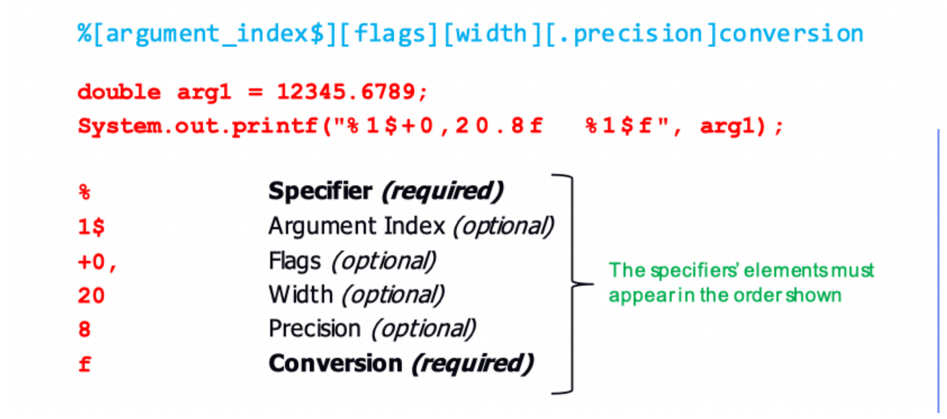
There are three standard stream objects, all of which are managed by the System class (a final class from java.lang package)
– System.out
The standard output stream object enables a program to output information to the screen
– System.err
The standard error stream object enables a program to output error messages to the screen
– System.in
The standard input stream object enables a program to input bytes from the keyboard
println is a method of PrintStream class called by the System.out object that is used for console output
oThe data to be output is given as an argument in parentheses
o A plus sign (+) is used to connect more than one item
o Every call of println ends a line of output
System.out.println("This is " + "one" + " argument");
print It is like println, but it does not end a line of output
System.out.print("This is " + "one" + " argument");
printf is another method of PrintStream class called by the System.out object that is used to display formatted output have multiple arguments that are separated by a comma (,)
System.out.printf("format-string" [, arg1, arg2, …]);
Format specifiers are special characters that format the arguments
Begin with (%) and end with a converter character (indicating the type of argument)
In betw een (%) and the converter, you can have optional flags and specifiers
The args is a list of the variables to be printed using that format



```
%[argument_index$][flags][width][.precision]conversion

double arg1 = 12345.6789;
System.out.printf("%1$+0,20.8f   %1$f", arg1);
```

%	**Specifier *(required)***
1$	Argument Index *(optional)*
+0,	Flags *(optional)*
20	Width *(optional)*
8	Precision *(optional)*
f	**Conversion *(required)***

The specifiers' elements must appear in the order shown

Format Specifier

–the Optional Elements
🚍 **Argument index** – is a positive integer indicating the position of the argument in the argument list. The first "1$", second "2$", …
🚍 **Flags** – a set of formatting options (some flags cannot be used with certain other flags or with certain conversions):
o - : left-justify (default is to right-justify)
o + : output a plus (+) or minus (-) sign for a numerical value
o 0 : forces numerical values to be zero-padded (default is blank padding)

o , : comma grouping separator (for numbers > 1000)
o (: format negative numbers with parentheses rather than a minus sign
🚍 **Precision** – the number of digits of precision when outputting floating-point values or the length of a substring to extract from a String

Packages the Java classes are arranged into groups called packages
Import Declaration To use a class from a package, you could use its fully qualified name
import java.util.Scanner;
To import all classes in a particular package, you use the (*) wildcard character
import java.util.*;
Scanner that enables a program to read data from a keyboard or a file on disk
The following import statement must be included in your program
import java.util.Scanner;
Must instantiate a Scanner object and associate it with a data source
Scanner keyboard = new Scanner(System.in);
System.in (the standard input object), will be used as our data source, which by default is tied to the keyboard
Scanner's object translates bytes of information typed by a user into types that can be used in a program
Scanner Method

Method	Description
`nextBoolean()`	Reads a `boolean` value from the user
`nextByte()`	Reads a `byte` value from the user
`nextDouble()`	Reads a `double` value from the user
`nextFloat()`	Reads a `float` value from the user
`nextInt()`	Reads a `int` value from the user
`nextLine()`	Reads a `String` value from the user
`nextLong()`	Reads a `long` value from the user
`nextShort()`	Reads a `short` value from the user

The Math class provides a collection of static fields and methods that enable you to perform common mathematical calculations
It is part of the java.lang package, so we do not need to use the import statement
Math fields for common mathematical constants
o Math.PI – (3.1415…)
o Math.E – e, the base of the natural logarithm system (2.7182…)

Method	**Description**
Math.abs()	It will return the Absolute value of the given value.
Math.max()	It returns the Largest of two values.
Math.min()	It is used to return the Smallest of two values.
Math.round()	It is used to round of the decimal numbers to the nearest value.
Math.sqrt()	It is used to return the square root of a number.
Math.cbrt()	It is used to return the cube root of a number.
Math.pow()	It returns the value of first argument raised to the power to second argument.
Math.ceil()	It is used to find the smallest integer value that is greater than or equal to the argument or mathematical integer.
Math.floor()	It is used to find the largest integer value which is less than or equal to the argument and is equal to the mathematical integer of a double value.
Math.log()	It returns the natural logarithm of a double value.
Math.random()	It returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.

🚍 **The Math class** provides a facility to generate pseudo-random numbers
A sequence of values produced by a complex algorithm
Generates a random double value greater than or equal to 0.0 and less than 1.0 (0 <= Math.random() < 1.0)
🚍 Example:
(int)(Math.random() * 10) Returns a random integer between 0 and 9.
🚍 **The Random class** also produces pseudo-random numbers Class Random is located in (java.util package)
class Random can produce random boolean, byte, float, double, int, long and Gaussian values
Random method nextInt receives an int argument and returns a value from 0 up to the argument's value (excluded)
obj.nextInt(10) Returns a random integer between 0 and 9.
50 + obj.nextInt(50) Returns a random integer between 50 and 99.
a + obj.nextInt(b). Returns a random number between a and a + b, excluding a + b.