**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) ).

**Solution:**

public class bool {

public static void main(String[] args) {

boolean status = true ;

String stringstr = Boolean.toString(status);

System.out.println(stringstr);

}

}

Output:



**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)).

**Solution:**

**public class boolean1{**

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

**String strStatus =new String("true");**

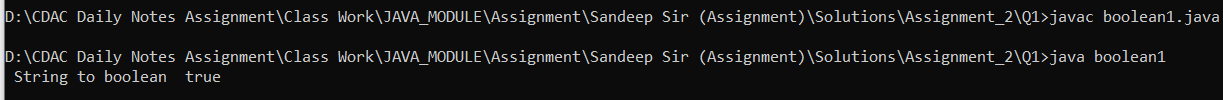
**boolean boolstatus=Boolean.parseBoolean(strStatus);**

**System.out.println(" String to boolean "+boolstatus);**

**}**

**}**

**Output:**

****

**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").

**Solution:**

**public class BooleanConversion {**

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

**String strStatus = "1"; // or "0"**

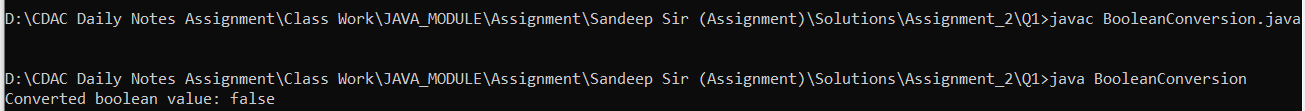
**boolean boolStatus = Boolean.parseBoolean(strStatus);**

**System.out.println("Converted boolean value: " + boolStatus);**

**}**

**}**

Output:

****

**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)).

**Solution:**

public class BooleanWrapper{

public static void main(String[] args) {

boolean status = true;

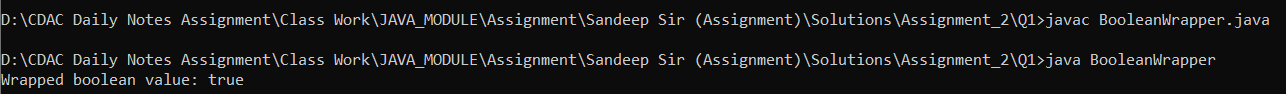
Boolean wrappedStatus = Boolean.valueOf(status); // Autoboxing also works

System.out.println("Wrapped boolean value: " + wrappedStatus);

}

}

Output:



**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)).

**Solution:**

**public class Boolea {**

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

**String strStatus = "true";**

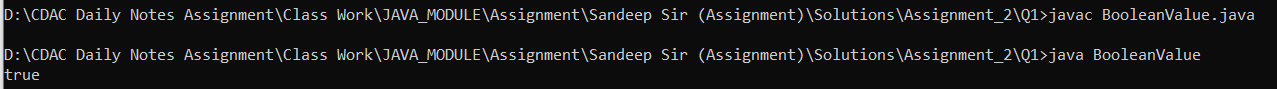
**Boolean wrappedStatus = Boolean.valueOf(strStatus);**

**System.out.println(wrappedStatus);**

**}**

**}**

**Output:**

****

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

**Solution:**

**public class Result{**

**public static void main(String args [])**

**{**

**boolean status=false;**

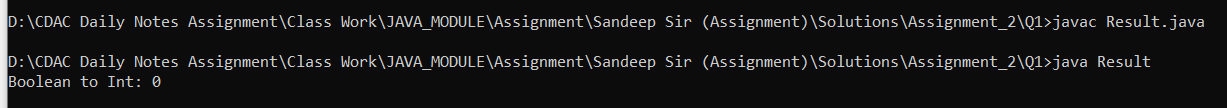
**int strbool = (status) ? 1 :0;**

**System.out.println("Boolean to Int: "+strbool);**

**}**

**}**

**Output:**



#### ****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).

**Solution:**

**class Byte1{**

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

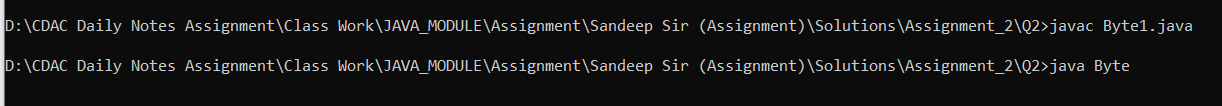
**int bytesUsed = Byte.BYTES;**

**System.out.println("Bytes used to represent a byte value: " +bytesUsed);**

**}**

**}**

**Output:**

****

**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).

**Solution:**

**public class Byte2 {**

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

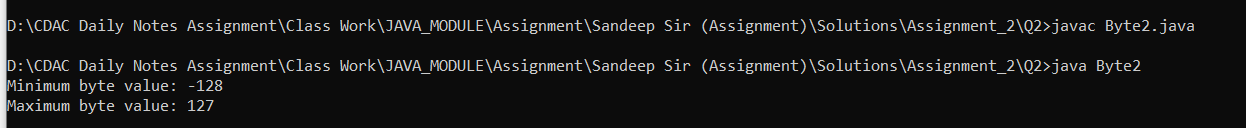
**System.out.println("Minimum byte value: " + Byte.MIN\_VALUE);**

**System.out.println("Maximum byte value: " + Byte.MAX\_VALUE);**

**}**

**}**

**Output:**

****

**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)).

**Solution:**

**public class Byte3 {**

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

**byte number = 100;**

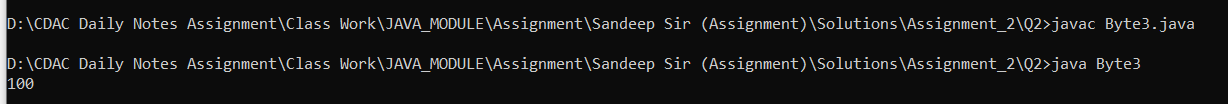
**String strNumber = Byte.toString(number);**

**System.out.println(strNumber);**

**}**

**}**

**Output:**

****

**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)).

**Solution:**

**public class Byte5{**

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

**byte number = 42;**

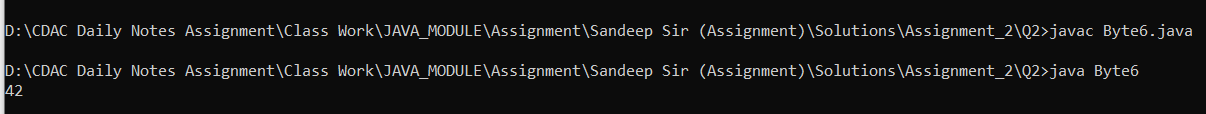
**Byte byteObject = Byte.valueOf(number);**

**System.out.println(byteObject);**

**}**

**}**

**Output:**

****

**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).

Solution:

public class Byte6{

public static void main(String[] args) {

String strNumber = "Ab12cd3";

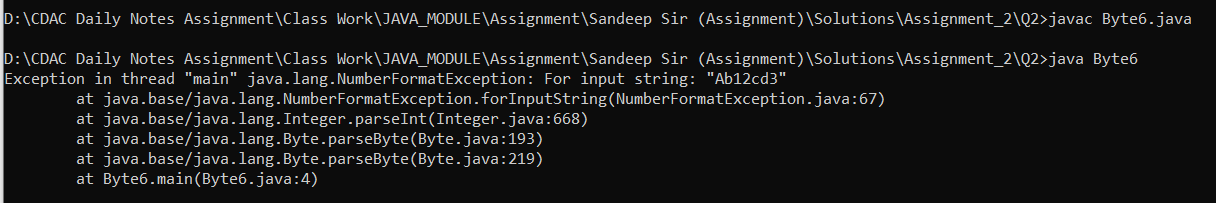
byte number = Byte.parseByte(strNumber);

System.out.println(number);

}

}

Output:



**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)).

**Solution:**

**public class Byte8{**

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

**String strNumber = "127";**

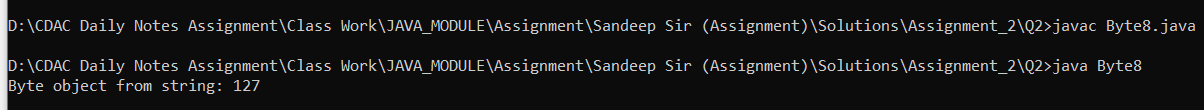
**Byte byteObject = Byte.valueOf(strNumber);**

**System.out.println("Byte object from string: " + byteObject);**

**}**

**}**

**Output:**



**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)).

**Solution:**

**public class Byte9{**

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

**byte number = 10;**

**int intValue = number;**

**short shortValue = number;**

**long longValue = number;**

**float floatValue = number;**

**double doubleValue = number;**

**System.out.println("Byte value as int: " + intValue);**

**System.out.println("Byte value as short: " + shortValue);**

**System.out.println("Byte value as long: " + longValue);**

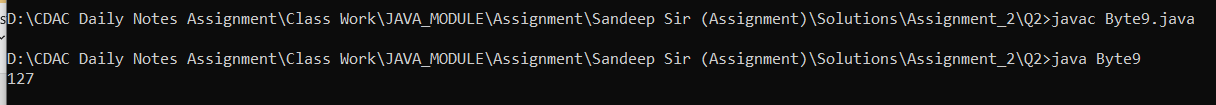
**System.out.println("Byte value as float: " + floatValue);**

**System.out.println("Byte value as double: " + doubleValue);**

**}**

**}**

Output:



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

Solution:

public class Byte10{

public static void main(String[] args) {

byte number = 10;

int intValue = number;

short shortValue = number;

long longValue = number;

float floatValue = number;

double doubleValue = number;

System.out.println("Byte value as int: " + intValue);

System.out.println("Byte value as short: " + shortValue);

System.out.println("Byte value as long: " + longValue);

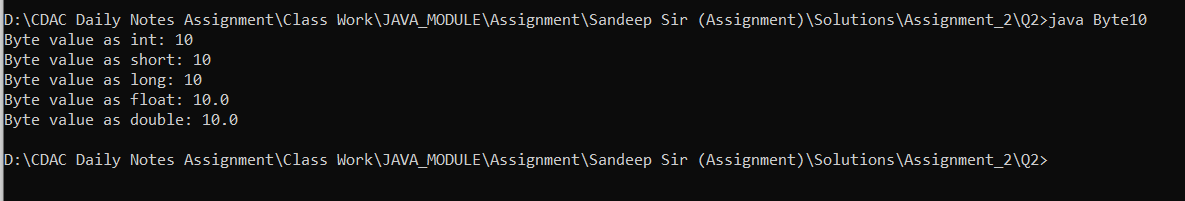
System.out.println("Byte value as float: " + floatValue);

System.out.println("Byte value as double: " + doubleValue);

}

}

Output:



#### ****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).

**Solution:**

**public class Short1 {**

**public static void main(String[] args)**

**{**

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

**}**

**}**

**Output:**

****

**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).

**Solution:**

**public class Short2{**

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

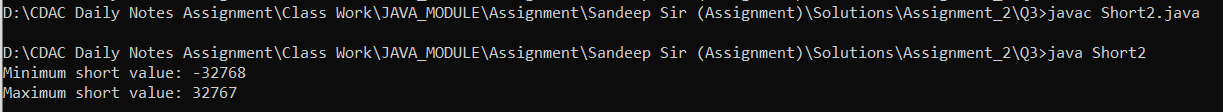
**System.out.println("Minimum short value: " + Short.MIN\_VALUE);**

**System.out.println("Maximum short value: " + Short.MAX\_VALUE);**

**}**

**}**

**Output:**



**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)).

**Solution:**

public class Short3{

public static void main(String[] args) {

short number = 24000;

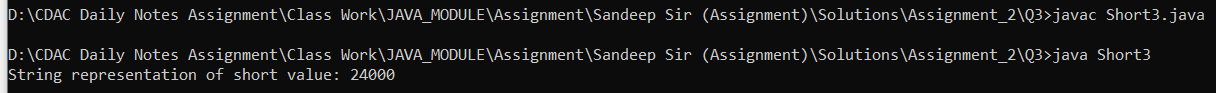
String strNumber = Short.toString(number);

System.out.println("String representation of short value: " + strNumber);

}

}

Output:



**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)).

**Solution:**

public class Short4 {

public static void main(String[] args) {

String strNumber = "12345";

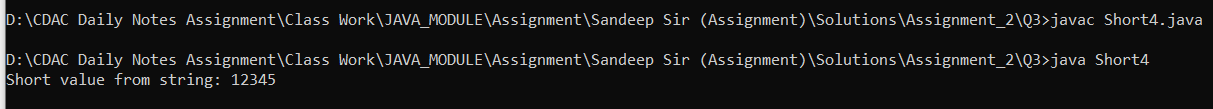
short number = Short.parseShort(strNumber);

System.out.println("Short value from string: " + number);

}

}

Output:



**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).

**Solution:**

public class Short5{

public static void main(String[] args) {

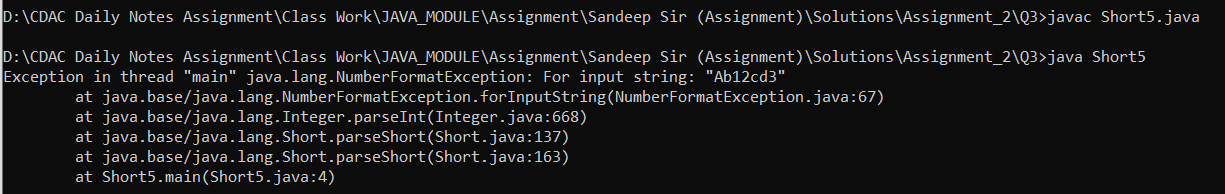
String strNumber = "Ab12cd3";

short number = Short.parseShort(strNumber);

System.out.println("Short value from string: " + number);

}

}



**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)).

**Solution:**

**public class Short6{**

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

**short number = 10000;**

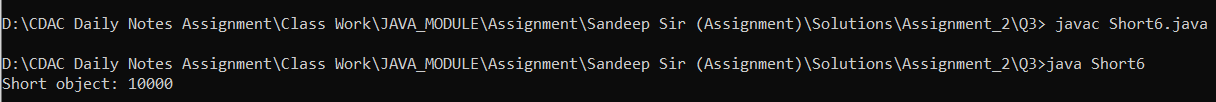
**Short wrapper = Short.valueOf(number);**

**System.out.println("Short object: " + wrapper);**

**}**

**}**

**Output:**

****

**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)).

**Solution:**

public class Short7{

public static void main(String[] args) {

String strNumber = "32767";

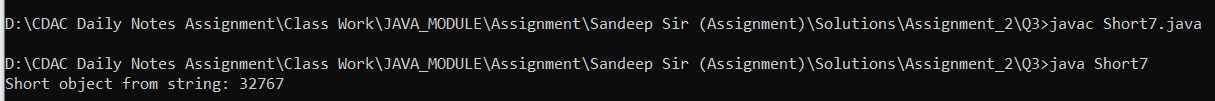
Short shortval = Short.valueOf(strNumber);

System.out.println("Short object from string: " + shortval);

}

}

Output:



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

Solution:

public class Short8 {

public static void main(String[] args) {

short number = 100;

int intValue = number;

byte byteValue = (byte) number; // Casting needed

long longValue = number;

float floatValue = number;

double doubleValue = number;

System.out.println("Short value as int: " + intValue);

System.out.println("Short value as byte (with casting): " + byteValue);

System.out.println("Short value as long: " + longValue);

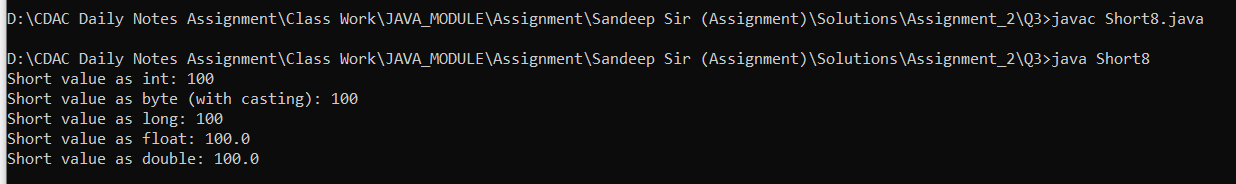
System.out.println("Short value as float: " + floatValue);

System.out.println("Short value as double: " + doubleValue);

}

}

Output:



#### ****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).

**Solution:**

public class Integer1 {

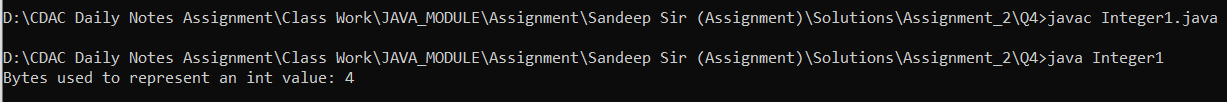
public static void main(String[] args) {

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

}

}

Output:



**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).

**Solution:**

**public class Integer2{**

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

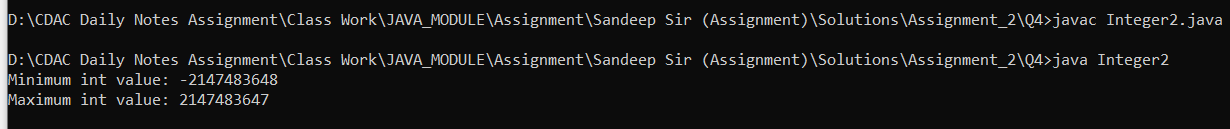
**System.out.println("Minimum int value: " + Integer.MIN\_VALUE);**

**System.out.println("Maximum int value: " + Integer.MAX\_VALUE);**

**}**

**}**

**Output:**

****

**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)).

Solution:

public class Integer3 {

public static void main(String[] args) {

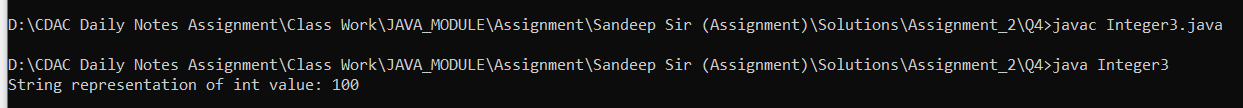
int number = 100;

String strNumber = Integer.toString(number);

System.out.println("String representation of int value: " + strNumber);

}

}



**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)).

**Solution:**

public class Integer4 {

public static void main(String[] args) {

String strNumber = "123";

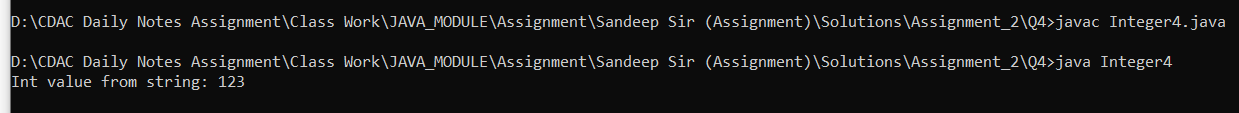
int number = Integer.parseInt(strNumber);

System.out.println("Int value from string: " + number);

}

}

Output:



**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).

**Solution:**

**public class String5 {**

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

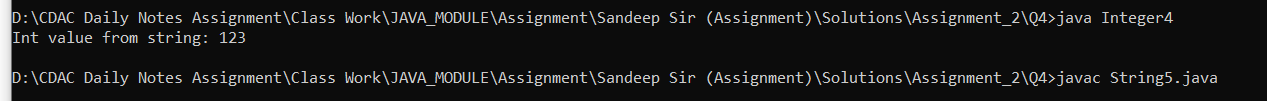
**String strNumber = "Ab12cd3";**

**int number = Integer.parseInt(strNumber);**

**System.out.println("int value from string: " + number);**

**}**

**}**



**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)).

**Solution:**

public class Integer6{

public static void main(String[] args) {

int number = 100;

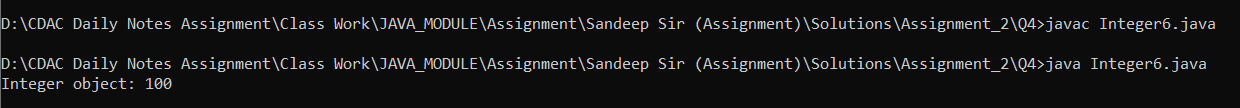
Integer wrapperObject = Integer.valueOf(number);

System.out.println("Integer object: " + wrapperObject);

}

}

Output:



**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)).

**Solution:**

public class Integer7{

public static void main(String[] args) {

String strNumber = "456";

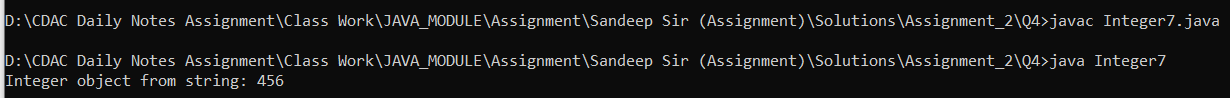
int wrapperObject = Integer.valueOf(strNumber);

System.out.println("Integer object from string: " + wrapperObject);

}

}

Output:



**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)).

Solution:

public class Integer8 {

public static void main(String[] args) {

int a = 10;

int b = 20;

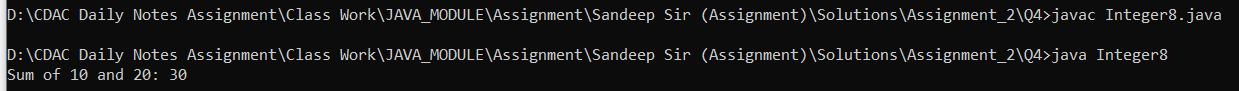
int sum = Integer.sum(a, b);

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

}

}

Output:



**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)).

**Solution:**

public class Integer9{

public static void main(String[] args) {

int a = 10;

int b = 20;

int minValue = Integer.min(a, b);

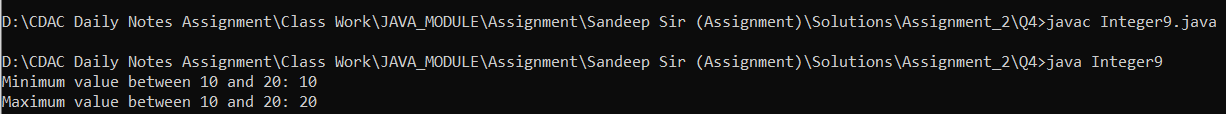
int maxValue = Integer.max(a, b);

System.out.println("Minimum value between 10 and 20: " + minValue);

System.out.println("Maximum value between 10 and 20: " + maxValue);

}

}



**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)).

**Solution:**

public class Integer10 {

public static void main(String[] args) {

int number = 7;

String binary = Integer.toBinaryString(number);

String octal = Integer.toOctalString(number);

String hex = Integer.toHexString(number);

System.out.println("Binary representation of 7: " + binary);

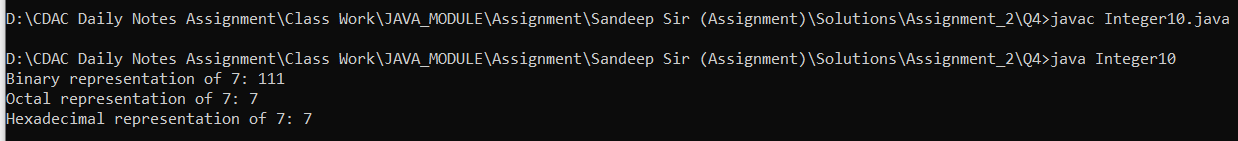
System.out.println("Octal representation of 7: " + octal);

System.out.println("Hexadecimal representation of 7: " + hex);

}

}

Output:



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

**Solution:**

public class Integer11 {

public static void main(String[] args) {

int number = 45 ;

byte byteValue = (byte) number; // Explicit casting needed

short shortValue = (short) number; // Explicit casting

long longValue = number;

float floatValue = number;

double doubleValue = number;

System.out.println("Int value as byte : " + byteValue); // with casting

System.out.println("Int value as short : " + shortValue); //with casting

System.out.println("Int value as long: " + longValue);

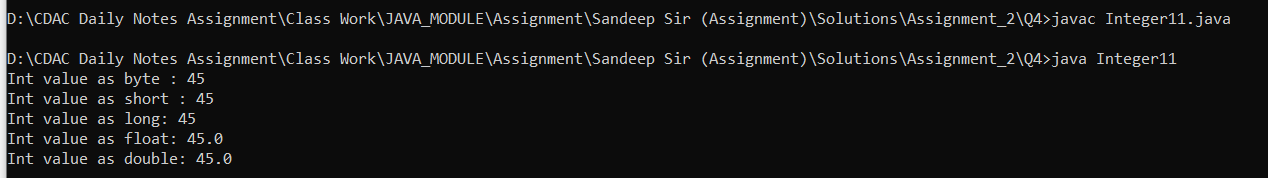
System.out.println("Int value as float: " + floatValue);

System.out.println("Int value as double: " + doubleValue);

}

}

Output:



#### ****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).

**Solution:**

public class LongBytes {

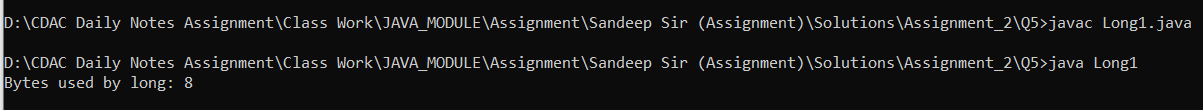
public static void main(String[] args) {

System.out.println("Bytes used by long: " + Long.BYTES);

}

}

**Output:**

****

**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).

**Solution:**

**public class Long2 {**

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

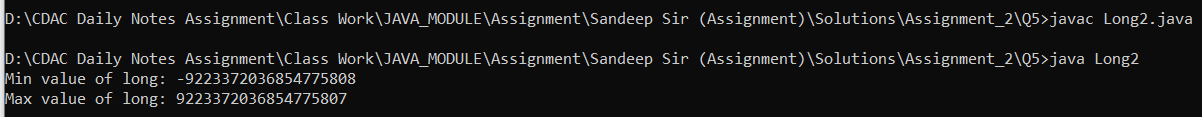
**System.out.println("Min value of long: " + Long.MIN\_VALUE);**

**System.out.println("Max value of long: " + Long.MAX\_VALUE);**

**}**

**}**

Output:



**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)).

**Solution:**

public class Long3 {

public static void main(String[] args) {

long number = 12345L; // Method-local variable

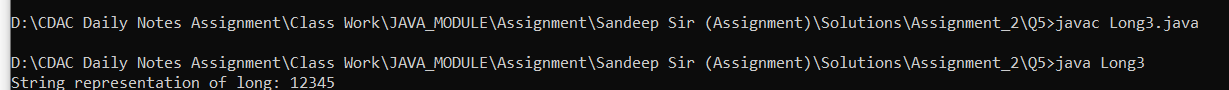
String strNumber = Long.toString(number); // Conversion to String

System.out.println("String representation of long: " + strNumber);

}

}

Output:



**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)).

**Solution:**

public class StringToLong {

public static void main(String[] args) {

String strNumber = "54321"; // Method-local variable

long number = Long.parseLong(strNumber); // Conversion to long

System.out.println("Converted long value: " + number);

}

}

Output:



**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).

**Solution:**

class InvalidStringToLong {

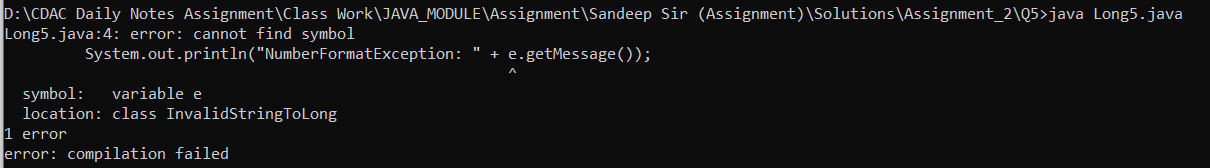
public static void main(String[] args) {

String strNumber = "Ab12Cd3"; // Method-local variable

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)).

**Solution:**

public class LongToWrapper {

public static void main(String[] args) {

long number = 97642SL; // Method-local variable

Long longWrapper = Long.valueOf(number); // Conversion to Long wrapper class

System.out.println("Long wrapper object: " + longWrapper);

}

}

Output:



**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)).

**public class StringToWrapper {**

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

**String strNumber = "67890"; // Method-local variable**

**Long longWrapper = Long.valueOf(strNumber); // Conversion to Long wrapper class**

**System.out.println("Long wrapper object: " + longWrapper);**

**}**

**}**

**Output:**

****

**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)).

**Solution:**

public class LongSum {

public static void main(String[] args) {

long num1 = 1123L;

long num2 = 9845L;

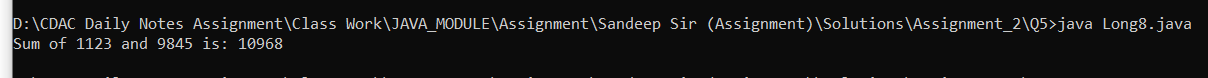
long sum = Long.sum(num1, num2); // Sum using Long class method

System.out.println("Sum of " + num1 + " and " + num2 + " is: " + sum);

}

}

Output:



**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)).

**Solution:**

public class long9{

public static void main(String[] args) {

long num1 = 1122;

long num2 = 5566;

long min = Long.min(num1, num2);

long max = Long.max(num1, num2);

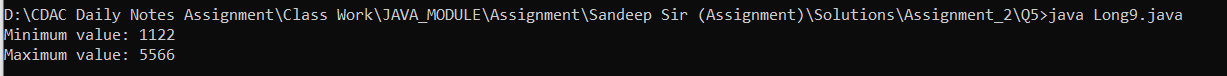
System.out.println("Minimum value: " + min);

System.out.println("Maximum value: " + max);

}

}

**Output:**

****

**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)).

**Solution:**

public class long10 {

public static void main(String[] args) {

long num = 7;

String binaryString = Long.toBinaryString(num);

String octalString = Long.toOctalString(num);

String hexString = Long.toHexString(num);

System.out.println("Binary representation: " + binaryString);

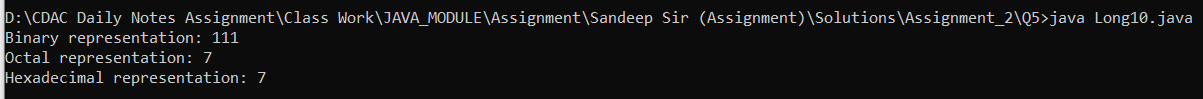
System.out.println("Octal representation: " + octalString);

System.out.println("Hexadecimal representation: " + hexString);

}

}

Output:



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

**Solution:**

public class long11 {

public static void main(String[] args) {

long num =1245679L;

int intValue = (int) num;

short shortValue = (short) num;

byte byteValue = (byte) num;

double doubleValue = (double) num;

float floatValue = (float) num;

System.out.println("Original long value: " + num);

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

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

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

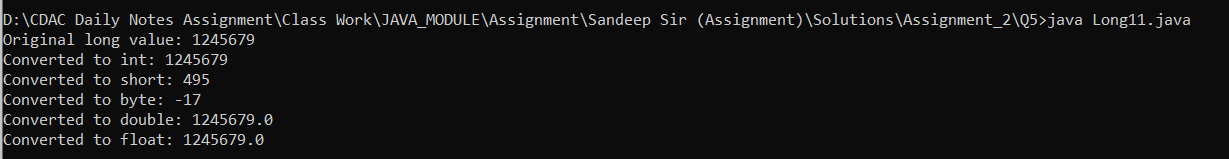
System.out.println("Converted to double: " + doubleValue);

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

}

}

Output:



#### ****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).

**Solution:**

public class FloatB {

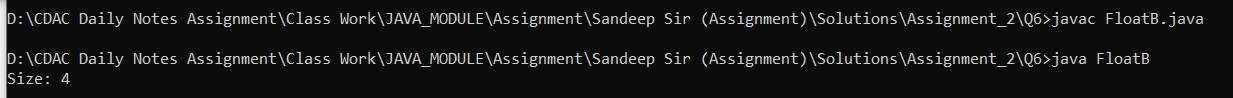
public static void main(String[] args) {

System.out.println("Size: " + Float.BYTES);

}

}

Output:



**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).

**Solution:**

public class FloatC{

public static void main(String[] args) {

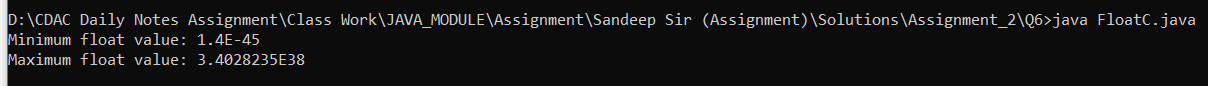
System.out.println("Minimum float value: " + Float.MIN\_VALUE);

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

}

}

Output:



**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)).

**Solution:**

public class floatD {

public static void main(String[] args) {

float number = 145.45f;

String str = Float.toString(number);

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

}

}

Output:



**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)).

**Solution:**

public class floatE {

public static void main(String[] args) {

String strNumber = "145.45";

float number = Float.parseFloat(strNumber);

System.out.println("String to float: " + number);

}

}

Output:



**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).

**Solution:**

public class floatF {

public static void main(String[] args) {

String strNumber = "Ab12Cd3";

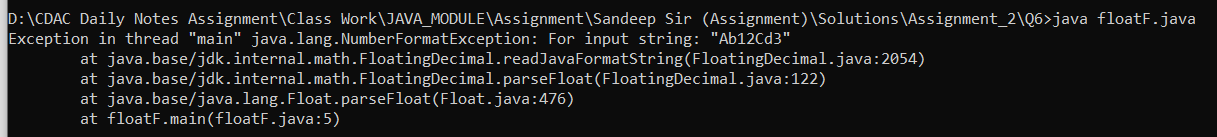
float number = Float.parseFloat(strNumber);

System.out.println("Converted value: " + number);

}

}

Output:



**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)).

**Solution:**

public class floatG {

public static void main(String[] args) {

float a = 114.2f;

float b = 656.6f;

float min = Float.min(a, b);

float max = Float.max(a, b);

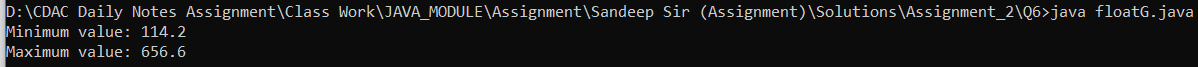
System.out.println("Minimum value: " + min);

System.out.println("Maximum value: " + max);

}

}

Output:



**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)).

**Solution:**

public class floatH {

public static void main(String[] args) {

String strNumber = "123.45";

Float floatWrapper = Float.valueOf(strNumber);

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

}

}

Output:



**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)).

Solution:

public class floatI {

public static void main(String[] args) {

float a = 112.3f;

float b = 984.5f;

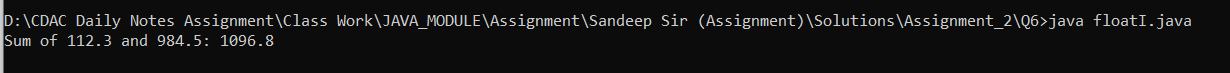
float sum = Float.sum(a, b);

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

}

}

Output:



**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)).

**Solution:**

public class floatJ {

public static void main(String[] args) {

float a = 112.2f;

float b = 556.6f;

float min = Float.min(a, b);

float max = Float.max(a, b);

System.out.println("Minimum value: " + min);

System.out.println("Maximum value: " + max);

}

}

Output:

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

**Solution:**

public class floatK {

public static void main(String[] args) {

float number = -25.0f;

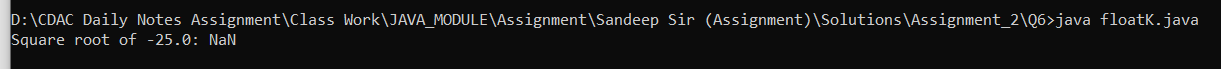
double sqrt = Math.sqrt(number);

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

}

}

Output:



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

**Solution**

public class floatL {

public static void main(String[] args) {

float a = 0.0f;

float b = 0.0f;

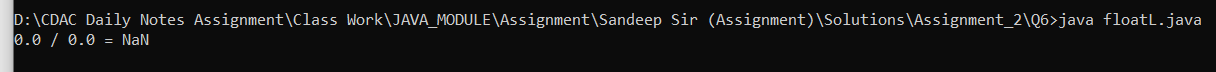
float result = a / b;

System.out.println("0.0 / 0.0 = " + result);

}

}

Output:



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

**Solution:**

public class floatM {

public static void main(String[] args) {

float floatValue = 123.45f;

int intValue = (int) floatValue;

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

double doubleValue = floatValue;

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

intValue = 123;

float floatFromInt = (float) intValue;

System.out.println("Int to float: " + floatFromInt);

doubleValue = 123.45;

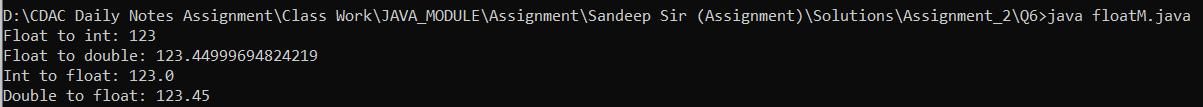
float floatFromDouble = (float) doubleValue;

System.out.println("Double to float: " + floatFromDouble);

}

}

Output:



#### ****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).

**Solution**

public class doubleB {

public static void main(String[] args) {

System.out.println("Number of bytes used to represent a double: " + Double.BYTES);

}

}

Output:



**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) **.**

**Solution:**

public class doubleC {

public static void main(String[] args) {

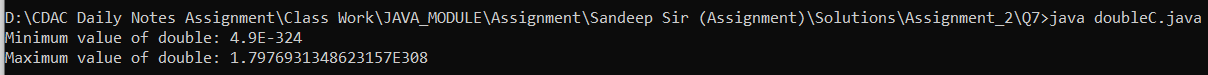
System.out.println("Minimum value of double: " + Double.MIN\_VALUE);

System.out.println("Maximum value of double: " + Double.MAX\_VALUE);

}

}

Output:



**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)).**

**Solution:**

**public class doubleD {**

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

**double number = 123.456;**

**String strNumber = Double.toString(number);**

**System.out.println("String representation of the double value: " + strNumber);**

**}**

**}**

**Output:**



**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)).

Solution:

public class doubeE{

public static void main(String[] args) {

String strNumber = "456.789";

double number = Double.parseDouble(strNumber);

System.out.println("Double representation of the string value: " + number);

}

}

Output:



**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).

**Solution:**

public class doubleF {

public static void main(String[] args) {

String strNumber = "Ab12Cd3"; // Declare method-local variable with the given value

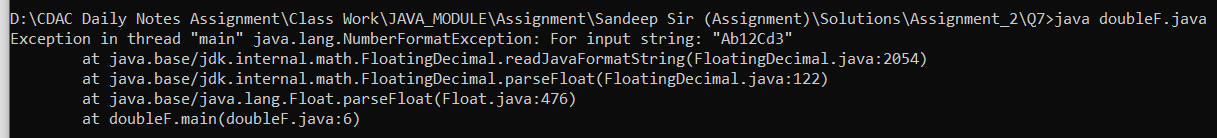
float number = Float.parseFloat(strNumber);

System.out.println("Converted value: " + number);

}

}

Output:



**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)).

**Solution:**

public class DoubleValueOfTest {

public static void main(String[] args) {

double number = 789.123;

Double wrapperDouble = Double.valueOf(number);

System.out.println("Double wrapper class object: " + wrapperDouble);

}

}

Output:



**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)).

**Solution:**

public class StringToDoubleValueOfTest {

public static void main(String[] args) {

String strNumber = "121.556";

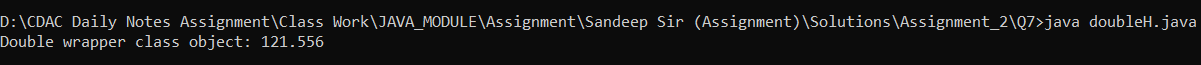
Double WDS= Double.valueOf(strNumber);

System.out.println("Double wrapper class object: " +WDS);

}

}

Output:



**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)).

Solution:

**public class doubleI {**

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

**double num1 = 112.3;**

**double num2 = 984.5;**

**double sum = Double.sum(num1, num2);**

**System.out.println("Sum of the two double values: " + sum);**

**}**

**Output:**

****

**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)).

**Solution:**

public class DoubleMinMaxTest {

public static void main(String[] args) {

double num1 = 112.2;

double num2 = 556.6;

double min = Double.min(num1, num2);

double max = Double.max(num1, num2);

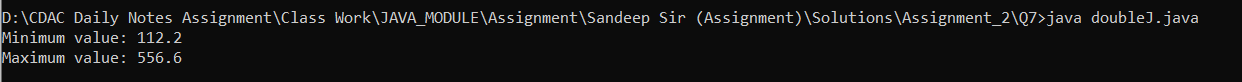
System.out.println("Minimum value: " + min);

System.out.println("Maximum value: " + max);

}

}

Output:



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

**Solution:**

public class doubleK {

public static void main(String[] args) {

double number = -25.0;

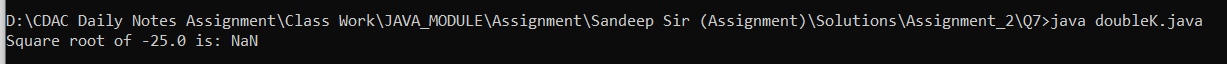
double sqrt = Math.sqrt(number);

System.out.println("Square root of " + number + " is: " + sqrt);

}

}

Output:



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

**Solution:**

**public class doubleL {**

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

**double num1 = 0.0;**

**double num2 = 0.0;**

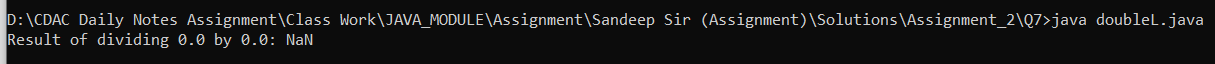
**double result = num1 / num2;**

**System.out.println("Result of dividing 0.0 by 0.0: " + result);**

**}**

**}**

**Output:**

****

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

**Solution:**

**public class doubleM {**

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

**double number = 123.456;**

**int intValue = (int) number;**

**float floatValue = (float) number;**

**long longValue = (long) number;**

**System.out.println("Double value: " + number);**

**System.out.println("Converted to int: " + intValue);**

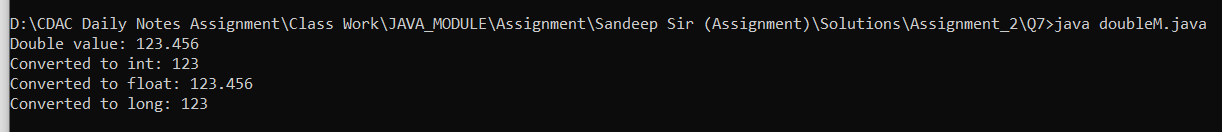
**System.out.println("Converted to float: " + floatValue);**

**System.out.println("Converted to long: " + longValue);**

**}**

**}**

Output:



#### ****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()).

Solution:

import java.util.Scanner;

public class Primitive {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter an integer: ");

int intVal = sc.nextInt();

System.out.print("Enter a double: ");

double doubleVal = sc.nextDouble();

System.out.print("Enter a boolean: ");

boolean boolVal = sc.nextBoolean();

// Converting using toString() method of wrapper classes

String intToString = Integer.toString(intVal);

String doubleToString = Double.toString(doubleVal);

String boolToString = Boolean.toString(boolVal);

System.out.println("\nConversion using toString() method:");

System.out.println("Integer as String: " + intToString);

System.out.println("Double as String: " + doubleToString);

System.out.println("Boolean as String: " + boolToString);

// Converting using valueOf() method of String class

String intValueOf = String.valueOf(intVal);

String doubleValueOf = String.valueOf(doubleVal);

String boolValueOf = String.valueOf(boolVal);

System.out.println("\nConversion using valueOf() method:");

System.out.println("Integer as String: " + intValueOf);

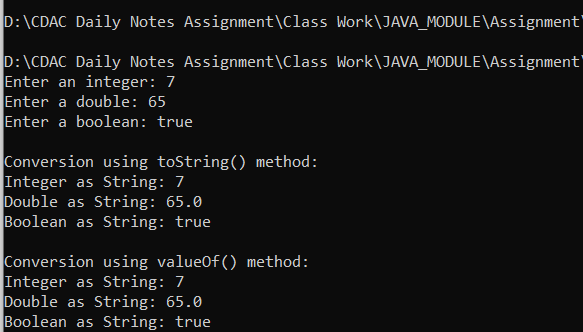
System.out.println("Double as String: " + doubleValueOf);

System.out.println("Boolean as String: " + boolValueOf);

}

}

Output:



PrimitiveTypesandStrings

Solution:

import java.util.Scanner;

public class PrimitiveTypesandStrings {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter an integer: ");

int intVal = sc.nextInt();

System.out.print("Enter a double: ");

double doubleVal = sc.nextDouble();

System.out.print("Enter a boolean: ");

boolean boolVal = sc.nextBoolean();

// Converting using toString() method of wrapper classes

String intToString = Integer.toString(intVal);

String doubleToString = Double.toString(doubleVal);

String boolToString = Boolean.toString(boolVal);

System.out.println("\nConversion using toString() method:");

System.out.println("Integer as String: " + intToString);

System.out.println("Double as String: " + doubleToString);

System.out.println("Boolean as String: " + boolToString);

// Converting using valueOf() method of String class

String intValueOf = String.valueOf(intVal);

String doubleValueOf = String.valueOf(doubleVal);

String boolValueOf = String.valueOf(boolVal);

System.out.println("\nConversion using valueOf() method:");

System.out.println("Integer as String: " + intValueOf);

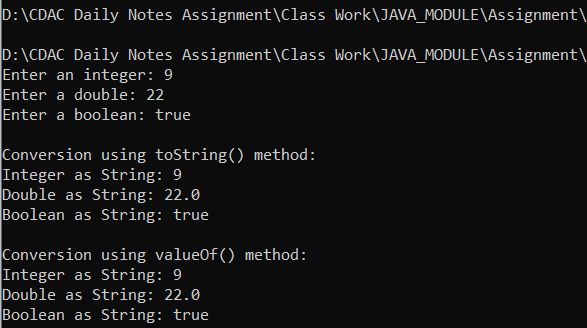
System.out.println("Double as String: " + doubleValueOf);

System.out.println("Boolean as String: " + boolValueOf);

}

}

Output:



#### ****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).

Solution:

public class DefaultValues {

// Instance variables (have default values)

byte byteVal;

short shortVal;

int intVal;

long longVal;

float floatVal;

double doubleVal;

char charVal;

boolean booleanVal;

// Static variables (also have default values)

static byte staticByteVal;

static short staticShortVal;

static int staticIntVal;

static long staticLongVal;

static float staticFloatVal;

static double staticDoubleVal;

static char staticCharVal;

static boolean staticBooleanVal;

public static void main(String[] args) {

// Create an instance to check instance variable default values

DefaultValues obj = new DefaultValues();

// Display default values of instance variables

System.out.println("Default values of instance variables:");

System.out.println("byte: " + obj.byteVal);

System.out.println("short: " + obj.shortVal);

System.out.println("int: " + obj.intVal);

System.out.println("long: " + obj.longVal);

System.out.println("float: " + obj.floatVal);

System.out.println("double: " + obj.doubleVal);

System.out.println("char: [" + obj.charVal + "]"); // Displays an empty space

System.out.println("boolean: " + obj.booleanVal);

// Display default values of static variables

System.out.println("\nDefault values of static variables:");

System.out.println("byte: " + staticByteVal);

System.out.println("short: " + staticShortVal);

System.out.println("int: " + staticIntVal);

System.out.println("long: " + staticLongVal);

System.out.println("float: " + staticFloatVal);

System.out.println("double: " + staticDoubleVal);

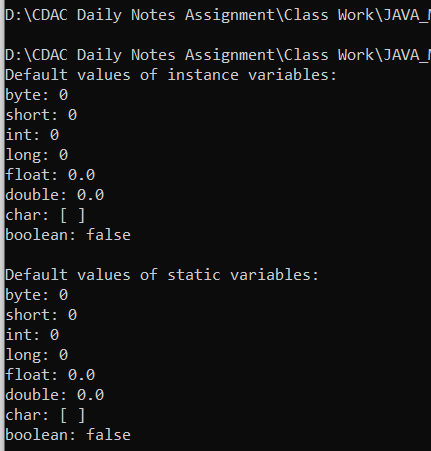
System.out.println("char: [" + staticCharVal + "]");

System.out.println("boolean: " + staticBooleanVal);

}

}

Output:



#### ****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).

Solution:

import java.util.Scanner;

public class ArithmeticOperations {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input two integers

System.out.print("Enter the first integer: ");

int num1 = scanner.nextInt();

System.out.print("Enter the second integer: ");

int num2 = scanner.nextInt();

// Input the operator

System.out.print("Enter an arithmetic operator (+, -, \*, /): ");

char operator = scanner.next().charAt(0);

// Perform the operation using switch-case

switch (operator) {

case '+':

System.out.println("Result: " + (num1 + num2));

break;

case '-':

System.out.println("Result: " + (num1 - num2));

break;

case '\*':

System.out.println("Result: " + (num1 \* num2));

break;

case '/':

if (num2 != 0) {

System.out.println("Result: " + (num1 / num2));

} else {

System.out.println("Error: Division by zero is not allowed.");

}

break;

default:

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

}

scanner.close();

}

}

Output:

