# Practical-Exam-01: Intro to Java, Conditions, and Loops

**Due** 16 Aug by 12:00 **Points** 100 **Available** 16 Aug at 10:15 - 16 Aug at 12:00 about 2 hours

This assignment was locked 16 Aug at 12:00.

## Introduction

**This session ends 12 pm**

**Make sure to complete your submission in time.**

### Submission

- Create the repository on the SVN server
- You must checkout only the folder practical-exam-01 from your server
- No other folders from your SVN will be allowed during the practical exam.

```
svn mkdir -m "first assessment commit" --parents https://version-control.adelaide.edu.au/svn/<YOUR-ID>/2019/s2/fcs/week-03/practical-exam-01

```

## Assessment

- **This is a practical exam - your work must be entirely your own.**
- Marks for this practical exam contribute 2 marks to the final mark of this course.
- Marks will be awarded later by your workshop supervisor (30%) and websubmission marker (70%).
- **Due date: the end of this session (12 pm).**
- **Do Not Forget** To Submit on **WebSubmission**
- **Late penalties:** Only work submitted during your enrolled practical session from a Linux system in the practical lab will be marked.

Regarding functional marks, please consider:

```
(1) only codes that can compile will be marked;
(2) only codes that are in the suggested directory will be marked;
(3) only codes submitted to SVN and WebSubmission before the deadline will be marked;
(4) only codes containing your signature on the top of the file will be marked by tutors;
(5) you will have your markers decreased in 3 points if *.class file present in your folders;
```

## Signature on your files

Note that all your coding files must contain on the top of it this information:

```
//================================
```

```
// Foundations of Computer Science
// Student: you name
// id: your id
// Semester:
// Year:
// Practical Exam Number:
//==================================
```

## Note

- To acquire full marks **(1)** all your functionalities must be working perfectly, **(2)** your code has to be well and proportionally commented, and **(3)** your code must follow correct indentation (4 spaces for every new indentation level) and **(4)** you have to use all the content from latest lectures.
- We argue that you are not just asked to solve a problem but use the more sophisticated way to solve it. For instance, you can solve a problem using ten variables, but it will always be better to solve the same problem with an array.

# Practical Exam 01

## Part 01 - Basic Programming

### Problem 01

In this problem you are required to design, implement and code a program that prints the following information on the command line console:

```
Hello World!
```

Constraints:

```
file path: ..practical-exam-01/problem-01/Problem.java

class name: Problem
method to be implemented: public static void main(String [] args)
```

### Problem 02

In this problem you are required to design, implement and code a program that prints on the screen the count down from 101 to 89. The expected output is:

```
[101, 100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89]
```

Constraints:

```
file-path: ..practical-exam-01/problem-02/Problem.java

class name: Problem
method to be implemented: public static void main(String [] args)

Note that, there are multiple ways to solving this problem. Please consider that
```

```
for auto-prac-marker all the solutions will be awarded with the same weight. However,
for tutor marking repetition structures will be awarded with total, and hard-coded
solutions will be lightly weighted.
```

## Problem 03

In this problem you are required to design, implement and code a program that:

- The code must count from 1 to 40, as integer numbers;
- Print the number using the following marks:
  - Use the marker `*` for numbers [1, 10);
  - Use the marker `=` for numbers [10, 20);
  - Use the marker `x` for numbers [20, 30);
  - Use the marker `o` for numbers [30, 40];

Expected output:

```
*
**
***
****
*****
******
*******
********
*********
==========
===========
============
=============
==============
===============
================
=================
==================
===================
xxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxx
ooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooo
```

```
ooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooo
```

Constraints

```
file-path: ..practical-exam-01/problem-03/Problem.java

class name: Problem
method to be implemented: public static void main(String [] args)

Note that, there are multiple ways to solving this problem. Please consider that
for auto-prac-marker all the solutions will be awarded with the same weight. However,
for tutor marking repetition structures will be awarded with total, and hard-coded
solutions will be lightly weighted.
```

## Problem 04

In this problem you are required to design, implement and code a program that:

- The code must count down from 50 to 3, as integer numbers;
- You should only print multiples of the number 3 (i.e. the number can be divided by 3 with no remainder);
- Print the number using the following marks:
  - Use the marker `*` for numbers [1, 10);
  - Use the marker `=` for numbers [10, 20);
  - User the marker `3` if the number is 33;
  - Use the marker `x` for numbers [20, 30);
  - Use the marker `o` for numbers [30, 50];

Expected output:

```
ooooooooooooooooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooooooooooooooooo
33333333333333333333333333333333
ooooooooooooooooooooooooooooooo
xxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxx
==================
===============
============
*********
******
***
```

Constraints

```
file-path: ..practical-exam-01/problem-04/Problem.java

class name: Problem
method to be implemented: public static void main(String [] args)

Note that, there are multiple ways to solving this problem. Please consider that
for auto-prac-marker all the solutions will be awarded with the same weight. However,
for tutor marking repetition structures will be awarded with total, and hard-coded
solutions will be lightly weighted.
```

## Problem 05

In this problem, you are given several lines of a code that was already functional. However, this code was shuffle by some file transferring issue that occurred in our servers. You are asked to re-organize the code in order to make it functional once again, where the expected output is:

```
This is the last line of the code!
i: 1 j: 5 k: 5 - value 25
i: 2 j: 5 k: 5 - value 50
i: 3 j: 5 k: 5 - value 75
i: 4 j: 5 k: 5 - value 100
i: 5 j: 5 k: 5 - value 125
i: 5 j: 5 k: 6 - value 150
i: 5 j: 5 k: 7 - value 175
i: 5 j: 5 k: 8 - value 200
i: 5 j: 5 k: 9 - value 225
This is the first line of the code!
```

The lines of code for you to re-organize are:

```
System.out.println("This is the first line of the code!");
System.out.println("This is the last line of the code!");
public static void main(String [] args){
int ths = 25;
for (int k = j; k < 10; k ++){
}
for(int i = 0; i < 10; i++){
for(int j = i; j < 10; j++){
int value = i * j * k;
public class Problem{
if(value % ths == 0 ){
if(value != 0){
System.out.print(" j: " + j);
System.out.print("i: " + i);
System.out.println(" k: " + k + " - value " + value);
}
}
}
}
}
}
```

Constraints

```
file-path: ..practical-exam-01/problem-05/Problem.java


Note that, the following code was delivered to you not following any identation.
Consider that for tutor marking, we will assess whether you apply good Java
conventions to the file.
```

| Practical Exam 01 | | | | | |
|---|---|---|---|---|---|
| **Criteria** | **Ratings** | | | | **Pts** |
| Functional | **60.0 Pts**<br>**Excellent**<br>Your code (1) perform all the functions correctly, (2) use latest concepts learned in class, (3) has a clear, creative and sophisticated way of solving the problem. | **50.0 Pts**<br>**Good**<br>Your code (1) perform all the functions correctly, (2) use concepts learned in class, (3) has a clear way of solving the problem. | **40.0 Pts**<br>**Fair**<br>Your code (1) perform almost all the functions correctly, (2) use concepts learned in class, (3) has a way of solving the problem. | **0.0 Pts**<br>**No marks**<br>You code (1) does not exist. | 60.0 pts |
| Code Style | **40.0 Pts**<br>**Excellent**<br>Your code (1) has the right proportion of comments and place line and block comments correctly, (2) follow correct indentation every new indentation level, (3) has good variable naming, (4) has clear organization between tabs and is easy to read. | **15.0 Pts**<br>**Good**<br>Your code (1) has useful comments and place line and block comments correctly, (2) follow indentation, (3) has good variable naming. | **5.0 Pts**<br>**Fair**<br>Your code (1) has comments, (2) has variables, (4) has clear organization. | **0.0 Pts**<br>**No marks**<br>You code (1) does not exist. | 40.0 pts |
| | | | | | Total points: 100.0 |