AGH AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

Python – Podstawy programowania obiektowego

1. **Proszę zapoznać się z przykładowym kodem klasy w języku Python:**

```
class Bike():  # Definicja klasy Bike
    vehicle_type = "two-wheeler"  # Pole statyczne klasy

    def __init__(self, wheel, color):  # konstruktor klasy Bike z argumentami
        self.wheel = wheel  # pola klasy Bike
        self.color = color

    def wheel_get(self):  # metoda klasy Bike, która zwraca wartość pola wheel, tzw. "getter"
        return self.wheel

    def color_get(self):  # metoda klasy Bike, która zwraca wartość pola color, tzw. "getter"
        return self.color

    def color_set(self, new_color):  # metoda klasy Bike, która ustawia wartość pola color,
tzw. "setter"
        self.color = new_color


my_bike = Bike(2, "red")  # obiekt klasy Bike

print(my_bike.vehicle_type)  # wyświetlamy zmienna publiczną instancji (taka sama wartość
niezalenie od obiektu)

print(my bike.wheel get())  # wyświetlamy atrybut wheel klasy Bike

print(my_bike.color_get())  # wyświetlamy atrybut color klasy Bike
my_bike.color_set("green")  # zmieniamy atrybut klasy Bike
print(my_bike.color_get())  # ponownie wyświetlamy atrybut color klasy Bike
```

2. **Proszę zdefiniować nową klasę *Car* , która będzie posiadała następujące atrybuty:**

(a) Pola klasy: *wheels, engine, max_velocity, color, seats*

(b) Metody klasy pobierające (gettery) i zapisujące (settery) dane do pól klasy.

- Po zdefiniowaniu klasy proszę stworzyć kilka (minimum 3 różne) obiekty klasy *Car*, dla każdego podając inne wartości wymaganych atrybutów.

- Proszę w każdej instancji klasy zmienić za pomocą settera wartość co najmniej jednego atrybutu.

- Proszę przy pomocy gettera wyświetlić na ekran wartość jednego atrybutu każdego ze stworzonych obiektów. Czy za pomocą gettera można zmienić wartość atrybutu obiektu ? Dlaczego ?

3. Proszę zapoznać się z przykładem kodu przedstawiającym dziedziczenie atrybutów klasy przez inną klasę:

```
class Ciasto():
    def __init__(self, wielkosc, smak):
        self.wielkosc = wielkosc
        self.smak = smak

    def informacja(self):
        print("To ciasto ma rozmiar " + self.wielkosc + " i smak " + self.smak)


class Tort(Ciasto):
    def __init__(self, wielkosc, smak):
        Ciasto.__init__(self, wielkosc, smak)

    def Tort(self):
        print("Ten tort ma rozmiar " + self.wielkosc + " i smak " + self.smak)


ciasto_czekoladowe = Ciasto("średni", "czekoladowy")  # obiekt klasy Ciasto

ciasto_czekoladowe.informacja()  # wywołanie metody klasy Ciasto --> "To ciasto ma rozmiar średni i smak
czekoladowy"

tort_truskawkowy = Tort("duży", "truskawkowy")  # Obiekt klasy Tort dziedziczący metody i pola klasy Ciasto

tort_truskawkowy.Tort()  # ---> "Ten tort ma rozmiar duży i smak truskawkowy"

tort_truskawkowy.informacja()  # # ---> "To ciasto ma rozmiar duży i smak truskawkowy"
```

Proszę zauważyć w powyższym przykładzie, że metoda *informacja* może być wywołana jako metoda obiektu *tort_truskawkowy* mimo, że sama nie jest w niej zdefiniowana.

Na podstawie powyższego przykładu proszę zdefiniować klasę *Electric*, która będzie dziedziczyła po klasie *Car*, a następnie zdefiniować w niej kilka nowych pól i metodę wyświetlającą WSZYSTKIE pola tej klasy na ekran. Proszę utworzyć obiekt klasy *Electric* i przy pomocy gettera z klasy *Car* wyświetlić liczbę kół tego obiektu.

Wybierz **jedno** z poniższych zadań które wykonasz w ramach pracy domowej.

1. Zadanie domowe na ocenę max 80%:

Zmodyfikuj rozwiązanie zadania domowego z poprzednich zajęć w taki sposób aby zamiast funkcji użyć odpowiednio zdefiniowanych klas:

-Stworzyć klasę Solid() która przechowuje wspólne metody i pola takie jak:
name – pole statyczne klasy przechowujące nazwe bryły
rho – pole statyczne klasy przechowujące wartość gęstości
raport() – Metoda klasy wyświetlająca za pomocą print informację o wszystkich właściwościach bryły (Nazwa, wymiary, gęstość, objętość, masa, pole powierzchni)
masa() – Metoda klasy obliczająca masę bryły na podstawie gęstości i funkcji obliczającej objętość

-6 klas definiujących poszczególne bryły. Każda z klas powinna dziedziczyć po klasie Solid aby mieć dostęp do jej metod i pól statycznych. Konstruktor każdej z klas powinien pozwolić na zdefiniowanie: wymiarów geometrycznych, gęstości i nazwy bryły a metody powinny dawać możliwość obliczenia pola powierzchni i objętości.

Stwórz po jednym obiekcie odpowiadającym każdej z brył i nadaj im wybrane przez siebie nazwy oraz właściwości geometryczne i gęstość.
Umieść stworzone obiekty w liście następnie przy użyciu pętli for wykonaj na każdym elemencie tej listy metodę raport() i wyświetl wynik w konsoli.

2. Zadanie domowe na ocenę max 100%:

Proszę stworzyć dwie klasy: samochód oraz parking. Klasa samochód powinna zawierać pola: numer rejestracyjny, kolor, typ pojazdu (osobowy, ciężarowy, jednoślad) oraz metody: wjazd na parking, wyjazd z parkingu, wyświetlenie na ekranie wszystkich informacji o pojeździe. Klasa parking powinna posiadać pola: całkowita liczba miejsc (wartość początkowa 5), obecna liczba zajętych miejsc (wartość początkowa 0), utarg (wartość początkowa 0), listy numerów rejestracyjnych samochodów które z parkingu korzystały z rozróżnieniem na typ pojazdu, metody umożliwiające wjazd samochodu na parking oraz jego opuszczenie. Przy wyjeździe naliczana jest opłata w zależności od typu pojazdu (np. osobowe 10, ciężarowe 30, jednoślady 5).

W celu przetestowania stworzonych klas należy stworzyć funkcję testującą w której tworzone jest 6 obiektów klasy samochód (2 osobowe, 2 ciężarowe i 2 jednoślady) i obiekt klasy parking, a następnie przeprowadzić symulację następujących operacji:

1. samochod1 wjeżdża na parking,
2. samochód2 wjeżdża na parking,
3. samochód3 wjeżdża na parking,
4. samochód2 opuszcza parking,
5. parking wyświetla liczbę zajętych miejsc
6. samochód2 wjeżdża ponownie na parking,
7. samochód4 wjeżdża na parking,
8. samochód5 wjeżdża na parking,
9. Parking wyświetla liczbę zajętych miejsc oraz bieżący utarg
10. samochód6 próbuje wjechać na parking (parking zapełniony, wyświetla się odpowiedni komunikat)
11. samochód1 opuszcza parking,
12. samochód6 wjeżdża na parking
13. parking wyświetla liczbę zajętych miejsc oraz bieżący utarg
14. wszystkie samochody opuszczają parking
15. parking wyświetla liczbę zajętych miejsc oraz bieżący utarg
16. parking wyświetla listę numerów rejestracyjnych wszystkich pojazdów jakie z niego korzystały.
17. parking wyświetla listę numerów rejestracyjnych samochodów ciężarowych jakie z niego korzystały.

Sposób oceniania:

Pola klasy samochód: 10%
Metody klasy samochód: 10%
Pola klasy parking: 20%
Metody klasy parking: 20%
Funkcja testująca: 40%