Opis i wymagania do projektu zaliczeniowego nr 2

Celem projektu jest napisanie prostej hierarchii klas w języku Python (z wykorzystaniem abstrakcyjnej klasy bazowe, deskryptorów i szczątkowym wykorzystaniem biblioteki `tkinter`).

- **Gotowy program należy zaprezentować osobiście na konsultacjach oraz przesłać na platformę Moodle.**
- Wymagana jest bezwzględna znajomość prezentowanego kodu źródłowego tzn. trzeba umieć odpowiedzieć na pytanie: "Co to jest i do czego służy?".
- Program powinien działać poprawnie wykonując zamierzone przez programistę zadania.
- Program musi oczywiście wykorzystywać paradygmat programowania obiektowego.
  W tym:
  - należy tworzyć własne klasy;
  - wykorzystywać dziedziczeni.
- Ponadto należy zadbać o:
  - poprawne rozmieszczenie kodu w pliku źródłowym (lub plikach);
  - czytelność kodu i konsekwentnie stosować styl nazewniczy;
  - ogólną estetykę kodu i działanie programu.
- **Opis projektu:** Opracuj i zaimplementuj odpowiednią hierarchię dziedziczenia klas opartą o abstrakcyjną klasę bazową `ConvexPolygon` posiadającą cztery abstrakcyjne metody: `__init__()`, `area()`, `perimeter()`, `draw()`. Konstruktor powinien być również metodą konkretną, która tworzy dwa atrybuty: `fill_colour`, `otuline_colour`. Zdefiniuj następujące klasy: `Triangle`, `ConvexQuadrilateral`, `RegularPentagon`, `RegularHexagon` oraz `RegularOctagon`, które dziedziczą po klasie `ConvexPolygon`. Następnie zdefiniuj takie klasy jak: `IsoscelesTriangle`, `EquilateralTriangle`, `Parallelogram`, `Kite`, `Rhombus` oraz `Square` zachowując odpowiednie relacje dziedziczenia (tj. na przykład każdy kwadrat to prostokąt i romb, każdy prostokąt to równoległobok itp.). Na koniec napisz prosty program (proste menu tekstowe), który umożliwia użytkownikom wybór wielokąta dowolnego typu, wprowadzanie potrzebnych danych geometrycznych, a następnie

program wyświetla jego pole, obwód i rysuje na płótnie (obiekt klasy `Canvas` z modułu `tkinter`). Wykorzystaj klasę deskryptora do zarządzania atrybutami klas. W szczególności, długości boków (`length_of_side_a`, `length_of_side_b` itp.) powinny być dodatnimi liczbami rzeczywistymi (`numbers.Real`), kolory wypełnienia i obramowania (`fill_colour`, `otuline_colour`) - ciągami tekstowymi ze definiowanej wcześniej listy kilkunastu kolorów, kąty (`angle`) - np. liczbami z zakresu $(0, \frac{\pi}{2})$, stosunki (`ratio`) - liczbami z zakresu $(0, 1)$.

Uwagi i wskazówki: Zastanów się w jaki sposób można "dezaktywować" niektóre zbędne instancje deskryptorów w klasach pochodnych. Definiując czworokąt (`ConvexQuadrilateral`) może poprosić użytkownika o podanie długości jego przekątnych, kąta i stosunków ich przecięcia. Przydatne może okazać się twierdzenia Pitagorasa i cosinusów. Odpowiednią klasę deskryptora znajdziesz w materiałach z wykładów. Rysując figury możesz wykorzystać informacje z geometrii analitycznej (lub/i liczb zespolonych). Wygodnie jest rysować daną figurę zaczynając od punktu $(0, 0)$ i w odpowiednich miejscach wykorzystać przesunięcia o wektor, tak aby narysować ją np. na środku ekranu.