219114/115 Programming I

Exercise 8: Objected-Oriented Programming (OOP) II

In this exercise, you will program two games, Mastermind and Blackjack, in OO style.

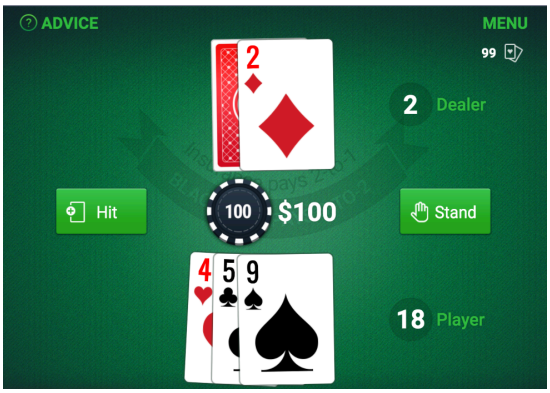
Mastermind

Create two python files, mastermind.py and play_mastermind.py. The first file accommodates class definitions and the second simulates the Mastermind game playing. See sample runs of this program and guidelines on how to design and implement it in OO style in the lecture materials for week 11 (OOP Design).

In mastermind.py, include docstrings and doctests for classes and methods. The attached file, OO_fraction.py, that is posted along with this exercise instructions shows how to do doctest in methods in a class module.

Blackjack

Create three python files, card_deck.py, blackjack.py, and play_blackjack.py. The first two files contains class definitions for objects that is a deck of cards and a Blackjack game itself. The last file simulates a two-players Blackjack game. Do not forget to include docstrings and doctests in files that contains class and methods definitions. See sample runs of this program and guidelines on how to design and implement it in OO style in the lecture materials for week 12 (OOP Design II).



Multi-player Blackjack

If you are to program a multi-player Blackjack where one dealer handles more than a single player, what modification to the two-player Blackjack is needed to accommodate multiple players? Do you need additional classes? Outline your design in a file named multiplayer_blackjack_design.pdf. You can use class diagrams to make your design more visual and understandable.

Submission:

- ***Create StudentID_Firstname_ex8 folder, where StudentID is your KU ID and Firstname is your given name***
- ***Put the files to submit, mastermind.py, play_mastermind.py, card_deck.py, blackjack.py, play_blackjack.py, and multiplayer_blackjack_design.pdf into this folder***
- ***Zip the folder and submit the zip file to the course's Google Classroom before the due date***

Grading:

1. Correctness (50%); your code must run and produce correct outcomes; code that does not run because of, for example, syntax errors or name misspelling receives zero credit.

2. Good OOP style and documentation (40%); your code must define classes and create appropriate objects that interact to produce the desire result. Class design must be fitting for the given problem. Docstrings and doctests must be included.

3. Cleanliness (10%): your code must be clean, following PEP 8 style guide; variable names must be meaningful, following PEP 8 convention; comments must be put in for others to be able to read and understand your code, again following PEP 8 convention.