Lista 9

Uwaga. Do każdego z zadań napisz (proste, z użyciem `assert`) testy ilustrujące działanie napisanych klas.

Zadanie 1 (1 punkt). Napisz klasę `Car`, reprezentującą samochód. Klasa powinna przechowywać następujące dane: maksymalną prędkość samochodu, pojemność paliwa (w dowolnej jednostce, np. l, kg, kWh) i zużycie paliwa na 100 km; oraz zawierać następujące metody:

- `__init__(self, max_speed, max_fuel, fuel_consumption)` — konstruktor tworzący samochód o maksymalnej prędkości `max_speed`, pojemności paliwa `max_fuel` i zużyciu `fuel_consumption` paliwa na 100 km.
- `max_travel_time(self)` — maksymalny czas jazdy samochodu w godzinach (rozpoczynającego podróż z pełnym bakiem/baterią) aż do wyczerpania paliwa.
- `max_range(self)` — zwraca zasięg samochodu, czyli maksymalną ilość km, jakie samochód `self` może przebyć pomiędzy uzupełnieniami paliwa.
- `total_travel_time(self, distance)` — Zwraca czas (w godzinach) potrzebny samochodowi `self` na przebycie `distance` km. `distance` może przekraczać zasięg samochodu. Przyjmij, że samochód rozpoczyna podróż pełny paliwa, zawsze porusza się z maksymalną prędkością aż do wyczerpania go, a uzupełnianie paliwa trwa 10 minut.

Zadanie 2 (1,5 punktu). Napisz **własną** klasę `Fraction` reprezentującą ułamki. Zaimplementuj następujące metody specjalne[1]:

- `__init__(self, numer=0, denom=1)` — konstruktor tworzący ułamek `numer`/`denom`.
- `__abs__(self)` — zwraca ułamek odpowiadający |`self`|.
- `__neg__(self)` — zwraca ułamek odpowiadający −`self`.
- `__str__(self)` — zwraca napis reprezentujący `self`.
- `__add__(self, other_frac)` — zwraca ułamek odpowiadający `self` + `other_frac`.
- `__sub__(self, other_frac)` — zwraca ułamek odpowiadający `self` − `other_frac`.
- `__mul__(self, other_frac)` — zwraca ułamek odpowiadający `self` · `other_frac`.
- `__truediv__(self, other_frac)` — zwraca ułamek odpowiadający `self` / `other_frac`.
- `__eq__(self, other_frac)` — sprawdza, czy `self` = `other_frac`.
- `__lt__(self, other_frac)` — sprawdza, czy `self` < `other_frac`.

Zarówno licznik jak i mianownik ułamka powinny być przechowywane w postaci reprezentującej **nieskracalny** ułamek. Znak ułamka powinien być przechowywany w liczniku[2]. Do testów możesz użyć analogicznej klasy `Fraction` z modułu `fractions`.

Zadanie 3 (1,5 punktu). Napisz klasę `Triangle` reprezentującą domknięty trójkąt na płaszczyźnie, zawierającą następujące metody:

- `__init__(self, x1, y1, x2, y2, x3, y3)` — konstruktor tworzący trójkąt, którego wierzchołki (w kolejności przeciwnej do ruchu wskazówek zegara) to kolejno (`x1`, `y1`), (`x2`, `y2`), (`x3`, `y3`).
- `circumference(self)` — zwraca obwód trójkąta `self`.
- `area(self)` — zwraca pole trójkąta `self`.
- `contains_point(self, x, y)` — zwraca `True`, jeśli trójkąt `self` zawiera punkt o współrzędnych (`x1`, `y1`) i `False` w przeciwnym wypadku.
- `contains(self, other_triangle)` — zwraca `True`, jeśli trójkąt `other_triangle` zawiera się w trójkącie `self`, `False` w przeciwnym wypadku.
- `contained_in(self, other_triangle)` — zwraca `True`, jeśli trójkąt `self` zawiera się w trójkącie `other_triangle`, `False` w przeciwnym wypadku.
- (za 0,25 punktu) `draw(self)` — rysuje i wyświetla trójkąt `self` z użyciem biblioteki `matplotlib`.

Możesz założyć, że reprezentujemy jedynie trójkąty niezdegenerowane.

[1] Metody w tym zadaniu przeciążają odpowiednie operatory i operacje specjalne (np. `abs(x)`).
[2] Zadanie 2 z Listy 4.