Object-Oriented Programming Fundamentals

Lecture/Workshop (Week 4)

Formatted output: printf()

Task 1

Write a program using while loops to output the first 10 integers on the first line and their cubes on the second line. ***Note that we do not have an easy way to format the output yet…. We will learn about that next…***

```
  1    2    3    4    5    6    7    8    9   10
  1    8   27   64  125  216  343  512  729 1000
```

You can copy sample files from the library area and check answers to the rest of this worksheet when you again have access to your latcs account. You should make an appropriate directory and *cd* to that directory to copy the files:

`cp   /home/1st/csilib/cse1oof/ws04/*   .`   ***don't forget the dot***

Formatting output

There are several ways of formatting output in Java - one way is to use the **`printf()`** method (which can be called with **`System.out`**). Note the `f` in **`printf`** means *formatted*

printf

The **`printf()`** method takes one or more arguments

```
    printf(formatstring, otherarguments)
```

The first argument is the output string

Any other arguments are a list of values to be put into the output string at certain given places

So, the output string (first argument) contains

- Ordinary or fixed text and
- Format Specifiers (the places to put the other arguments)

The format specifiers also define how an argument will be formatted when it is placed into the output string.

Example1WS04 *(Have you copied all examples from the library area to run them…)*

`cp   /home/1st/csilib/cse1oof/ws04/*   .`   *don't forget the dot*

```
String dogName = "Harry";
int dogAge = 6;
double weight = 60;

System.out.printf("My dog %s is %d years old and weighs %.2f kgs\n",
                  dogName, dogAge, weight);
```

Produces the output

```
My dog Harry is 6 years old and weighs 60.00 kgs
```

In this example, %s, %d and %.2f are the format specifiers for the last 3 arguments, **`dogName`** (a **`String`**), **`dogAge`** (an **`int`**) and **`weight`** (a **`double`**).

%s is a format specifier related to a **`String`**, %d is a format specifier related to an **`int`**, and %f is a format specifier related to a **`double`**

Format specifiers

Start with a % symbol

End with a conversion character (e.g. s, d or f) that determines what type of value will be output

There may be *additional optional* information between the % and the conversion character – we will look at some examples below

Common Conversions

b	– boolean
s	– String
c	– character
d	– decimal integer
o	– octal integer
x	– hexadecimal integer
f	– floating point number

Additional information – an Argument Index

The first additional optional information is the argument index (a number followed by a $ character – the number is the position of the argument in the list of values)

This allows us to use one of the arguments in the list multiple times or out of order in the output (see the code below in *Example1WS04.java* from the library area)

```
String dogName = "Harry";
int dogAge = 6;
double weight = 60;

System.out.printf("My dog %1$s is %2$d years old. I love %1$s.\n",
                  dogName, dogAge, weight);
```

Produces the output

```
My dog Harry is 6 years old. I love Harry.
```

(Note we don't have to use all the values in the argument list)

Task 2 (in Task2WS04.java)

```
String name = "Suzie";
String sport = "squash";
char grade = 'B';
int age = 22;
double courtHire = 19.80;
```

Given the declarations above, write `printf()` statements to produce the output below.

```
Suzie plays squash.


22 year old Suzie plays B grade squash.


Court hire is 19.800000. Suzie must pay 19.800000 to play.
```

Flags

The second additional optional information is called a flag – certain characters are used (between the % and the conversion character) as flags which have set meanings to format the output

Common flags

'-'	left justifies (so -10 say would left justify the argument in a width of 10 spaces)
'+'	means a sign will be output for all numbers
' '	means positive numbers will have a leading space
'0'	pad number with zeroes
'('	puts negative numbers in brackets

Example 2 (in Example2WS04.java)

```
int number1 = 23;
int number2 = -145;
double number3 = 12321.345;
double number4 = -99.9;

System.out.printf("1: %d, 2: %d, 3: %f, 4: %f\n",
                  number1, number2, number3, number4);
System.out.printf("2: %-10d, 2: %-10d, 3: %-10f, 4: %-10f\n",
                  number1, number2, number3, number4);
System.out.printf("3: %+d, 2: %+d, 3: %+f, 4: %+f\n",
                  number1, number2, number3, number4);
System.out.printf("4: % d, 2: % d, 3: % f, 4: % f\n",
                  number1, number2, number3, number4);
System.out.printf("5: %010d, 2: %010d, 3: %010f, 4: %010f\n",
                  number1, number2, number3, number4);
System.out.printf("6: %,d, 2: %,d, 3: %,f, 4: %,f\n",
                  number1, number2, number3, number4);
System.out.printf("7: %(d, 2: %(d, 3: %(f, 4: %(f\n",
                  number1, number2, number3, number4);
System.out.printf("8: %+(d, 2: %+(d, 3: %+(f, 4: %+(f\n",
                  number1, number2, number3, number4);
System.out.printf("9: %(+d, 2: %(+d, 3: %(+f, 4: %(+f\n",
                  number1, number2, number3, number4);
```

Example 2 output

```
1: 23, 2: -145, 3: 12321.345000, 4: -99.900000
2: 23        , 2: -145      , 3: 12321.345000, 4: -99.900000
3: +23, 2: -145, 3: +12321.345000, 4: -99.900000
4:  23, 2: -145, 3:  12321.345000, 4: -99.900000
5: 0000000023, 2: -000000145, 3: 12321.345000, 4: -99.900000
6: 23, 2: -145, 3: 12,321.345000, 4: -99.900000
7: 23, 2: (145), 3: 12321.345000, 4: (99.900000)
8: +23, 2: (145), 3: +12321.345000, 4: (99.900000)
9: +23, 2: (145), 3: +12321.345000, 4: (99.900000)
```

Width and precision

Width is the minimum number of spaces to use when outputting a value

Precision is the number of decimal places after the decimal point in floating point numbers

For strings, precision is the maximum number of characters to be output

Example 3 (in Example3WS04.java)

```
String s1 = "Hello Out There!";
String s2 = "hi";
int n1 = 13;
double d1 = 123.4567;

System.out.printf("s1: %s, s2: %s, n1: %d d1: %f\n",
                  s1, s2, n1, d1);
System.out.printf("s1: %10s, s2: %10s, n1: %10d d1: %10f\n",
                  s1, s2, n1, d1);
System.out.printf("s1: %-10s, s2: %-10s, n1: %-10d d1: %-10f\n",
                  s1, s2, n1, d1);
System.out.printf("s1: %10.3s, s2: %10.3s, n1: %10d d1: %10.3f\n",
                  s1, s2, n1, d1);
System.out.printf("s1: %3.10s, s2: %3.10s\n", s1, s2);
```

Example 3 output

```
s1: Hello Out There!, s2: hi, n1: 13 d1: 123.456700
s1: Hello Out There!, s2:         hi, n1:         13 d1: 123.456700
s1: Hello Out There!, s2: hi        , n1: 13         d1: 123.456700
s1:        Hel, s2:         hi, n1:         13 d1:    123.457
s1: Hello Out , s2:  hi
```

Task 3 (in Task3WS04.java)

Rewrite your program to output the first 10 integers on the first line and their cubes on the second line. Use the **`printf()`** statement to format the output.

```
    1    2    3    4    5    6    7    8    9   10
    1    8   27   64  125  216  343  512  729 1000
```

Task 4 (in Task4WS04.java)

What is output by each print statement below for the following declarations?

```
int anInt = 12;
double aDouble = 34.5678;
String aString = "Howdy!";
int anotherInt = 2 * anInt;
```

```
a) System.out.println(anInt + " " + aDouble + " " + aString);
```

```
b) System.out.printf("This is an int: %d.", anInt);
```

```
c) System.out.printf("This is an int: %d.\n", anInt);
```

```
d) System.out.printf("%d, %d\n", anInt, anInt);
```

```
e) System.out.printf("%d, %d\n", anInt, anotherInt);
```

```
f) System.out.printf("%1$d, %2$d, %1$d, %3$d\n",
                     anInt, anotherInt, anInt * 3);
```

g) `System.out.printf("%10d, %1d\n", anInt, anotherInt);`

h) `System.out.printf("%10d, %10d|\n", anInt, anotherInt);`

i) `System.out.printf("%-10d, %-10d|\n", anInt, anotherInt);`

j) `System.out.printf("%010d, %010d|\n", anInt, anotherInt);`

k) `System.out.printf("%10f, %10f|\n", (double) anInt, aDouble);`

l)
```
System.out.printf("%10.2f, %10.2f|\n",
                  (double) anInt, aDouble);
```

m)
```
System.out.printf("%10.2f, %10.2f|\n",
                  (double) anInt * 10000, aDouble * 10000);
```

n)
```
System.out.printf("%,10.2f, %,10.2f|\n",
                  (double) anInt * 10000, aDouble * 10000);
```

```
o) System.out.printf("%,10.2f, %,10.2f|\n",
                    (double) anInt * 1000000, aDouble * 1000000);
```

```
p) System.out.printf("%+10f, %+10f|\n",
                     (double) anInt, aDouble);
```

```
q) System.out.printf("%10s, %10s, %10s\n",
                      aString, "Hi There", "Hello");
```

```
r) System.out.printf("%3$10s, %1$10s, %2$10s\n",
                      aString, "Hi There", "Hello");
```

```
s) System.out.printf("%2$10s, %3$10s, %1$10s\n",
                      aString, "Hi There", "Hello");
```

```
t) System.out.printf("%2$-10s, %3$-10s, %1$-10s\n",
                      aString, "Hi There", "Hello");
```

<u>**Wanting help?**</u> Check put the Codehub session time in LMS.