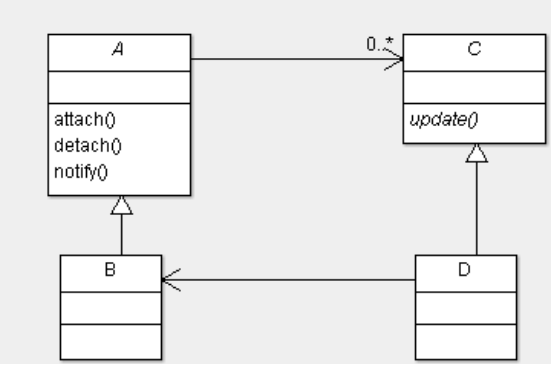
Universitatea "Al.I.Cuza" Iasi
Facultatea de Informatică

Nume:

An: | I | II | III | **Semian:** | A | B | **Grupa:** | 1 | 2 | 3 | 4 | 5 | 6 |

POO – Test 1 A - Barem
18.04.2010

Observații:
1. *Nu este permisă consultarea bibliografiei.* ***2.*** *Toate întrebările sunt obligatorii.* ***3.*** *Dacă nu este precizat altfel, fiecare întrebare este notată cu 3 puncte. Repartiția punctelor la întrebările grilă este: 1 punct alegerea corectă a variantei, 2 puncte justificarea. Alegerea corectă se punctează numai dacă justificarea este total sau parțial corectă.* ***4.*** *Nu este permisă utilizarea de foi suplimentare.*



Figură 1

1) Să se explice termenul „abstractizare" în contextul programării orientate-obiect. Se va preciza cum se implementează în C++.
Răspuns.

- o clasa este o abstractizare (0.5 +0.5 C++)
  pastreaza trasaturile (atribute, metode) esentiale
  ignora ce nu este esential
- ierarhie generalizare/specializare (0.5 +0.5 C++)
  clasa parinte este o abstractizare a claselor copii
- interfata (clasa abstracta) (0.5 +0.5 C++)

2) Să se explice componenta "View" din MVC.
Răspuns.

vizualizeaza modelul (0.75)
cere actualizari de la model (0.75)
trimite evenimentele utilizator la controller (0.75)
permite controllerului sa schimbe modul de vizualizare (0.75)

3) Să se identifice ce șablon de proiectare (design pattern) este reprezentat în Figura 1. Să se descrie succint rolul fiecărui participant.
Răspuns.
Observer**. (**1p**)**
A = Subject (0.5p)
  cunoaste observatorii (numar arbitrar)
C = Observer (0.5p)
  defineste o interfata de actualizare a obiectelor ce trebuie notificate de schimbarea subiectelor
B = ConcreteSubject (0.5p)
  memoreaza starea de interes pentru observatori
  trimite notificari observatorilor privind o schimbare
D = ConcreteObserver (0.5p)
  mentine o referinta la un obiect ConcreteSubject
  memoreaza starea care ar trebui sa fie consistenta cu subiectii

4) Să se descrie în C++ clasa D din şablonul din Figura 1 (fără definiții metode).
Răspuns.

relatia de mostenire (0.75p)
implementarea lui update() (0.75p)
relatia de asociere cu B (0.75p)
nivel de acces (0.75p)

5)

```
class Bar
{
public:
      Bar(int x = 0) { v = new int(x); }
      int getVal() { return *v; }
      void setVal(int x) { *v = x; }
private:
      int *v;
};

int main()
{
      Bar b (5);
      Bar c = b; c.setVal(6);
      cout << b.getVal() << endl;
      Bar d;
      d = b; d.setVal(7);
      cout << c.getVal() << endl;
      return 0;
}
```

Să se precizeze ce va afişa programul alăturat.
Răspuns + justificare.
iesire corecta (1p)
6
7
observa data membra dinamica (1p)
se utilizeaza constr. copiere si op.de atrib. definiti de sistem (1p)

6) Să se adauge funcțiile/operatorii la clasa Bar definită la Exercițiul 5 pentru ca programul main() să afişeze corect valorile pentru obiectele b şi c.

Răspuns.

constructor de copiere (1.5p =0.75 signatura + 0.75 impl.)
operator de atribuire (1.5p =0.75 signatura + 0.75 impl.)

7) (9 puncte) Se consideră clasele MinPri si MaxPri cu următoarele responsabilități:

- ambele memorează câte o multime de elemente;
- fiecare clasă are o operație lookup();
- lookup() din MinPri întoarce cel mai mic element şi lookup() din MaxPri întoarce cel mai mare element.

Se cere:

a) care soluție este mai bună: MaxPri moştenește MinPri sau invers? Există altă soluție mai bună decât moştenirea directă dintre cele două clase?
3 puncte din care
Niciuna sau alta solutie acceptabila (1p)
Solutia cea mai buna: considerarea unei clase abstracte *Pri* din care deriveaza MaxPri si MinPri. (2p)

b) Să se descrie în C++ cele două clase, conform răspunsului dat la a.
3 puncte (1.5 puncte fiecare clasa).

c) Să se descrie implementarea uneia dintre cele două metode lookup().
3 puncte (1 punct signatura, 1 punct algoritmul, 1punct scrierea corecta a metodei)

Răspuns .