**String formatting:-**

The **java string format()** method returns a formatted string using the given **locale**, specified **format string** and **arguments**.We can concatenate the strings using this method and at the same time, we can format the output concatenated string.

There are two type of **string format()** method:

* public static String **format**(Locale loc, String form, Object… args)
* public static String **format**(String form, Object… args)

**Examples:-**

class myprogram {

    public static void main(String args[])

    {

        String str = "String Formatting";

        // Concatenation of two strings

        String str1 = String.format("My Company name is %s", str);

        // Output is given upto 8 decimal places

        String str2 = String.format("My answer is %.8f", 47.65734);

        // between "My answer is" and "47.65734000" there are 15 spaces

        String str3 = String.format("My answer is %15.8f", 47.65734);

        System.out.println(str1);

        System.out.println(str2);

        System.out.println(str3);

    }

}

**EXAMPLE2// concatenation of arguments to the string**

class pro2 {

    public static void main(String args[])

    {

        String str1 = "Muhammad Saqib";

        String str2 = "programming";

        //%1$ represents first argument, %2$ second argument

        String gfg2 = String.format("My Company name" +

                 " is: %1$s, %1$s and %2$s", str1, str2);

        System.out.println(gfg2);

    }

}

**EXAMPLE3:-e// format() method and left padding using**

class Gfg3 {

    public static void main(String args[])

    {

        int num = 7044;

        // Output is 3 zero's("000") + "7044",

        // in total 7 digits

        String gfg3 = String.format("%07d", num);

        System.out.println(gfg3);

    }

}

**EXAMPLE4// PrintWriter format(String, Object) method**

import java.io.\*;

import java.util.\*;

class GFG {

    public static void main(String[] args)

    {

        try {

            // Get the parameters

            Locale locale = Locale.getDefault();

            String arg1 = "CUI";

            String arg2 = "Programming Fundamentals";

            String format = "A Computer Science "

                            + "Portal  %1$s, %1$s and %2$s";

            // Create a PrintWriter instance

            PrintWriter writer

                = new PrintWriter(System.out);

            // print the formatted string

            // to this writer using format() method

            writer.format(format, arg1, arg2);

            writer.flush();

        }

        catch (Exception e) {

            System.out.println(e);

        }

    }

}

**Boolean formatting:-**

To format boolean values, we use the %b format. It works the following way: If the input value is true, the output is true. Otherwise, the output is false.

So, if we do:

|  |  |
| --- | --- |
|  | System.out.printf(**"%b%n"**, **null**); |
|  | System.out.printf(**"%B%n"**, **false**); |
|  | System.out.printf(**"%B%n"**, 5.3); |
|  | System.out.printf(**"%b%n"**, **"random text"**); |
|  |  |

**Output:-FALSE ,FALSE,TRUE, true**

Basically it is used to print output in formatted manner and starts with %

**%b or %B are used for boolean**

***Eg 1:***

int a=5,b=0,c=-5;

System.out.println("%b %b %b",a,b,c);

**output: true true true**

**all integers are treated as true in JAVA**

***Eg 2:***

String a="",b=null,c="cat";

System.out.println("%b %b %b",a,b,c);

**output: true false true**

**for string only null is treated as false**

***Eg3:-***

float floatVar=1.0f;

int intVar=1;

String stringVar="hi";

boolean boolVar=false;

System.out.printf("The value of the float " +

"variable is %f, while " +

"the value of the " +

"boolean variable is %b, " +

"and the string is %s",

floatVar, boolVar, stringVar);

Boolean keyword is a primitive data type. It is used to store only two possible values, either true or false. It specifies 1-bit of information and its "size" can't be defined precisely.

The boolean keyword is used with variables and methods. Its default value is false. It is generally associated with conditional statements.

public class **BooleanExample2** {

    public static void main(String[] args) {

        int num=7;

        boolean flag=false;

          for(int i=2;i<num;i++)

          {  if(num%i==0)

              {

                  flag=true;

                          break;

              }

          }

        if(flag)  {

          System.out.println("Not prime");

        }  else

        {

           System.out.println("prime");

        }

}

}

**Date and Time formatting:-**

Java SimpleDateFormat and DateFormat classes are used for date formatting. It is mostly used where we need to display or utilize the date and time functionality of Java..

Java does not have a built-in Date class, but we can import the java.time package to work with the date and time API. The package includes many date and time classes. For example:

|  |  |
| --- | --- |
| Class | Description |
| LocalDate | Represents a date (year, month, day (yyyy-MM-dd)) |
| LocalTime | Represents a time (hour, minute, second and nanoseconds (HH-mm-ss-ns)) |
| LocalDateTime | Represents both a date and a time (yyyy-MM-dd-HH-mm-ss-ns) |
| DateTimeFormatter | Formatter for displaying and parsing date-time objects |

**Example1:-**

import java.time.LocalDateTime; // Import the LocalDateTime class

import java.time.format.DateTimeFormatter; // Import the DateTimeFormatter class

public class MyClass {

public static void main(String[] args) {

LocalDateTime myDateObj = LocalDateTime.now();

System.out.println("Before formatting: " + myDateObj);

DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

String formattedDate = myDateObj.format(myFormatObj);

System.out.println("After formatting: " + formattedDate);

}

}

**Example2:-**

import java.time.LocalTime; // import the LocalTime class

public class MyClass {

public static void main(String[] args) {

LocalTime myObj = LocalTime.now();

System.out.println(myObj);

}

}

**Example3:-**

import java.time.LocalDateTime; // import the LocalDateTime class

public class MyClass {

public static void main(String[] args) {

LocalDateTime myObj = LocalDateTime.now();

System.out.println(myObj);

}

}

**Example4:-**

import java.time.LocalDate; // import the LocalDate class

public class MyClass {

public static void main(String[] args) {

LocalDate myObj = LocalDate.now(); // Create a date object

System.out.println(myObj); // Display the current date

}

}