.valueOf(booleanValue));

System.out.println("String.valueOf(char): " + String.valueOf(charValue));

}

#### ****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).

Sol-- public class DefaultValues {

// Instance variables

boolean bool;

char character;

byte byteValue;

short shortValue;

int intValue;

long longValue;

float floatValue;

double doubleValue;

// Static variables

static boolean staticBool;

static char staticCharacter;

static byte staticByteValue;

static short staticShortValue;

static int staticIntValue;

static long staticLongValue;

static float staticFloatValue;

static double staticDoubleValue;

public static void main(String[] args) {

DefaultValues defaultValues = new DefaultValues();

// Print instance variable default values

System.out.println("Instance variable default values:");

System.out.println("boolean: " + defaultValues.bool);

System.out.println("char: [" + defaultValues.character + "]");

System.out.println("byte: " + defaultValues.byteValue);

System.out.println("short: " + defaultValues.shortValue);

System.out.println("int: " + defaultValues.intValue);

System.out.println("long: " + defaultValues.longValue);

System.out.println("float: " + defaultValues.floatValue);

System.out.println("double: " + defaultValues.doubleValue);

// Print static variable default values

System.out.println("\nStatic variable default values:");

System.out.println("boolean: " + DefaultValues.staticBool);

System.out.println("char: [" + DefaultValues.staticCharacter + "]");

System.out.println("byte: " + DefaultValues.staticByteValue);

System.out.println("short: " + DefaultValues.staticShortValue);

System.out.println("int: " + DefaultValues.staticIntValue);

System.out.println("long: " + DefaultValues.staticLongValue);

System.out.println("float: " + DefaultValues.staticFloatValue);

System.out.println("double: " + DefaultValues.staticDoubleValue);

}

}

#### ****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).

Sol-- public class ArithmeticOperations {

public static void main(String[] args) {

if (args.length != 3) {

System.out.println("ArithmeticOperations <number1> <number2> <operator>");

System.exit(1);

}

int num1 = Integer.parseInt(args[0]);

int num2 = Integer.parseInt(args[1]);

String operator = args[2];

int result = 0;

boolean validOperator = true;

switch (operator) {

case "+":

result = num1 + num2;

break;

case "-":

result = num1 - num2;

break;

case "\*":

result = num1 \* num2;

break;

case "/":

if (num2 != 0) {

result = num1 / num2;

} else {

System.out.println("Error: Division by zero");

validOperator = false;

}

break;

default:

System.out.println("Error: Invalid operator");

validOperator = false;

break;

}

if (validOperator) {

System.out.println("Result: " + result);

}

}

}

}