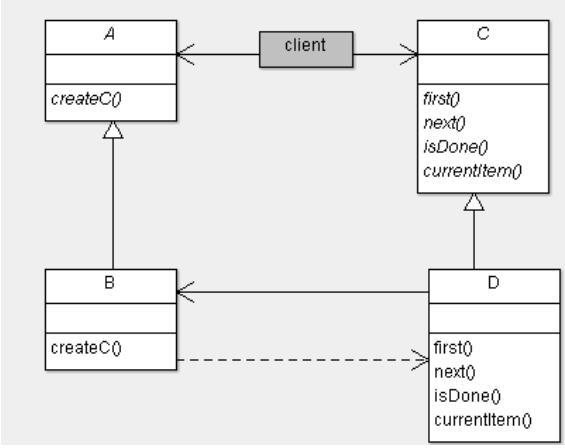
Universitatea "Al.I.Cuza" Iasi **Nume:**
Facultatea de Informatică

**An:** | I | II | III | **Semian:** | A | B | **Grupa:** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

# POO – Test scris B BAREM CORECTARE
# 07.06.2011

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




**Figură 1**

**1)** Să se identifice ce şablon de proiectare este reprezentat în Fig. 1 şi, în acest context, să se explice cum este aplicat principiul (conceptul) POO referitor la polimorfism. Se va preciza şi cum este realizat acest polimorfism în C++.

**Răspuns.**

1. Identificare șablon (iterator: A container și C iterator) 1p
2. Polimorfism realizat prin suprascriere de metode 0.5p
3. Clasele A și C joacă rol de interfață 0.5p
4. Clasele B și C trebuie să implementeze metodele din A și C respectiv 0.5p
5. Metode virtuale 0.5

**2)** Să se explice relaţiile dintre clasele din diagrama reprezentată în Fig. 1.

**Răspuns.**

1. Relația între A –client - C - asociere 1p
2. Relația dintre B și D - asociere, dependență 1p
3. Relația dintre D - C și B – A 1p

**3)** Să se descrie în C++ relaţiile dintre clasele din diagrama reprezentată în Fig. 1.

**Răspuns.**

1. Cod C++ relația între A –client - C 1p
2. Cod C++ realția dintre B și D - 1p
3. Cod C++ relația dintre D - C și B – A 1p

**4)**

```
#include <iostream>
using namespace std;
class T{
public:
      T(int d=0):data(d){ }
      operator int() const{return data;}
      T& operator+=(const T& other){
        data += other.data;
        return *this;
    }
public: int data;
```

Să se precizeze ce va afişa programul alăturat.

a) 110
b) 101
c) 111
d) 1

**Justificare.**

- Constructor implicit 0.5p
- operator+= 0.5p
- operator+ 0.5p
- Conversie 0.5p
- Răspuns corect a 1p

```
};
const T operator+(const T& t1,const T& t2){
      T rezultat(t1); rezultat += t2;
      return rezultat;
}
int main(){
    T t1(10), t2(1); int i=100;
    t2 = operator+(t1, i);
    cout << t2 << endl; return 0;
}
```

**5)**

```
#include <iostream>
#include <math.h>
using namespace std;
template <class T=float> class pereche {
public:
    pereche (T xx, T yy){x=xx; y=yy;}
    T dist () {return T(sqrt(x*x + y*y));}
private: T x, y;
};
template <> class pereche <int> {
public:
    pereche (int xx, int yy){x=xx; y=yy;}
    int dist () { return x%y; }
private: int x, y;
};
int main () {
  pereche <int> x (7,8); pereche <> y(7,8);
  cout << x.dist() << ' ';
  cout << int(y.dist()); return 0;
}
```

Să se precizeze ce va afişa programul alăturat.

**a)** 7 7
**b)** 7 10
**c)** 10 10
**d)** nimic deoarece programul are erori

**Justificare.**

- Clasa parametrizata 0.5p
- Specializare pentru int 0.5p
- Suprascriere dist() 0.5p
- Conversie de la float la int 0.5p
- Varianta corectă b 1p

**6)**

Explicaţi următorul program şi precizaţi rezultatul execuţiei?

```
#include <list>
# include <iostream>
using namespace std;
template <class T>
class my_list : public list<T>{
public:
  T operator[](int i){
    list<T>::iterator it = (*this).begin();
    for( int k = 0; k < i; ++k) ++it;
    return *it;
  }
};
int main(){
      int a[] = {1,2,3,4,5,6,7,8,9};
      const int asz = sizeof(a)/sizeof(*a);
      my_list<int> ml;
      for(int i = 0; i < asz; ++i)
            ml.push_back(a[i]);
      for(int i = 0 ; i < asz ; i+=3)
            cout << ml[i] << ' ';
      return 0;
}
```

**Răspuns.**

- my_list derivata din list 0.5p
- utilizare iterator in operator[] 0.5p
- crearea lui ml 0.5p
- afişare 0.5p
- rezultat corect 1p (**1 4 7**)

**7)**

```
#include <iostream>
using namespace std;
class Baza {
public: virtual int h() = 0;};
class Der1:public Baza {
public: int h(){cout << b <<' '; return 1;}
```

Ce va afişa programul alăturat?

**a)** 10 2 11 2
**b)** 10 1 11 1
**c)** 10 1 10 1
**d)** 10 2 10 2

**Justificare.**

- ierarhia de clasa 0.5p

```
protected:  static int b;
};
class Der2:public Der1 {
public:Der2(){b++;}
      int f(){ cout << b << ' '; return 2 }
};
int Der1::b = 10;
int main(){
      Baza* x = new Der1;
      cout << x->h() << " ";
      Der1* y = new Der2;
      cout << y->h() << " ";
      delete x;delete y;return 0;
}
```

- funcția h virtuală 0.5p
- x și y instanțe 0.5p
- apel corect h() 0.5p
- rezultat corect b) 1p

**8)** (9 puncte) Un *monom* este format din *coeficient* şi *puterea* variabilei *X* (se consideră monoame cu o singură variabilă). *Valoarea* unui monom într-un punct *x* (număr întreg) este dată de produsul dintre coefient şi puterea corespunzătoare a lui *x*. Un *polinom* este reprezentat de o listă de monoame. Un polinom este dinamic, în sensul că se poate *adăuga* sau *şterge* un monom. Valoarea unui polinom este dată de suma valorilor monoamelor componente calculate în *x*.

- Să se descrie complet în C++ clasa *Monom* corespunzătoare monoamelor.
- Să se descrie complet în C++ clasa *Polinom* corespunzătoare polinoamelor. Se va utiliza un container din STL pentru a reprezenta relaţia dintre clasa *Polinom* şi *Monom*.
- Să se descrie o secvenţă de program C++ care să creeze polinomul $12 + 23X + 34X^5 + 45X^9$.

**Răspuns .**

- clasa *Monom*:
  - constructor 0.5
  - metoda *valoare()* 1p
  - atributele *coeficient, putere* 0.5p
- clasa *Polinom*:
  - constructor 0.5
  - metoda *valoare()* 1p
  - adăugare 1p
  - ştergere 1p
  - atribut *monoame* 0.5p
- utilizarea claselor pt. crearea polinomului 2p
- stil de scriere 1p