Laboratorul 9

­­Clase Abstracte, how to ace your exam

# Takeaways



* Vom folosi cât mai simplu clasele abstracte cu așa-zisele „funcții virtuale pure”;
* Ce este pur la ele: faptul ca una dintre ele nu poate face nimic, deoarece nu are cod

# Exerciții

## 0. Clasele abstracte – exemplul cu formele de copt

În laboratorul 7 (de la 143), aveam o cerință simplu pe care este probabil s-o fi auzit și la seminar. Anume că avem mai multe tipuri de forme de copt (pătratice, dreptunghiulare, rotunde etc), și ni se cerea să aflăm arclaia totala și volumul prăjiturilor.

Am folosit moștenire (datele comune sunt înălțimea formei), dar și funcție virtuală pentru ariaBazei(), deoarece aria diferă în funcție de tipul formei.

CakeShape -> SquareShape

-> RectangleShape

-> CircleShape

In Laborator 9 -> Clase Abstracte -> Laborator7\_solved avem exercitiul din Laborator7, rezolvat cu ajutorul claselor abstracte.

**Am comentat cu // TODO Informațiile noi despre clase abstracte**

## Adăugăm moștenire multiplă claselor abstracte

Vom continua codul de la exercițiul 0

Ne dorim să formăm prăjituri care au baza rotundă și formă de cilindru de la bază până la înălțimea **height**, dupa care au formă de con de la înălțimea **height** până la înălțimea totală de **height\*2** cm.

Se cere:

* Creați o clasă ConeShape, care va fi forma de tort pentru varful prăjiturii mai speciale

!!! Ce are diferit clasă ConeShape diferit față de celelalte trei (care au forma de cub, paralelipiped, cilindru) ?

* Creăm prăjitura specială, prin moștenirea virtuală a claselor cu baza cerc, respectiv a formei ConeShape. Vom moșteni virtual, deoarece ne dorim să păstrăm o singură variabilă denumite **height** provenită din clasa de bază CakeShape.

Aceasta este moștenirea diamant. Adică

CakeShape -> ConeShape

-> CircleShape

Iar cele două clase Cone si Circle sunt ambele folosite în SpecialShape

**ConeShape** **---> \**

**|** **--->** **SpecialShape** (calculeaza si con, si cilindru)

**CircleShape** **--->**

## Tablouri în ulei pentru fiecare

**Bonus +10p**, deoarece lucram de la zero acest exercitiu.

Similar cu exercitiul 1, vom folosi functii virtuale pure.

### ***Takeaways despre metode și metode virtuale:***

**Cum ne dăm seama dacă niște date sunt atribute sau funcții?**

* **Dacă se calculeaza din alte atribute, avem de implementat o funcție.**

E.g. aria bazei nu poate să fie atribut, deoarece când modificăm o latură, ar trebui schimbată și ea. Ceea ce ar fi prea complex și fără rost.

*Menționez că există și ideea de caching. Putem avea și un atribut, și o funcție, dar adăugăm și reguli de recalculare a variabilei când se modifică atributele de care depinde. Pe scurt, prea multă bătaie de cap. Folosim mai ales când limbajul permite. Spre exemplu, Python permite să adăugăm simpla caching.*

**Când este o funcție virtuală?**

* **Când are efect diferit în funcție de derivata din care este apelată.**

E.g. ne putem gândi chiar la citirea/ scrierea virtuală

**Când este o funcție virtuală pură?**

1. **Când în clasă părinte nu putem calcula/ afișa ceea ce ne cere, deoarece nu avem toate informațiile. (pure = lacks any implementation)**

E.g. nu putem calcula salariul net, dacă nu știm rolul angajatului.

## **Cerința**

S-a deschis un magazin care automatizează crearea de tablouri în ulei. Folosește atât inteligență artificială modernă, precum [**DALL-E**](https://openai.com/blog/dall-e/)**,** dar și filtre automate, pentru a crea tablouri care imită rezultatele unor pictori celebri, precum Picasso, etc

Există trei tipuri de tablouri:

Miniaturi, cu lățimea de maxim 10cm

Tablouri, cu lățimea de maxim 57cm, dar nu mai puțin de 19.

Tablouri de perete, cu lățimea de la 1m până la 10m.

Pentru a crea mai ușor imaginile și selecta filtrele potrivite, orice tablou va reține **lățimea** și va conține **rația dintre lățime și înălțime**. Rația va fi întotdeauna între 1 și 3 pentru miniaturi, între 0.618 și 1.618 pentru tablouri, și între 1 și 10 pentru tablourile de perete.

Cerințe:

1. Implementați clase pentru toate tipurile de tablouri, dar și o clasă de bază, care va fi abstractă.
   1. În constructori veți include teste și veți arunca (throw) un întreg/ o eroare dacă nu sunt respectate condițiile.
2. Veți implementa o funcție cantitateUlei() care va calcula de cât ulei este nevoie pentru a crea tabloul, după formula

**cantitateUlei = ariaTabloului \* 0.1mL/cm3**

Metodă se va afla în clasa părinte și va folosi o funcție virtuală utilă, pentru a simplifica calcularea formulei.

Hints:

1. Fiecare tablou are lățimea salvată clar în centimetri, dar e nevoie de rație pentru a calcula lungimea. Puteți folosi întrebările de mai sus ca să vedeți cum implementați

2. Se vor implementa teste (cu try-catch preferabil, dar putem și cu if-uri unde se poate, precum la setarea rației). Cum ele diferă în funcție de tipul de tablou, este întărită utilitatea moștenirii.

## 3. Cum luăm note bune la Colocviu

*Vom folosi un colocviu simplificat, în care am micșorat numărul de clase cerute și păstrat doar cerințe esențiale.*