Bartosz Gembalczyk 320683
Podstawy Informatyki i Programowania – Projekt

1. Cel i opis projektu
   Celem projektu było stworzenie gry Mastermind.
   Zasady gry: https://en.wikipedia.org/wiki/Mastermind_(board_game)
   Grę w pełnej funkcjonalności da się zaimplementować w terminalu, więc ze względu na możliwość wyboru nie robiłem GUI.
2. Podział na pliki
   a. Pliki źródłowe
      i. main.py
      Plik main.py zawiera funkcje main odpowiadającą za uruchomienie i trwanie gry.
      ii. mastermind.py
      Plik mastermind.py opisuje klasę Mastermind, która odpowiada za obsługę całej rozgrywki.
      Atrybuty klasy Mastermind dotyczą danej pojedynczej rozgrywki reprezentowanej klasą Game; ilość rozegranych rozgrywek; listę zawierającą graczy reprezentowanych klasą Player lub jej subklasami; dziennik zawierający zasady gry.
      Metody klasy Mastermind to konstruktor; gettery; metoda odpowiadająca za wprowadzanie zasad gry; za rozpoczęcie gry; za obsługę pojedynczej rundy gry Mastermind; za wypisywanie wyników każdego z graczy.
      iii. game.py
      Plik game.py opisuje klasę Game, która odpowiada za obsługę pojedynczej gry w Mastermind
      Atrybuty klasy Game dotyczą odgadywanego klucza; gracza szyfrującego oraz deszyfrującego reprezentowanych klasą Player oraz dziennik z zasadami.
      Metody klasy Game to konstruktor; gettery; metoda wprowadzająca klucz tworzony przez szyfrującego gracza do gry; rozpoczynająca pojedynczą rundę gry Mastermind; obsługę pojedynczej tury w rundzie; sprawdzająca czy kod został złamany; metoda wywoływana w momencie złamania kodu i zwracającą referencje na gracza deszyfrującego; analogiczna w przypadku niepowodzenia; kończąca pojedynczą rozgrywkę.
      iv. player.py
      Plik player.py opisuje klasę Player, która odpowiada za obsługę każdego gracza.
      Atrybuty klasy Player dotyczą imię gracza; zdobytych punktów; listę zawierającą poprzednie próby odgadnięcia oraz odpowiedzi do nich; dziennik z zasadami.
      Metody klasy Player to konstruktor; getter; rzutowanie na klasę String; metoda dodająca punkt za wygraną danemu graczowi; metoda sprawdzająca poprawność danej kombinacji zgodnie z zasadami gry.
      Trzy metody klasy Player w jej subklasie są przeciążane, więc mają

zastosowanie tylko dla graczy fizycznych, te metody to metoda pobierająca klucz z terminala do odgadnięcia; pozwalająca wpisanie deszyfrującemu graczowi próbę odgadnięcia szyfru; szyfrującemu graczowi odpowiedź na próbę złamania kodu.

v. AIPlayer.py
Plik AIPlayer.py opisuje subklasę klasy Player – klasę AIPlayer oraz jej dwie subklasy – klasę AIPlayerEasy oraz klasę AIPlayerMedium, które odpowiadają za obsługę gracza komputerowego.
    1. AIPlayer
    Klasa AIPlayer przyjmuje z góry określone imię gracza oraz nowy atrybut informującą o trudności rozgrywki z graczem komputerowym.
    Dodatkowe metody klasy AIPlayer to dwie przeciążone metody nadklasy – tworzącą losowy szyfr oraz odpowiadającą na wprowadzoną przez fizycznego gracza próbę złamania kodu.
    2. AIPlayerEasy
    Dodatkowa metoda klasy AIPlayerEasy to przeciążona metoda nadklasy i zawsze zgadująca losowo.
    3. AIPlayerMedium
    Dodatkowowy atrybut to lista wszystkich dalej dostępnych prawdopodobnych szyfrów.
    Dwie nowe metody klasy AIPlayerMedium oraz przeciążona metoda nadklasy odpowiadają za możliwie jak najlepszą próbę odgadnięcia kodu przez gracza komputerowego na podstawie algorytmu.
    Algorytm wzorowany jest na algorytmie Donalda Knutha, lecz jednak jest jego uproszczoną wersją.
    Omówienie algorytmu Knutha:
    https://en.wikipedia.org/wiki/Mastermind_(board_game)#Worst_case:_Five-guess_algorithm
    Algorytm zastosowany przeze mnie jest trochę słabszy, zwykle odgaduje kod, po jakiś sześciu-ośmiu próbach.
    Na początku tworzona jest lista zawierająca wszystkie możliwe szyfry zgodne z wprowadzonymi zasadami
    Pierwsze zapytanie następuję w pełni losowo.
    Jeżeli zapytanie poprzednie było kompletnie nietrafione tj. żaden kolor się nie pokrywa, na jego podstawie wybierana jest potencjalna próba złamania szyfru. Jeśli ten potencjalny strzał dalej jest liście dostępnych kodów, staję się finalną próba odgadnięcia. Jeśli nie to jego następny dostępny w tej liście kod, jeśli nie ma kolejnego to pierwszy na liście.
    Jeśli poprzednie zapytanie było jakkolwiek trafione to lista dostępnych kodów jest filtrowana i zostają tylko dalej zgodne z poprzednim zapytaniem a kolejnym zapytaniem staje się ten pierwszy na tej liście.

vi. output.py
Plik output.py opisuje trzy funkcje służące na wyświetlanie kodu w

terminalu, żeby w możliwie zbliżony sposób przypominała planszę do gry w Mastermind.
Funkcja display wyświetla plansze po zakończonej rundzie w nowym oknie terminala, korzystając z modułu curses. W finalnej wersji projektu tak powinna wyglądać rozgrywka jednak wprowadzanie kodów w nowym oknie było trochę zbyt skomplikowane.

vii. auxiliaries.py
Dodatkowe funkcje pomocnicze wykorzystywane w więcej niż jednym pliku.

viii. exceptions.py
Customowe wyjątki

b. Pliki testowe
   i. test_mastermind.py
   ii. test_player.py
   iii. test_AIPlayer.py
   iv. test_AIPlayerEasy.py
   v. test_AIPlayerMedium.py
   vi. test_auxiliaries.py

3. Instrukcja użytkownika
Przed rozgrywką należy wprowadzić zasady gry za pomocą prostych i intuicyjnych inputów w terminalu.
Zasady gry, które trzeba wprowadzić: ilość graczy, dostępne jest tryb jedno- lub dwuosobowy; liczbę tur w każdej grze, (od 1 do 15); długość klucz, (od 2 do 8); liczbę kolorów (od 2 do 8); jeżeli gra odbywa się w wersji jednoosobowej to dodatkowo wprowadza się poziom trudności gracza komputerowego, dostępne są dwa tryby trudności łatwy i średni.
4. Rozwój i plany na przyszłość
To czego nie udało mi się zrobić, to wprowadzania szyfru oraz możliwych prób odgadnięcia w nowym oknie terminala, skupiłem się bardziej na części algorytmicznej projektu, a dopiero w tej bardziej złożonej wersje graficznie bardziej by to przypominało planszę do gry w Mastermind.
Dodatkowo należałoby dopisać i uzupełnić części testów jednostkowych.
To co mogłoby zaskakiwać to nazwanie tylko dwóch poziomów trudności łatwym i średnim, wynikało to z początkowego planu na stworzenie trzeciego poziomu trudności z kompletną implementacja algorytmu Donalda Knutha.
Kolejną możliwością rozwoju projektu byłoby umieszczenie wszystkich tekstów wypisywanych na ekranie w pliku tekstowym i odpowiednio sczytywanie z pliku, co ułatwiłoby implementacje innych języków.