

# Fagyi

## OOP dogozat

Rostagni Csaba

Ön a **Acme** cég fejlesztési részlegén dolgozik. Az aktuális feladat mélyhűtött termékek árusításával foglalkozó szoftver elkészítése. Az első lépésben a fagyfaltokat kell megvalósítani.

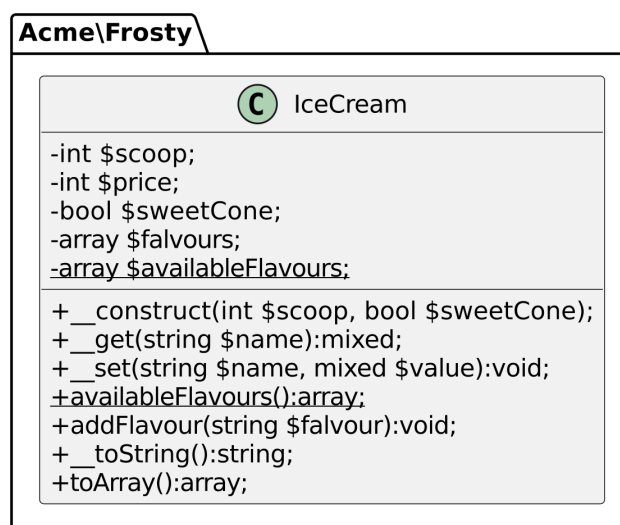
A megoldás elkészítése során figyeljen oda, hogy bizonyos információkat az UML ábrából kell meghatározni, amennyiben nem tartalmazza a leírás.

Az osztály írása során minden esetben határozza meg a megfelelő típust, első sorban az UML diagram alapján!

### Előkészületek

1. Hozzon létre egy projektmappát **vezeteknev-keresztnev-frosty** néven, amiben dolgozni fog!
2. Vegye fel a **fakerphp/faker** csomagot a projekthez.
3. A projektmappában hozzon létre egy mappát **out** néven a kimeneti fájloknak.
4. 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!
5. Ahogy az UML ábrán is látható, a felső szintű (vendor) névtér legyen *“Acme”*! Az alnévtér neve legyen *“Frosty”*. Hozza létre az ennek megfelelő mappaszerkezetet az **src** mappán belül.
6. Módosítsa **composer.json** fájlt, ahol az autoloader PSR-4 szabvány szerint a **Acme** névtérhez tartozó kódot az **src/Acme** mappában keresse.
7. Hozzon létre egy **.gitignore** fájlt, ami alapján ignorálja a gyökér mappában található **vendor** és **out** mappákat a git verziókezelő!

### IceCream



1. ábra. IceCream osztálydiagram

8. Hozza létre az **IceCream** nevű osztályt a **Acme\Frosty** névtérben.
9. Vegye fel a példány szintű tulajdonságokat az osztálydiagramnak megfelelően.

10. Készítse el a konstruktort, ami az UML ábrán látható sorrendben kapja meg a gombócok számát (**scoop**) és hogy édes tölcsérben lett -e kiadva (**sweetCone**).
  - A fagylalt árát az alábbiak szerint állítsa be vele:
    - Minden gombóc 400 Ft-ba kerül,
    - az édes tölcsér 80 Ft- felárat jelent.
11. Minden **létező** tulajdonság kapjon **gettert** és **settert** is, magic method segítségével.
  - Amennyiben nem létezik a tulajdonság, úgy getter esetén **null** értékkel térjen vissza, setter esetén ne csináljon semmit.
  - A ki- és bemeneti értékek típusát az osztálydiagram szerint határozza meg!
12. Hozzon létre egy **osztályszintű tulajdonságot** **availableFlavours** néven, ami a lehetséges ízeket tartalmazza egy tömbben. Vegye fel a tömbbe a *“vanilla”*, *“strawberry”*, *“chocolate”*, *“coconut”* értékeket.
13. Hozzon létre egy **osztályszintű metódust** **availableFlavours** néven, ami visszaadja a választható ízeket.
14. Az osztály implementálja a **Stringable** interface-t, és valósítsa meg a **\_\_toString()** metódust, ami a leírásnak és a mintának megfelelően megjeleníti az osztály tulajdonságait:
  - Jelenítse meg a gombócok számát, ezt a *“gombócos fagylalt”* szöveg kövesse
  - Utána szögletes zárójelben, vesszővel elválasztva és nagybetűsen sorolja fel az ízeket, ezt a *“ízekkel”* szöveg kövesse
  - Utána amennyiben a **sweetCone** értéke igaz, úgy *“édes tölcsérben”*, különben *“normál tölcsérben”* szöveg szerepeljen
  - A legvégén az ár szerepeljen zárójelben, az Ft mértékegység is szerepeljen.

```
2 gombócos fagylalt [VANILLA, VANILLA] ízekkel édes tölcsérben (880 Ft)
```

15. Bővítsé ki az osztályt egy **toArray** nevű metódussal, ami tömbként adja vissza az alábbi értékeket:
  - a gombócok száma
  - az ízek vesszővel és szóközzel elválasztva
  - **édes**, ha az **sweetCone** értéke igaz volt, egyébként **normál**
  - a fagylalt ára, formázatlanul és pénznem **nélkül**

Lehetséges értékek:

```
[ 3, "vanilla, vanilla, strawberry", "édes", 1280]
```

## demo.php

17. Hozzon létre a projektmappában egy fájlt **demo.php** néven. Ez lesz a konzolos szkript belépési pontja.
  - Állítsa be, hogy a szkript szigorúan kezelje a típusokat!
  - Alkalmazza a composer autoloaderjét!
  - Hozzon létre megfelelő példányt a Faker-ből, ami **magyarul** (hu\_HU) generál adatokat.
18. A szkript ellenőrizze a paraméterek számát. Amennyiben 1-nél kevesebb paramétert kapott, úgy tájékoztassa a felhasználót, majd lépjen ki 7-es hibakóddal.

```
$ php demo.php
Legalább 1 paraméter megadása szükséges!
```

19. A szkript első paramétere a legenerálandó fagylaltok száma. Ellenőrizze, hogy csak szám lehet, és az értéke legalább 1. Egyéb esetben adjon hibaüzenetet, majd lépjen ki 17-es hibakóddal.

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

20. Hozzon létre egy tömböt `icecreams` néven, amiben eltárolja a generált fagyikat.
21. Példányosítson az első paraméterben meghatározott darabszámú fagyaltot. A konstruktort a `faker` segítségével paraméterezze fel!
  - A `scoop` 1 és 5 közötti szám, a határokat is beleértve.
  - A `sweetCone` egy véletlenszerű logikai értéket kapjon.
  - Minden fagyaltnak annyi ízt adjon hozzá, ahány gombócos.
22. Jelenítse meg az összes fagyaltot külön sorban a `__toString()` felhasználásával.
23. Minden fagyalt árát csökkentse 10 százalékkal! A kapott értéket kerekítse egészre, és módosítsa a fagyaltok árát!
24. Az `out` mappában hozza létre a `sale.csv` fájlt, ami az összes leggenerált fagyalt összes adatát tartalmazza a **kedvezményes árakkal**.

A `sale.csv` egy lehetséges tartalma:

```
1;360;"édes";"vanilla"
1;360;"édes";"vanilla"
5;1800;"normál";"chocolate, strawberry, coconut, chocolate, chocolate"
```

25. Az elkészített feladatot dockerizálja. A Dockerfile neve legyen `Frosty.Dockerfile`, míg az image neve legyen `monogram/frosty`.
  - A `build` és `run` parancsokat egy `buildandrun.txt` fájlba írja bele!

**Ahhoz, hogy a kimenet látható legyen csatoljon fel egy mappát a container `/app/out` mappájára!**