Kolokwium poprawkowe 1

Informacje wstępne

1. Klonujemy repozytorium o nazwie PO-nr_indeksu-zaliczenia do wybranego katalogu.
2. W PyCharm w sklonowanym katalogu z repozytorium tworzymy projekt o nazwie K1P-nr_indeksu, np. K1P-12345.
3. W projekcie tworzymy 3 pliki: element.py, file.py oraz directory.py
4. W trakcie kolokwium można korzystać z narzędzi wbudowanych w PyCharm lub zainstalowanych pakietów takich jak: flake8 lub mypy w celu sprawdzenia typowania i PEP8.
5. W przypadku braku internetu student zapisuje projekt jako archiwum zip. Kolokwia będą zebrane np. za pomocą pendrive.
6. Rozwiązania po terminie: z uwagi na rozbieżność czasu na stanowiskach w pracowni bieżący czas będzie wyświetlony za pomocą rzutnika. Studenci zostaną uprzedzeni ustnie o zbliżającym się końcu czasu w okresie między 10 a 5 minut przed końcem kolokwium. Brak przesłania repozytorium w wyznaczonym czasie powoduje dodanie dodatkowych 5 minut, ale wtedy maksymalna ocena to 3,5. Powyżej 5 minut spóźnienia - kolokwium pozostaje bez sprawdzenia z oceną niedostateczną. Czas liczony jest od momentu udostępnienia poleceń do poprawnego wypchnięcia na zdalne repo na Githubie lub utworzenia pliku zip w wypadku braku internetu.

Zadanie 1. Klasa Element (pol. Element) (8pt max.)

1. Napisz klasę Element. Klasa powinna posiadać prywatne atrybuty instancyjne:
    1. name (pol. Nazwa), typ string.
    2. year, (pol. Rok), typ int.
2. Zaimplementuj:
    1. **(3pt)** Inicjalizator z dwoma argumentami (o nazwach i w kolejności jak w pt 1), gdzie wartościami domyślnymi dla name będzie None, a dla year 1970. Zadbaj również o to, aby data utworzenia pliku nie mogła być sprzed 1970 roku oraz nie mogła być po 2023 roku. W razie podania nieprawidłowego roku, ustaw argument year na 1970.
    2. **(2pt)** Zaimplementuj właściwości (propercje) setter i getter dla każdego atrybutu. Jeśli dla setterów podane wartości argumentów nie spełniają założeń, wartość atrybutu nie powinna się zmieniać i powinien zostać wypisany odpowiedni komunikat.
3. Nadpisz:
    1. **(1pt)** Odpowiednią metodę magiczną, która zwróci reprezentację tekstową bieżącej instancji klasy: „Nazwa: Element1. Utworzony 1999.”
    2. **(2pt)** Metody magiczne __eq__, __ne__ porównujące obiekty po nazwie oraz roku utworzenia.

Zadanie 2. Klasa File (pol. Plik) oraz klasa Directory (pol. Katalog) (5pt+5pt max.)

1. Zaimplementuj klasę File dziedziczącą po klasie Element. Klasa ta powinna posiadać jeden atrybut instancyjny prywatny:
    1. size (pol. rozmiar), typu int
2. **(2pt)** Nadpisz inicjalizator (argumenty o nazwach takich jak atrybuty, w kolejnosci jak w pt 1). Domyślna wartość dla size to 0. Jeśli rozmiar jest ujemny powinna być ustawiona domyślna wartość.
3. **(1pt)** Zaimplementuj właściwości (propercje) setter i getter dla atrybutu. Jeśli dla settera podane wartości argumentów nie spełniają założeń, wartość atrybutu nie powinna się zmieniać i powinien zostać wypisany odpowiedni komunikat.
4. **(1pt) Nadpisz** metody magiczne __eq__, __ne__ tak by teraz jeszcze porównywały rozmiar.

5. **(1pt) Nadpisz** metodę magiczną, która zwróci reprezentację tekstową bieżącej instancji klasy: „Nazwa: PlikXYZ. Utworzony 1999. Rozmiar 123B”
6. Zaimplementuj klasę Directory dziedziczącą po klasie Element. Klasa ta powinna posiadać jeden atrybut instancyjny prywatny:
    1. elements (pol. elementy), typu list[File]
7. **(2pt)** Nadpisz inicjalizator (argumenty o nazwach takich jak atrybuty, w kolejnosci jak w pt 1). Domyślna wartość dla listy to [].
8. **(1pt)** Zaimplementuj właściwości (propercje) setter i getter dla atrybutu.
9. **(1pt)** Zaimplementuj metodę add_element(plik), która dodaje do listy elementów plik.
10. **(1pt)** Zaimplementuj metodę zwracającą rozmiar katalogu (suma rozmiarów wszystkich plików).
11. **(1pt) Nadpisz** metodę magiczną, która zwróci reprezentację tekstową bieżącej instancji klasy: „Nazwa: KatalogXYZ. Utworzony 1999. Rozmiar 1234B. Zawartość: [Nazwa: PlikXYZ. Utworzony 1999. Rozmiar 123], [Nazwa: PlikABC. Utworzony 1985. Rozmiar 567]”

Polecenia pomocnicze do testowania klas. Plik main.py z funkcją main() (Polecenia mają na celu pomoc w testowaniu napisanych klas)

```
e1 = Element("element1", 1998)
print(e1)
e2 = Element("element1", 1960)
print(e2)
print(e1 != e2)
print(e1 == e2)
e2.year = 1998
print(e2)
print(e1 != e2)
print(e1 == e2)
```

Podobnie możesz przetestować pozostałe klasy.