Subjects were recruited among several university in Lyon, France via a request oriented for student from any kind of studies. The requirements were to want to learn programming, have no skills in programming language and have the baccalaureate (it is the degree which people obtain at the end of high school). The request specified a range of time and ask for several questions about the candidate. Subjects come from Business and Economics university, Sciences and Technology university and Engineering school.

Thanks to the answers of the subject’s background, it was possible to divide them into groups were the characterises are equivalents.

## Experiment Design

To summarize, the use of an IDE or a TE are the independent variables. The dependent variables have been chosen (Time to implement, Number of lines of code, number of errors, number of blocking error).

\*\*Randomization.\*\* The selection of the subjects will be representative of students in bachelor or master degree with no knowledge of programming. The assignment to each treatment (using an IDE or a TE) is selected randomly. It is a completely randomized design experiment.

\*\*Blocking.\*\* To avoid to compare two tools but to type of tool. To minimize the effect of the efficiency (or inefficiency) of an environment, the subject will be divided in 4 groups: 2 will do the experiment with a different IDE (Eclipse, IntelliJ) and 2 with 2 different text editors (Gedit, Emacs).

\*\*Balancing.\*\*The experiment use a balanced design, each group will be composed by 10 persons. In that way, there are 20 persons in each category of tool.

## Instrumentation

\*\*Experiment object.\*\* The objects of the experiment is the code produce by the subjects and the submission of this code.

\*\*Guidelines.\*\* All the subjects will have the same theorical formation in Java. After, they will have a description of the features of their tools respectively.

\*\*Measurement instruments.\*\*In this experiment it is important to validate the subject’s submission. That’s way there is program which runs tests to be sure all the functional requirements are conformed to the specification. Another tool, called CLOC, will count the number of line in the program files.

## Data collection

The experiment will begin with a three days of intensive Java courses. All the subjects will be together and learn in the same way. After, they will be divided into the groups and each group will have a two days training of the features of their tool.

In the end, the subjects have to implement a naval battle game with a 5\*5 board, one 1\*3 boat and two 1\*2 boat. This project will be divided in some task describe below. For each task, the suject should submit his source code. This one will be analysed to count the number of lines and the number of errors.

At the end of the experiment, subjects will answer to a questionnaire about their feeling during the experiment.

### Task 1

The first one is a simple task. The subjects have to create naval battle game in command line. This version includes only one player who play against the computer. The computer shoots randomly to the enemy board. An example of the result expected (B = Boat, M = Missed, H = Hit)

```{shell}

$> shoot(1,2)

Missed!

Enemy missed!

Your board:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B | B | B |  |  |
|  |  |  |  | M |
| B | B |  | B |  |
|  |  |  | B |  |
|  |  |  |  |  |

Enemy board:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| M |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

$> shoot(5,5)

Hit!

Enemy hit!

Your board:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B | B | B |  |  |
|  |  |  |  | M |
| B | H |  | B |  |
|  |  |  | B |  |
|  |  |  |  |  |

Enemy board:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| M |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  | H |

```

The program has to know who has won the game.

### Task 2

This task is more about an algorithmic problem. Subjects have to implement a more intelligent computer. To validate this task, the artificial intelligent must have a win rate of 95% against a random intelligence.

### Tasks 3

The last task is to create a user interface with Java FX. The interface must display only the boards of the player. At the beginning, the player arrives to a menu where he can select a difficulty (“easy” = ramdom AI or “medium”=AI implemented in task 2).

## Analysis procedure

The analysis procedure includes a quantitative analysis and a short qualitative analysis. The hypotheses will be tested with the quantitative data. The qualitative data increase the comprehension of the results.

### Quantitative analysis

The analyses of the dependent variables will be performed for each task and across all tasks. It will be performed descriptive statistic and t-test for all the dependent variables (T, LOC, C and C’).

The risk level to reject or no the null hypothesis is 0.05. For each set of tests, we will provide the calculated p-value.

Thanks to the number of subjects it is possible to do an Analysis of Covariance, also called ANCOVA for the time, the LOC and the correctness for all the tasks and subjects. We have a total 120 data points for each dependent variable (40 subjects and three tasks).

### Qualitative analysis

The analysis of the answer the questionnaire about the feeling may help us to understand the results of the experiment and the increase the strength of the results. The questionnaire takes around 50 minutes to answer all the questions. For all the questions there is a nominal choice (like Does IDE help you to create user interface? Answers can be: Not at all, Few times, Many times, All the time). In that way, for each question, there are a distribution of answers which can confirm the results of the statistical experiment.

## Evaluation of validity

As it says before, after each submission the code will be analyse and test. If a submission is not sufficient or doesn’t fit with the specifications, the subject has to do the necessary changes (the time to do it will be added to the initial time). For example, to test the \*Task 2\* the subject’s AI will play around 1 000 games against a random AI, and the subject’s one has to win more than 950 times. For \*Tasks 3\*, the subjects have an GUI specification with example of what it is expected. The subjects will be timed for each work. These requirements are important to be able to compare the final solution in term of effort. The time allocated for each task is not fixed because some of them can be faster or slower than the others.

Each subtask will be rated as \*acceptable\* or \*unacceptable\* according to the following criteria.

Will be unacceptable :

\* duplication of existing code

\* illogic java class

\* non object-oriented programming