University of South Australia

UniSA STEM

COMP 1046
Object-Oriented Programming

Assignment
Mastermind Game Development
Part 1- Design

The University of South Australia

SP2 2021

Introduction

The assignment is intended to provide you with the opportunity to put into practice what you have learnt in the course by applying your object-oriented programming knowledge and skills. It will require you to apply object-oriented methods to design, implement, and test a text-based application software where users can play a Mastermind game.

The assignment is worth for 30% of the assessment in this course, and consists of two parts:

- Part 1 – Design (10%, due week 5)
- Part 2 – Implementation (20%, due week 13)

This document specifies the tasks for Part 1 of the assignment, UML design of the software, while Part 2 is be specified in a separate document.

This assignment is an **individual task** that will require an **individual submission**. You are required to **submit your work via LearnOnline as prescribed in the Submission Details section.**

This document is a kind of specification of the required end product that will be generated by implementing the assignment. Like many specifications, it is written in English and hence will contain some imperfectly specified parts. **Please make sure you seek clarification** if you are not clear on any aspect of this assignment.

Course Objectives

By completing this assignment, you are expected to partially fulfill the following course objects:

- CO1. Convert a problem statement into an object oriented solution
- CO2. Use inheritance and polymorphism to solve problems
- CO6. Analyse and explain the behaviour of object oriented programs

Assignment Learning Outcomes

This assignment is for assessing the following learning outcomes:

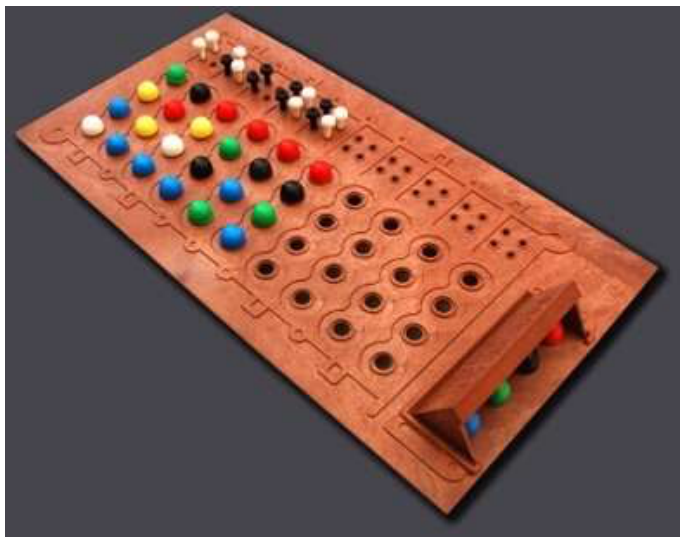
- Design a UML class diagram using UMLet
- Apply object-oriented principles including encapsulation, abstraction, inheritance, and polymorphism to the software design

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

The task for the Part 1 of the assignment is to produce a UML class diagram of an application software, called the World of Mastermind (WoM), a text-based application software where users can play a Mastermind game.

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 user name 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 1046 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. 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 3 + 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 after breaking the code. In addition, users also get points for the number of attempts made by the next player for whom s/he 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 = 7 points for breaking the code, and also 9 points for making the code for player #2. 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

The aim of the assignment is to allow you to practise creating an object-oriented design and implementing it in Python using object-oriented programming concepts such as *abstraction, encapsulation, class inheritance*, *polymorphism*.

In this first part of the assignment, you will need to ***develop a class design using UML class diagrams*** in UMLet based on the requirements above and in the remainder of this specification. The focus of the assignment is on identifying classes and applying appropriate relationships between the classes. **The UML class diagram MUST include all necessary classes, attributes and methods, and all relationships between classes. Also, visibility, data types, and cardinalities must be specified in detail.**

When designing the UML class diagram, you must keep in mind that the design is going to be implemented in the second part of the assignment. The UML design should be designed in such a way that code is reused and can be extended easily. Your design must presume the following Python code will run the application software you developed:

```
wom = WorldOfMastermind()

wom.run()
```

Think of how this `run()` method will be the starting point and call methods of other objects in the system to identify classes, their methods and attributes, and their relationships based on reading the assignment task specification.

While more detailed instructions on implementation will be provided with the specification for Part 2 of this Assignment, the description in this document should provide enough details for designing the UML class diagram. However, if you have any doubt or question while developing the UML class diagram, please contact the Course Coordinator for clarification through either online forum on the course website or through email.

Work Plan

To complete the first part of this assignment, you will need to perform the following steps:

1. Read the assignment specification carefully.
2. Review the relationships between classes in a UML class diagram: inheritance, association, aggregation, and composition.
3. Identify which classes are necessary to capture the requirements.
4. Identify attributes and methods of those classes. Specify visibility for them as well as data types for attributes, parameters and return values of methods.
5. Identify relationships between classes and add cardinalities and labels to the relationship where applicable.
6. Revisit all classes and see if you can use inheritance to minimize duplication of code.
7. If you have an abstract class, then set its name in italic.
8. You may add annotations to classes if it helps you later with the implementation (e.g., responsibilities of classes).

Identifying all methods and attributes does not have to be complete or 100% correct at this point in time. However, enough details should be captured in the UML class diagram as it should give you the opportunity to think about the structure and flow of the application before starting the implementation. You can revise it later in Part 2 of the assignment as you progress through the implementation, for example, you may realise you are missing some arguments to a member function or missing an entire function. The marking of this aspect will focus on the appropriate classes and relationships, separation of concerns, and good naming that captures the requirements and concepts described in this assignment specification document.

After completing the design, export it as an image file and copy it into a Word document or generate a PDF version and submit it to LearnOnline.

Submission Details

You MUST submit your UML Class Diagram in a **Word or PDF** file to LearnOnline. The UML class diagram MUST be generated from the UMLet or UMLetino modelling tool: a photo of a hand-drawn diagram will NOT be accepted. The submission must include your student details (Full name and Student ID).

The submission is **due by the end of Week 5**. Exact date and time for the deadline is as specified on the submission link on LearnOnline.

This assignment is an **individual task** that will require an **individual submission.**

There will be **no extensions for this course without one of the following exceptions:**

1. A medical certificate is provided that has the timing and duration of the illness and an opinion on how much the student's ability to perform has been compromised by the illness. Please note if this information is not provided the medical certificate WILL NOT BE ACCEPTED. Late assessment items will not be accepted unless a medical certificate is presented to the Course Coordinator. The certificate must be produced as soon as possible and must cover the dates during which the assessment was to be attempted. In the case where you have a valid medical certificate, the due date will be extended by the number of days stated on the certificate up to five working days.
2. A Learning and Teaching Unit councillor contacts the Course Coordinator on your behalf requesting an extension. Normally you would use this if you have events outside your control adversely affecting your course work.
3. Unexpected work commitments. In this case, you will need to attach a letter from your work supervisor with your application stating the impact on your ability to complete your assessment.
4. Military obligations with proof.

Applications for **extensions must be lodged with the Course Coordinator before the due date** of the assignment, or as soon as possible after the incident took place.

Note: Equipment failure, loss of data, 'Heavy work commitments' or late starting of the course are not sufficient grounds for an extension. Make sure you make a back-up of your **.uxf** file frequently, especially when working on a web-based UMLetino.

Late submissions without an approved extension will incur a penalty of -30% of your mark.
No more late submission will be accepted after a week from the due date.

Marking Criteria

Your UML Class Diagram will be marked against an expected design. The focus is on the correct identification of classes and their relationships, and capturing enough details from this assignment specification. Penalties will be applied if the design is not submitted in the correct format. After the submission deadline, the rubric used to allocate marks will be released to provide immediate feedback on what was expected.

Criteria	Mark
Correct identification of classes	20
Attributes with datatypes and visibility	10
Methods with parameters, return data types, and visibility.	10
Correct use of inheritance.	20
Correct use of aggregation and/or composition with cardinalities.	20
Correct use of association with labels and cardinalities.	20
Submission not in appropriate format (Only UMLet/UMLettino drawn diagram in a Word or PDF is acceptable.)	-15
Total Possible Marks	**100**

Academic Misconduct

This assignment is an individual task that will require an individual submission.

Students are reminded that they should be aware of the academic misconduct guidelines available from the University of South Australia website.

Deliberate academic misconduct, such as plagiarism, is subject to penalties. Information about Academic integrity can be found in Section 9 of the Assessment Policies and Procedures Manual at:

https://i.unisa.edu.au/policies-and-procedures/codes/assessment-policies/