FooDev: Pizza - OOP

OOP, static, self, névtér, faker, Stringable, json

Rostagni Csaba

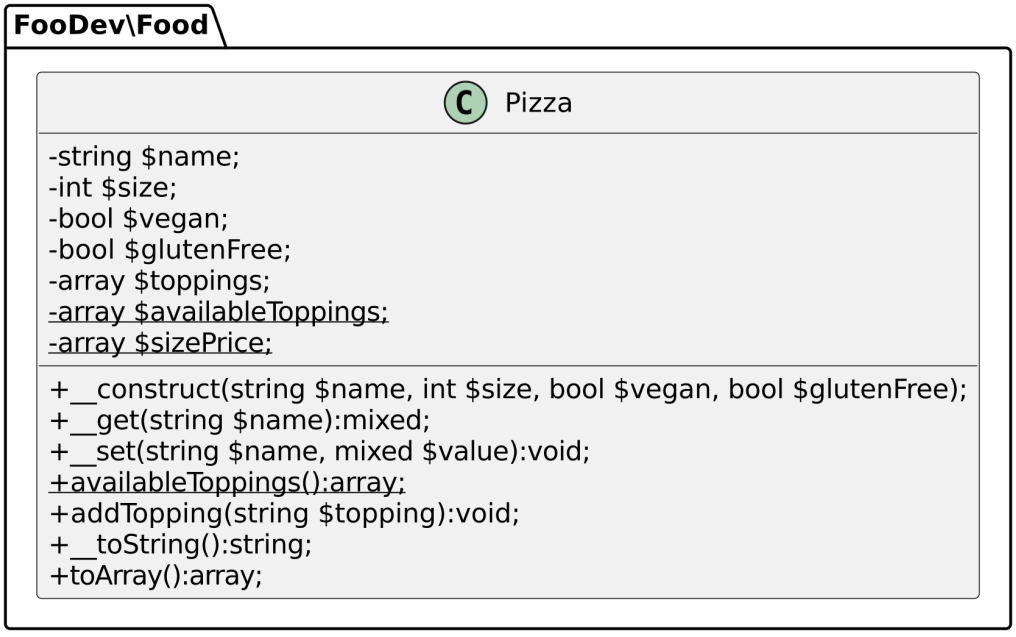
Ön a **FooDev** cég fejlesztési részlegén dolgozik. Az aktuális feladat a rendelhető ételeket kezelő alkalmazás készítése. Az első tennivaló a `Pizza` osztály létrehozása.

A megoldás elkészítése során figyeljen oda, hogy bizonyos információkat az UML ábrából kell meghatározni, amennyiben nem tartalmazza a leírás.

Előkészületek

1. Hozzon létre egy projektmappát, amiben dolgozni fog!
2. Amennyiben nem lenne elérhető, úgy "telepítse" a composert.
3. Vegye fel a `fakerphp/faker` csomagot a projekthez.
4. A projektmappában hozzon létre egy mappát `out` néven a kimeneti fájloknak.
5. Hozzon létre egy mappát `src` néven. A saját készítésű osztályok itt helyezkedjenek el a névtér szerint úgy, hogy a PSR-4-es szabványnak feleljenek meg!
6. Ahogy az UML ábrán is látható, a felső szintű (vendor) névtér legyen *"FooDev"*! Az alnévtér neve legyen *"Food"*. Hozza létre az ennek megfelelő mappaszerkezetet az `src` mappán belül.
7. Módosítsa `composer.json` fájlt, ahol az autoload PSR-4 szabvány szerint a `FooDev` névtérhez tartozó kódot az `src/FooDev` mappában keresi.
8. Hozzon létre egy `.gitignore` fájlt, ami alapján ignorálja a gyökér mappában található `vendor` és `out` mappákat a git verziókezelő!

Pizza



1. ábra. Pizza osztálydiagram

9. Hozza létra a `Jegy` nevű osztályt a `FooDev\Food` névtérben.
10. Vegye fel a példány szintű tulajdonságokat az osztálydiagramnak megfelelően.
11. Készítse el a konstruktort, ami az UML ábrán látható sorrendben kapja meg az értékeket.
    - A pizza neve biztosítsa, hogy "Szókezdő Nagybetűkkel Tárolja Az Nevét".
12. Minden **létező** tulajdonság kapjon **gettert** és **settert** is, magic method segítségével.
    - Amennyiben a `property_exists` függvény szerint nem létezik a tulajdonság, úgy ne csináljon semmit.
    - A ki- és bemeneti értékek típusát az osztálydiagram szerint határozza meg!
13. Hozzon létre egy **osztályszintű tulajdonságot** `availableToppings` néven, ami a lehetséges feltételeket tartalmazza egy tömbben. A tömbhöz az adatokat a `toppings.txt` fájlban találja.
14. Hozzon létre egy **osztályszintű metódust** `availableToppings` néven, ami visszaadja a választható feltételeket.
15. Bővítse az osztályt egy `addTopping` nevű metódussal, ami a paraméterként kapott feltétet hozzáadja a tényleges feltételeket tartalmazó `$toppings` tömbhöz.
16. Hozzon létre egy **osztályszintű tulajdonságot** `$sizePrice;` néven, ami a különböző méretekhez tartozó alapárakat tartalmazza asszociatív tömbként. Az árak euróban értendőek

méret	ár
24	10
32	15
55	20

17. Módosítsa úgy a `__get` magic method-ot úgy, hogy amennyiben a `price` tulajdonságot szeretnék elérni, úgy számítsa ki az árat:
    - Válassza ki a mérete alapján az alapárat
    - Minden egyes feltét ára plusz 1€
18. Az osztály implementálja a `Stringable` interface-t, és valósítsa meg a `__toString()` metódust, ami a leírásnak és a mintának megfelelően megjeleníti az osztály tulajdonságait:
    - A pizza nevét nagybetűvel írja ki
    - Az árát zárójelbe írja ki
    - Az ár után - még a zárójelben - jelenjen meg a pénznem is (€).

```
PRAESENTIUM INVENTORE ELIGENDI (14 €)
```

19. Bővítse ki az osztályt egy `toArray` nevű metódussal, ami asszociatív tömbként adja vissza az alábbi értékeket:
    - `name`: a pizza neve.
    - `size`: a pizza mérete.
    - `vegan`: `true`, ha vegán, egyébként `false`.
    - `glutenFree`: `true`, ha gluténmentes, egyébként `false`.
    - `price`: a pizza ára, figyelembe véve a méretét és a feltételeket. Egész szám, a pénznemet NE tartalmazza!
    - `toppings`: az összes feltétet tartalmazó szöveges tömb.

console.php

20. Hozzon létre a projektmappában egy fájlt `console.php` néven. Ez lesz a konzolos szkript belépési pontja.
    - Állítsa be, hogy a szkript szigorúan kezelje a típusokat!
    - Minden hibát figyelmeztetést, stb… jelenítsen meg!
    - Alkalmazza a composer autoloadját!
    - Hozzon létre megfelelő példányt a Faker-ből, ami **olasz** nyelven (`it_IT`) generál adatokat.
21. A szkript ellenőrizze a paraméterek számát. Amennyiben 1-nél kevesebb paramétert kapott, úgy tájékoztassa a felhasználót, majd lépjen ki 9-es hibakóddal.

    ```
    Az első paraméter megadása kötelező!
    ```

22. A szkript első paramétere a legenrálandó pizzák száma. Ellenőrizze, hogy csak szám lehet, és az értéke legalább 1. Egyéb esetben adjon hibaüzenetet, majd lépjen ki 4-es hibakóddal.

    ```
    $ php console.php hét
    Nem megfelelő paraméter!
    Az első paraméternek 0-nál nagyobb számnak kell lennie!
    ```

23. A szkript második paramétere opcionális. Ellenőrizze, hogy csak `V`, vagy `G` lehet. Egyéb esetben adjon hibaüzenetet, majd lépjen ki 7-es hibakóddal.

    ```
    $ php console.php 5 C
    Nem megfelelő paraméter!
    A második paraméter csak 'V' (vegán), vagy 'G' (glutén mentes)
    ```

24. Hozzon létre egy tömböt `pizzas` néven, amiben eltárolja a generált pizzákat.
25. Példányosítson az első paraméterben meghatározott darabszámú Pizzát. A konstruktort a faker segítségével paraméterezze fel!
    - A `name` értéke legyen legfeljebb 4 szóból álló véletlen szöveg.
    - A `size` értéke csak `24`, `32` és `55` értékek egyikét veheti fel.
    - A `vegan` és `glutenFree` egye-egy véletlenszerű logikai értéket kapjon.
26. Jelenítse meg az összes pizzát a `toString` felhasználásával.
27. Amennyiben nem lett meghatározva második paraméter, úgy az összes pizza adatát írja ki a `orders.json` fájlba, JSON formátumban.
28. Amennyiben a második paraméterben meg lett határozva egy speciális típús,
    - úgy annak megfelelően hozza létre az `orders-v.json` vagy `orders-g.json` fájlokat,
    - majd **csak** a meghatározott étrendnek megfelelő pizzákat írja ki JSON formátumban.