WARNING : Any fraud or plagiarism of the UML class diagram or Python code (partial or total reproduction of the diagram / code of another group or of the Internet) will be sanctioned by **a convocation to the disciplinary board**.
Changing variables' names or adding comments to code taken from elsewhere is still plagiarism.

RATING CRITERIA

This project aims at making you practice all the object's related concepts seen in this course and to apply them on a subject vaster than a simple Practical session. It is to be realized by groups of 3 in autonomy. No time slots are planned for the realization of this course but your teachers are available to follow your work (it is up to you to ask them !).
Even if some concepts may be missing at the beginning of the project (the subject being revealed to you before the end of the lectures), it is nevertheless possible to work on a "classic" solution (*i.e.* non-object but with functions / libraries) which will be enriched thereafter with these new concepts (*e.g.* libraries could become objects, functions will become methods of classes, *etc.*). This will allow you to work regularly on the project and will give you time to complete it.

The deliverables for this project are :

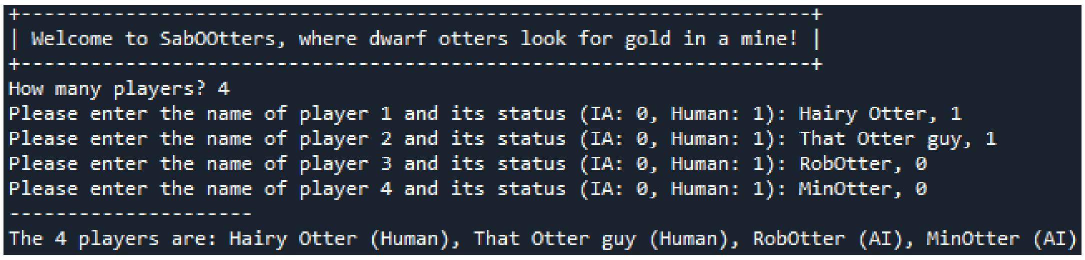
1. **UML class diagram :**
   (a) Your UML diagram should model the project while using all the object-oriented principles (links between classes, encapsulation, abstraction, polymorphism).
   (b) The project allows the use of these concepts and your diagram should therefore contain
      — at least one aggregation or composition
      — at least one inheritance link
      — at least one abstract class
   (c) The choices of classes' links (other than inheritance) and encapsulation must be justified (in a few lines).
   (d) The UML class diagram must be readable, and it is therefore recommended to draw it
      — with the help of a software (Dia, ArgoUML, . . .)
      — via a website (online.visual-paradigm.com, creately.com/lp/uml-diagram-tool)
2. **Python implementation :**
   (a) Your Python implementation must correspond rigorously to your UML class diagram. It must therefore contain the same classes, links and fields as those indicated in your diagram.
   (b) Your code should
      — compile and run without error
      — be commented so that it is easily read and understood
      — satisfy the "Zen of Python" (www.python.org/dev/peps/pep-0020).
      — satisfy the object-oriented principles (use of constructors, methods for converting objects into strings, comparison of objects, . . .)
      — be composed of several files/modules that will be imported by the main program
   (c) Your program should work
      — from clear displays in the console (textual mode, see images below)
      — in Player vs Player mode (*i.e.* only human users)
      — using values given by the user (to choose the action to execute, the next step, . . .)
   (d) The quality of your code and its proper use of Python functionnality will also be rated and in particular
      — the use of Python's functions in `for`-loop with lists (*e.g.* `enumerate`, `zip`) instead of counters and `range`
      — the use of list comprehensions instead of `for`-loops with multiple `append`
3. **Options :** in addition to the basic implementation described in Q2, your program should implement at least one of the proposed options.

The project groups are to be defined **within your TP group** before **October 30th**. The project (UML diagram and its justifications, Python code) is due by **Januеary 6th, 23h59**.

Project - SabOOtters

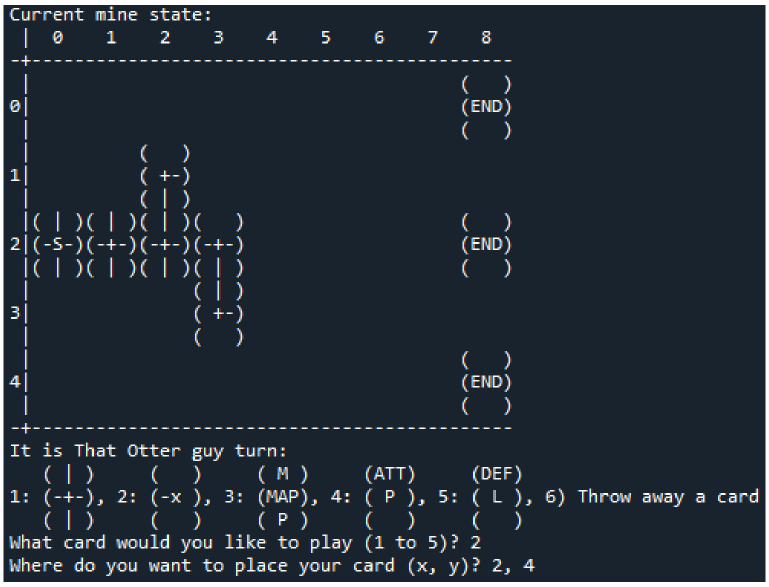
Saboteurs™ is a board game where dwar~~ves~~f otters dig in a mine looking for a golden nugget. However some of the dwarves have other plans and try to sabotage the mining operations.
The detailed rules (in French) can be found on this website :
— https ://www.gigamic.com/files/catalog/products/rules/rules-saboteur-2.pdf (pages 1 to 4)

The proposed options for this project are :
— a Graphical User Interface (but a textual version using the log consol must still be implemented)
— an AI to allow playing in a Human *vs.* Computer way. Several levels of difficulty can be proposed.
An AI will simply consist of a series of instructions allowing to simulate a human player (no need to recode AlphaGo or Watson). For example, the computer will first try to locate the gold (with the help of a map), then to advance (if a miner) or to attack (if a traitor).
— the implementation of the game's extension (pages 5+ of the rule book.)



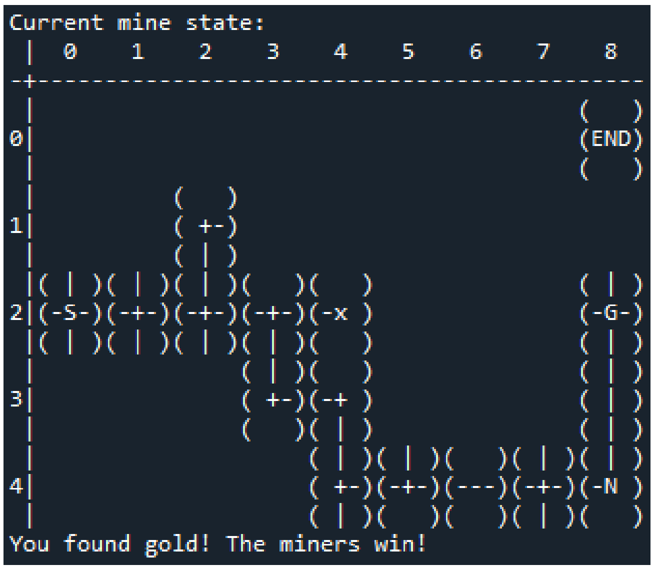
```
+-----------------------------------------------------------------+
| Welcome to SabOOtters, where dwarf otters look for gold in a mine! |
+-----------------------------------------------------------------+
How many players? 4
Please enter the name of player 1 and its status (IA: 0, Human: 1): Hairy Otter, 1
Please enter the name of player 2 and its status (IA: 0, Human: 1): That Otter guy, 1
Please enter the name of player 3 and its status (IA: 0, Human: 1): RobOtter, 0
Please enter the name of player 4 and its status (IA: 0, Human: 1): MinOtter, 0
---------------------
The 4 players are: Hairy Otter (Human), That Otter guy (Human), RobOtter (AI), MinOtter (AI)
```

(a) Welcome screen



```
Current mine state:
 |  0    1    2    3    4    5    6    7    8
-+---------------------------------------------
 |                                        (   )
0|                                        (END)
 |                                        (   )
 |          (   )
1|          ( +-)
 |          ( | )
 |( | )( | )( | )(   )                    (   )
2|(-S-)(-+-)(-+-)(-+-)                    (END)
 |( | )( | )( | )( | )                    (   )
 |               ( | )
3|               ( +-)
 |               (   )
 |                                        (   )
4|                                        (END)
 |                                        (   )
-+---------------------------------------------
It is That Otter guy turn:
   ( | )     (   )     ( M )     (ATT)     (DEF)
1: (-+-), 2: (-x ), 3: (MAP), 4: ( P ), 5: ( L ), 6) Throw away a card
   ( | )     (   )     ( P )     (   )     (   )
What card would you like to play (1 to 5)? 2
Where do you want to place your card (x, y)? 2, 4
```

(b) Player's turn



```
Current mine state:
 |  0    1    2    3    4    5    6    7    8
-+---------------------------------------------
 |                                        (   )
0|                                        (END)
 |                                        (   )
 |          (   )
1|          ( +-)
 |          ( | )
 |( | )( | )( | )(   )(   )               ( | )
2|(-S-)(-+-)(-+-)(-+-)(-x )               (-G-)
 |( | )( | )( | )( | )(   )               ( | )
 |               ( | )(   )               ( | )
3|               ( +-)(-+ )               ( | )
 |               (   )( | )               ( | )
 |                    ( | )( | )(   )( | )( | )
4|                    ( +-)(-+-)(---)(-+-)(-N )
 |                    ( | )(   )(   )( | )(   )
You found gold! The miners win!
```

(c) Game end