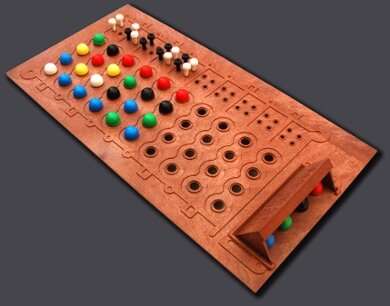
# Mastermind

Mastermind is a code-breaking game (<https://en.wikipedia.org/wiki/Mastermind_(board_game)>). The goal of the game is to discover a hidden code that consists of 4 colours.

The board game is played using:

* **a decoding board**, with a shield at one end covering a row of four large holes, and a set of rows containing four large holes next to a set of four small holes.
* **code marbles** of six different colours (Red, Green, Blue, Yellow, White, Black) are used to represent a code and placed in the large holes on the board; and
* **key pegs**, some coloured black, some white, are for providing feedback on the guessed code placed in the small holes on the board.



Played between two players in a board game, one player becomes the *codemaker*, the other the *codebreaker*.

The codemaker chooses a code, a pattern of four code marbles. Duplicates are allowed depending on player choice, so the player could even choose four code marbles of the same colour. The chosen code is placed in the four large holes covered by the shield, visible to the codemaker but not to the codebreaker.

The codebreaker tries to guess the code, in both order and colour, within a certain number of attempts (e.g., 10 in the picture above). Each guess is made by placing a row of code marbles on the decoding board. Once placed, the codemaker provides feedback by placing from zero to four key pegs in the small holes of the row next to the guess. A black key peg is placed for each code marble from the guess which is correct in both colour and position. A white key peg indicates the existence of a correct colour code marble placed in the wrong position.

If there are duplicate colours in the guess, they cannot all be awarded a key peg unless they correspond to the same number of duplicate colours in the hidden code. For example, if the hidden code is white-white-black-black and the player guesses white-white-white-black, the codemaker will award two black key pegs for the two correct whites, nothing for the third white as there is not a third white in the code, and a black key peg for the black. No indication is given of the fact that the code also includes a second black.

Once feedback is provided, another guess is made by the codebreaker; guesses and feedback continue to alternate until either the codebreaker guesses correctly, or the number of incorrect guesses reach the number of attempts allowed. If the code is not guessed correctly within the number of attempts allowed, then the shield is uncovered to reveal the code.

The following video clip may be also helpful to understand the concept: <https://www.youtube.com/watch?v=wsYPsrzCKiA>

# The Assignment Task Specification

As a text-based application software, WoM will use a command line interface where users type in a command or input and the relevant output is printed onto the screen in text. When the software is executed, it will show a greeting message and a list of commands, followed by a command prompt ‘>’ indicating the system is ready to take user’s input. At the prompt, users can give one of the following commands:

* r: register a new user, which will create a new user
* s: show the score board, which will print the list of the users and their scores
* p: play a game, which will start a new game
* q: quit, which will quit the application

For registering a new user, the system will ask for the name of the new user. The username is not allowed to be changed after being registered, and also the name must be unique (i.e. not allowed to have the same name as an existing user). A new user will always start with a score of 0 point.

Below is an example of the output on the screen (user’s input in ***bold italic***).

|  |
| --- |
| Welcome to the World of Mastermind!  Developed by Alan Turing  COMP 1048 UO Object-Oriented Programming  What would you like to do?  (r) register a new user  (s) show the score board  (p) play a game  (q) quit  > ***r***  What is the name of the new user?  > ***Alan***  Welcome, Alan!  What would you like to do?  (r) register a new user  (s) show the score board  (p) play a game  (q) quit  > ***r***  What is the name of the new user?  > ***Alan***  Sorry, the name is already taken.  What would you like to do?  (r) register a new user  (s) show the score board  (p) play a game  (q) quit  > ***r***  What is the name of the new user?  > ***Steve***  Welcome, Steve! |

Once the users are registered, they can play the game of Mastermind. Only one game will be played at a time, i.e., the game must be finished before playing another game. Before starting the game, users will be asked questions for setting it up. First, users will be asked how many players (between 2 to 4, inclusive) will play the game, followed by asking for the name of each player. If the name does not match to a valid registered user or if the user with the name is already in the game, an error message will be shown and asked again. There will be a special built-in name “HAL9000” and “VIKI” that are for players controlled by the computer. Users are not allowed to register with these names assigned to computer players. After the players are selected, the users will be asked to choose how many attempts (between 5 to 10, inclusive) of guessing the code will be allowed.

|  |
| --- |
| What would you like to do?  (r) register a new user  (s) show the score board  (p) play a game  (q) quit  > ***p***  Let’s play the game of Mastermind!  How many players (2-4)?  > ***2***  What is the name of player #1?  > ***Bill***  Invalid user name.  What is the name of player #1?  > ***Alan***  What is the name of player #2?  > ***Alan***  Alan is already in the game.  What is the name of player #2?  > ***HAL9000***  How many attempts will be allowed for each player (5-10)?  > ***5*** |

After it’s all set, the game will start. The game is played in a round-robin manner where each player takes a turn. First, each player will take a turn to make a code for the next player who has to break it. Think of it as if each player has a decoding board, and each player sets the code hidden behind the shield for the next player if it were in an actual board game. For example, if there are three players, first player sets the code for the second player to break, the second player sets the code for the third player, and the third player sets the code for the first player.

To set the code, users will enter a string of four characters, each character representing a colour that consists the code. The characters representing colours are, ‘R’ for red, ‘G’ for green, ‘B’ for blue, ‘Y’ for yellow, ‘W’ for White, and ‘K’ for black. For example, ‘RGBK’ will be a code of red, green, blue, and black. If the code entered is invalid (i.e. either not having four colours or including any invalid colour) an error message is printed and asked for entering the code again. If the player in turn is a computer, it will generate a random code.

|  |
| --- |
| \* Alan's turn to set the code for HAL9000 to break.  Please enter the code:  > ***BTWRA***  Invalid code.  It must be exactly four characters, each can be R, G, B, Y, W, or K.  Please enter the code:  > ***BYWR***  The code is now set for HAL9000 to break.  \* HAL9000's turn to set the code for Alan to break.  The code is now set for Alan to break. |

Once all of the players took turn to set the code for the next player, players take turns to guess the code set for him/her/itself. In each turn, a summary of previous attempts of the player taking the turn is shown on the screen. Users are then asked to enter their guess, while the computer players will randomly generate its guess and print it on the screen. The feedback on a guess is automatically provided by the system. If a player breaks the code (i.e., gives the correct code) the game continues with the rest of the players until everyone either succeeds to break the code or fails by using up all the number of attempts allowed.

|  |
| --- |
| \* Alan's turn to guess the code.  Previous attempts: 0  Attempts left: 5  Please enter the code:  > ***BGY***  Invalid code.  It must be exactly four characters, each can be R, G, B, Y, W, or K.  Please enter the code:  > ***BGYY***  Feedback: K  \* HAL9000's turn to guess the code.  Previous attempts: 0  Attempts left: 5  HAL9000’s guess: BRWG  Feedback: K K W  \* Alan's turn to guess the code.  Previous attempts: 1  ==============  Code Feedback  ==============  BGYY K  ==============  Attempts left: 4  Please enter the code:  > ***BKRW***  Alan broke the code in 2 attempts!  \* HAL9000's turn to guess the code.  Previous attempts: 1  ==============  Code Feedback  ==============  BRWG K K W  ==============  Attempts left: 4  HAL9000’s guess: GWKG  Feedback: W  \* HAL9000's turn to guess the code.  Previous attempts: 2  ==============  Code Feedback  ==============  BRWG K K W  GWKG W  ==============  Attempts left: 3  HAL9000’s guess: WGWK  Feedback: K  \* HAL9000's turn to guess the code.  Previous attempts: 3  ==============  Code Feedback  ==============  BRWG K K W  GWKG W  WGWK K  ==============  Attempts left: 2  HAL9000’s guess: BYWG  Feedback: K K K  \* HAL9000's turn to guess the code.  Previous attempts: 4  ==============  Code Feedback  ==============  BRWG K K W  GWKG W  WGWK K  BYWG K K K  ==============  Attempts left: 1  HAL9000’s guess: RYWG  Feedback: K K W  HAL9000 failed to break the code.  The game is now finished.  Alan receives 4 + 5 = 8 points.  What would you like to do?  (r) register a new user  (s) show the score board  (p) play a game  (q) quit  > **\_** |

Once everyone either breaks the code or uses up the number of attempts allowed, the game finishes with showing the results with the points each user receives. Note that the computer players don’t receive any points. Each user receives points according to the number of attempts left + 1 after breaking the code. In addition, users also get points for one less than the number of attempts made by the next player for whom they set the code for. For example, in a game with 10 attempts allowed, if player #1 broke the code after 3 attempts and player #2 broke the code after 9 attempts, player #1 will receive 10 – 3 + 1 = 8 points for breaking the code, and also 9 – 1 = 8 points for making the code for player #2. If both player #1 and #2 fail to break the code after 10 attempts, player #1 will get 0 (breaking) + 10 (making) = 10 points. After the game results are printed, it goes back to the main menu.

Users can check the score board to keep track of the scores after playing games. The score board will include the list of all registered users with their details, including their name, the current score (i.e. points), the number of games played, and the average points gained per game (i.e. the current score divided by the number of games played, rounded up to one digit under decimal point). Note that a computer player is not a user, hence it will not show up on the score board.

|  |
| --- |
| What would you like to do?  (r) register a new user  (s) show the score board  (p) play a game  (q) quit  > ***s***  =====================================  Name Score Games Average  =====================================  Alan 25 3 8.3  Steve 15 2 7.5  Elon 0 0 0.0  =====================================  What would you like to do?  (r) register a new user  (s) show the score board  (p) play a game  (q) quit  > ***q***  Thank you for playing the World of Mastermind! |

# Additional Requirements

**Requirement 1.**

Your implementation must be in Python 3.9

One exception is to show the computer-generated hidden code on the screen so it will be useful for the testing purpose (shown in bold character):

|  |
| --- |
| \* HAL9000's turn to set the code for Alan to break.  **DEBUG: RGBY**  The code is now set for Alan to break. |

**Requirement 2.** Apply proper Object-Oriented design principles to implementation.

Your implementation must practice Object-Oriented design principles as following:

1. Abstraction

Your implementation must include definition of at least 5 classes including the WorldOfMastermind class. All of your code must belong to one of the classes you define, except the import statements and the following code which will run the software:

wom = WorldOfMastermind()

wom.run()

This run() method in the WorldOfMastermind class must be the starting point of your software and call methods of other objects in the system.

1. Data Hiding and Encapsulation

All the data attributes defined in your classes must be **private**. In addition, following conditions must be met on attributes:

* Player’s (or user’s) name must not be changeable once initialised (i.e., must be read only).
* User’s score must be only allowed for being increased by an asked amount but never to be set with a value. (i.e., no setter method, but with a method to increase the score)
* The hidden code on a decoding board must be only settable (i.e., once set, its value never can be accessed from outside the class that it belongs to but can ask for feedback providing a guessed code.)

1. Inheritance and Polymorphism

There should be at least one inheritance relationship used and at least one instance of practicing polymorphism (i.e., overriding a method in the superclass). Overriding an initialiser or string conversion method is not counted in this requirement (i.e., you must override a method you defined in the superclass). A subclass must not repeat the code already in its superclass.

**Requirement 3.** Documentation with docstrings.

Your code must be documented appropriately using docstrings.

Each class and each method should have at least one docstring which can be brief. The responsibilities that you have specified in the UML diagram can be used as a starting point for the class docstring.

Document as you go, not all at the end. One strategy is to write a brief comment about what the   
"1 thing" the method is supposed to do when you declare it, then refer to that comment while implementing it: this helps maintain focus when implementing the method and helps stop yourself from making it do more than the '1' thing it is supposed to do.

**Requirement 4.** Use consistent coding style.

Your Python code must adhere to consistent coding style. Your class names must be in CapitalCamelCase style, while the methods and attributes could be in either smallCamelCase or under\_score style. While you may use one or the other it should be consistent throughout your code. If you are note sure, follow the Python style guide available here:   
<https://www.python.org/dev/peps/pep-0008/>

**Requirement 5.** Use exception handling.

Your implementation must use exception handling for dealing with incorrect user inputs or any other type of exceptions. For example, your program should not halt (a.k.a. crash) if the user gives non-numerical input where a number is expected, but rather handle such exceptions using try-except statements.

**Requirement 6.** Conduct testing.

Your submission must include Python module(s) for testing. It is recommended that you test individual methods as you go, rather than trying to test the whole application through the menu-driven interface. Writing code to test specific elements will speed up development as you will not need to constantly enter data through the menu interface.

Your submission should have code for testing the methods of each class you define.