Project Plan for Master Mind :: Escape the Room

Jiaying Wu

How to play Master Mind :: Escape the Room

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                        MasterMind :: Escape the Room
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
"You have been locked in an empty dark room, running out of food and water.
In front of you, there is a gate with a giant lock. The times you can enter
the numbers into the lock is limited! Once you ran out of the times, the
gate will lock forever! Hurry up! Clock is ticking......"

After you enter the elements into the lock, the lock will provide you some clues.

- One `#` will show up if you enter a correct element in correct position.

- One `*` will show up if you enter a correct element in wrong position.

- No symbol for a worng element

- The order of symbols to show up is random in each time.

If you enter all correct elements with correct position into the lock in the
limited times, the gate will open. That you can escape the room. Otherwise,
you will be lock in the room forever.

Three steps to start the game:

1. Enter your name and press Enter.

2. Select the gate you want to open:

 [1] Wood gate = easy: 4 elements out of 6 with 10 times to try.

 [2] Rock gate = tricky: 5 elements out of 8 with 12 times to try.

 [3] Iron gate = hard: 6 elements out of 10 with 14 times to try.

3. Select the type of element for the lock:

 [1] Number

 [2] Letter
```

```
[3] Symbol

[4] Word

You will then face to the gate you have chosen.
```

The text above is the game rule of Master Mind :: Escape the Room. It seperate into two part.

The first part is the first 3 lines start with "~"" and end with "~" as the title. It display by a function `displayTitle()`.

The second part is the body after the "~", store in a text file `masterMindRules.txt`.

Development outline of Master Mind :: Escape the Room

Outline the functionality of all game classes

After reading the assignment brief, I decided to use 3 classes for this game.

1. **Player class**

Player class is

2. **Board class**

Board class is

3. **Application file**

Application file include

The game setup

Four step for game setup:

1. Display the title of the game.
2. Load and display the `masterMindRules.txt` and include a "Press any key to continue" to pause the screen, allow player to read the game rules.
3. After player finishes reading the game rule, that is press a key to continue. Initials the game variables:

- Ask the name of player, store as a *string* `playerNmae`.
- Ask the player to select the gate to open, that is the game difficulty level. Store as a *int* `difficultyLevel`, possible value is {1, 2, 3}.
- Create a *int* `possibleElement` = 3 + 2* `difficultyLevel` to indicate the number of all possible elements, possible value is {6, 8, 10}.
- Create a *int* `codeElement` = 4 + `difficultyLevel` to indicate the number of elements in one row, that is the numbers of element in the secret code. Possible value is {4, 5, 6}.

- Create a *intr* `codeRow` = 8 + 2 * `difficultyLevel` to indicate the number of rows, that is the times players can try. Possible value is {10, 12, 14}.
- Ask the player what type of element they want to choose, store as a *int* `elementType`, possible value is {1, 2, 3, 4}

4. After player finish enter their name and game option, include a "Press any key to continue" in the end. To allow user final review the information before the screen be cleard.

What we need for game setup:

Functions:

1. `displayTitle()` to display the title of the game.
2. `displayRule()` to display `masterMindRules.txt`.
3. *int* `askForNumber(string question)` to display the question and ask uesr's choice, *string* **question** is the local variable within the function `askForNumber()`.

Variables:

1. *string* `playerNmae`
2. *int* `difficultyLevel`
3. *int* `possibleElement`
4. *int* `codeElement`
5. *int* `codeRow`
6. *int* `elementType`

The player's turn

1. After the press a key to continue, clear the screen.
2. display the title of the game.
3.

Processing player input

Providing feedback to player

The end game conditions

Additional features included

UML class diagrams