Laborator 8. Polimorfism. Clase abstracte

1. O firmă care vinde produse software şi-a împărţit produsele în următoarele categorii: sisteme de operare, utilitare, jocuri şi limbaje de programare. Fiecare produs are un preţ stabilit, dar luna aceasta firma oferă o reducere de 4% la toate produsele. În plus, se oferea deja o reducere de 30% la toate sistemele de operare şi o reducere de 3% la toate limbajele de programare.

Folosiţi diagrama UML de clase asociată aplicaţiei din Figura 1.

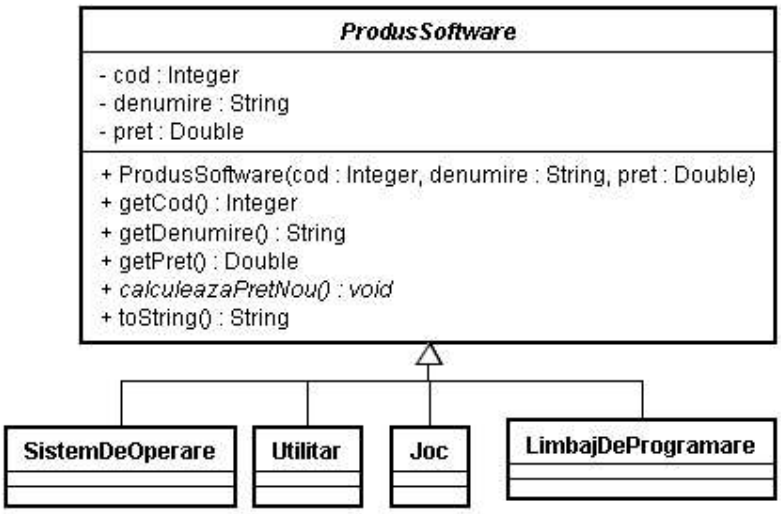


Figura 1. Diagrama UML de clase

Observăm că clasa `ProdusSoftware` este abstractă, cu metoda abstractă `calculeazăPretNou()`.

Cerinţe:

A. Implementaţi clasele din diagrama anterioară.
B. Pentru a testa clasele, scrieţi clasa `TestMagazin` ce conţine metoda `main` şi care realizează următoarele operaţii:
    a. Să se creeze un tablou de tip `ProdusSoftware` folosind datele:
        1245 WindowsNT 5000
        2134 Pascal 3500
        56890 NortonCommander 1200
        789 Mahjong 23456
    b. Să se calculeaze noul preţ al fiecărui produs din tablou.
    c. Să se afişeze informaţiile despre fiecare produs din tablou.

2. Proprietarul unui magazin de produse hardware a decis să facă o statistică despre raportul dintre preţul (în lei) a două tipuri din produsele sale (plăci video şi monitoare) şi performanţa acelui produs. În cazul plăcilor video, criteriul de performanţă este dat de numărul de puncte în 3DMark. Monitoarele sunt evaluate în funcţie de refresh-ul obţinut la o rezoluţie de 1152x864.

În funcţie de scorul său, performanţa (memorată ca un număr real) este calculată după următoarea formulă:

performanţa = ScorObţinut/ScorMaxim*100,

unde ScorMaxim reprezintă scorul maxim ce poate fi obţinut de toate produsele din aceeaşi categorie. Presupunem că punctajul maxim pe care îl pot obţine toate produsele de tip PlacaVideo este 100, iar pentru monitoare, punctajul maxim este 150.

Să se implementeze ierarhia de clase de mai jos (Figura 2), pentru a calcula şi afişa raportul pret(lei)/performantă pentru fiecare produs, în funcţie de clasa din care face parte.

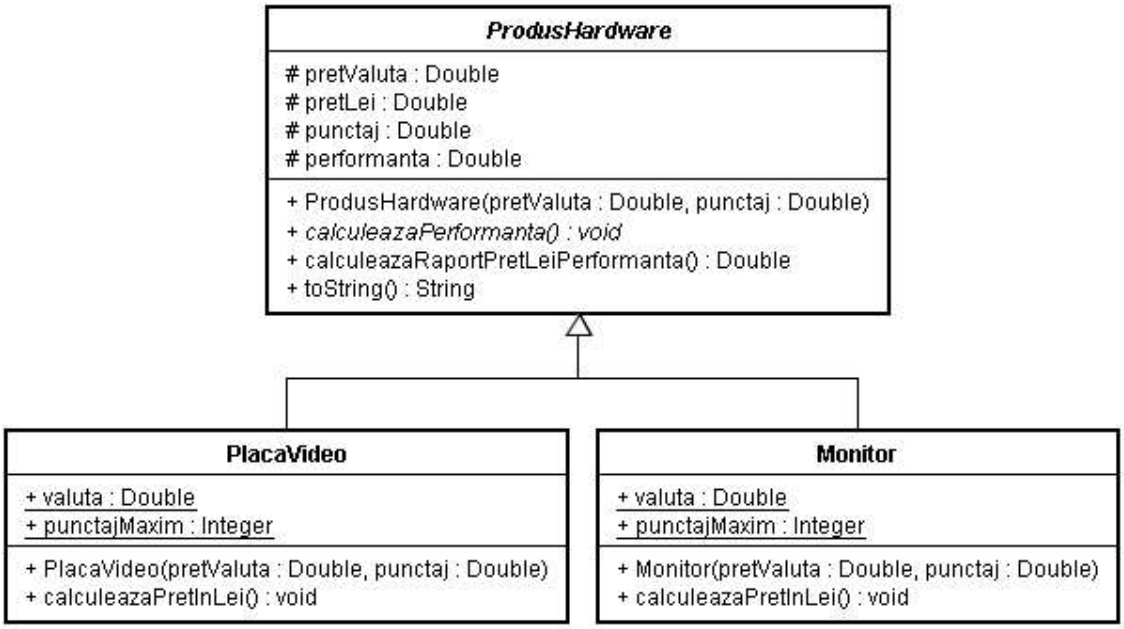


Figura 2. Diagrama UML de clase

Clasa ProdusHardware este abstractă, având metoda abstractă `calculeazaPerformanta()`. În plus, această clasă conţine variabilele protejate `pretInValuta, scor, pretLei` şi `performanta`. Valoarea variabilei `pretLei` va fi calculată în funcţie de `pretInValuta` şi tipul valutei folosite. Presupunem că preţul plăcilor video este în euro, iar cel al monitoarelor este în dolari.

Cerinţe:

A. Implementaţi clasele din diagrama anterioară.
B. Pentru a testa clasele, scrieţi clasa `TestMagazin` ce conţine metoda main şi care realizează următoarele operaţii:

   a. Crearea unui tablou de tip `ProdusHardware`, folosind datele:

PlaciVideo	Monitoare
130 76	249 78
108 77.79	224 95
152 72.55	249 96
53 54.69	279 107

   b. Calcularea performanţei fiecărui produs din tablou.

   c. Afişarea informaţiilor despre fiecare produs din tablou.

3. Fie diagrama UML de clase din Figura 3. Observăm că clasa `Forma` este rădăcina ierarhiei şi conţine metode ce vor fi redefinite în subclasele sale.

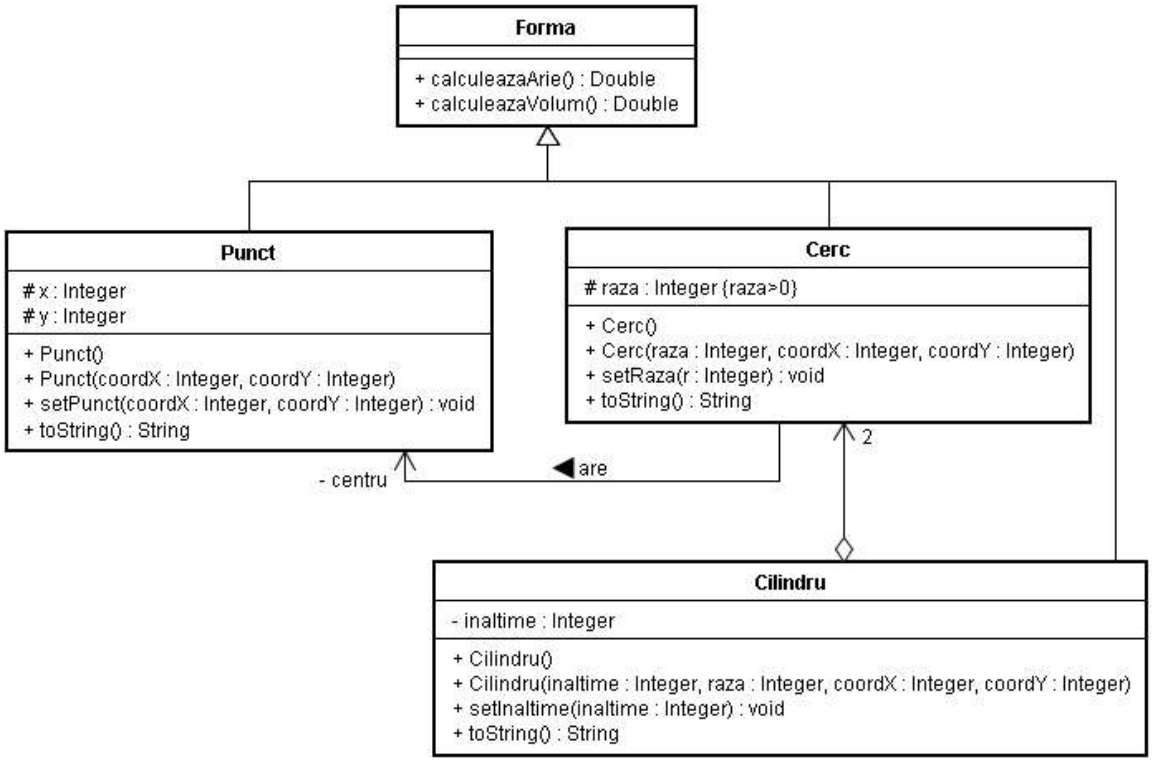


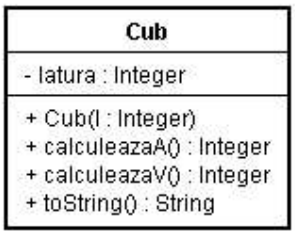
Figura 3. Diagrama UML de clase a problemei 3

Cerinţe:

a) Să se scrie o clasă `Test` care conţine un tablou de 3 forme grafice: punct, cerc şi cilindru, şi calculează, respectiv afişează, aria şi volumul formei geometrice.

b) Să se transforme clasa `Forma` într-o clasă abstractă, cu metodele abstracte `calculeazaArie()` şi `calculeazaVolum()` care vor fi implementate în toate subclasele sale.

4. Să se modifice programul de la problema anterioară astfel încât să calculeze aria şi volumul unui cub. Presupunem că avem deja clasa `Cub` cu următoarea interfaţă:



Pentru aceasta, se creează o clasă numită `CubAdaptor` care extinde clasa `Forma` şi adaptează metodele clasei `Cub` la cele date de clasa `Forma`.