Klasa Product

1. Utwórz klasę `Product` która będzie odzwierciedlała obiekt zakodowany w JSON (plik `product.json`).
2. Konstruktor przyjmuje za parametr nazwę pliku zawierającego JSON, sprawdza czy plik istnieje, jeżeli nie, wyrzuca wyjątek `ProductFileNotFound`. Jeżeli plik istnieje przypisuje do pól klasy wartości z odkodowanego pliku.
3. Dodaj do klasy pole `amount` zawierające wartość produktu (`quantity` * `price`), utwórz do niego `getter` korzystając z `lazy loading`.
4. Korzystając z metody magicznej stwórz możliwość wyświetlenia instancji tej klasy w formacie:
   `pole: wartosc_pola`
   `pole: wartosc_pola`
   …
   W tej funkcji uwzględnij aby nie wyświetlać pola `amount`.

Klasa ProductVariation

1. Utwórz klasę `ProductVariation` która będzie rozszerzała klasę `Product`.
2. Dodaj pole `color` tak aby podczas tworzenia instancji, jego wartość była wymagana i była `strinigiem`, w przeciwnym razie rzuć wyjątkiem `UndefinedVariantColor`.

Interfejs Item

1. Utwórz interfejs `Item` który będzie zawierał definicję metod `getId` oraz `getNet`.
2. `getId` zwraca id produktu/wariantu.
3. `getNet` zwraca wartość netto pola `price` (VAT 23%).
4. Interfejs zaimplementuj w klasach `Product` oraz `ProductVariation`.

Klasa Products

1. Utwórz klasę `Products` która będzie rozszerzała klasę `\ArrayIterrator` (klasa zawarta w `SPL`).
2. Spraw aby ta klasa zawierała zbiór instancji klas implementujących interfejs `Item`.

Klasa Basic

a. Utwórz klasę `Basic` która będzie zawierała funkcje których działanie będzie oparte na pracy z klasami `Product` oraz `ProductVariation`.
b. Utwórz funkcję `generateRandomItems` która za parametr przyjmie ilość losowo generowanych produktów a finalnie zapisze instancję klasy `Products`, zawierającą te produkty, jako plik JSON. Co drugi produkt powinien być instancją klasy `ProductVariation` z losowo wybranym kolorem z puli kolorów `red`, `green`, `blue`, `white`, `black`, `null` (wartość `null`) (pomocnicza tablica wewnątrz funkcji). Funkcja zwraca unikalne id pliku JSON. Pliki JSON powinny być przechowywane w katalogu `products`. W przypadku niepowodzenia utworzenia wariantu o kolorze `null` złap wyjątek, wyświetl informację dla użytkownika i pomiń ten produkt.
c. Utwórz funkcję `showItems` która na podstawie unikalnego id pliku pobierze plik z wygenerowanymi produktami przez funkcję `generateRandomItems` a następnie wyświetli o nich dane. Wykorzystaj do tego magiczną metodę z pkt. 4 w sekcji **Klasa Product**.

Poza tagiem `<pre>` nie używaj innych tagów HTML. Jeżeli plik nie istnieje utwórz go poprzez `generateRandomItems` z parametrem 20.

d. Utwórz funkcję `deleteProducts` usuwającą plik o unikalnym ID którzy będzie przekazywany jako parametr tej funkcji. Funkcja zwraca wartość `boolean` w przypadku powodzenia usunięcia pliku. (Jeżeli plik o podanym id nie istnieje zwróć wartość `true`).

Zadania

1. W pliku `index.php` strukturalnie utwórz autoloader klas korzystając z `spl_autoload_register()`. W przypadku braku klasy wyrzuć wyjątkiem `\Exception` (SPL).
2. Zainicjuj instancję klasy `Basic` na której będziesz pracował przy zadaniach poniżej.
3. Następnie sprawdź czy skrypt uruchamiany jest poprzez wiersz poleceń
    a. W przypadki uruchomienia przez wiersz poleceń umożliw użytkownikowi podanie argumentu podczas uruchamiania skrypty, na podstawie pierwszego argumentu (pozostałe zignoruj) wykonaj poniższe działania:
        i. Dla argumentu `generate` wygeneruj losowe produkty (`Basic::generateRandomItems`).
        ii. Dla argumentu innego niż `generate` przyjmij że jest on unikalną nazwą pliku JSON w katalogu `products` i usuń go. Wyświetl informację o powodzeniu/niepowodzeniu działania.
    b. W przypadku gdy skrypt uruchamiany jest przez przeglądarkę korzystając z funkcji `Basic::showItems` wyświetl informacje o wszystkich produktach z wszystkich plików w katalogu `products` (każdy produkt w osobnej linii, każdy plik oddzielony znakami '*', znaki '*' powinna poprzedzać nazwa pliku). Jeżeli produkt będzie wariantem dodatkowo wyświetl jego kolor.