

Examen de licență 2018 - Informatică

Exemple de întrebări - Limbaje de programare și inginerie software

In atenția studenților:

Proba scrisă a examenului de licență din sesiunile iulie-septembrie 2018 va consta din 60 de întrebări similare, ca structură și nivel de dificultate, celor din această culegere. Pentru fiecare dintre cele trei categorii (Structuri discrete și algoritmi, Limbaje de programare și inginerie software, Sisteme de calcul și baze de date) vor fi câte 20 de întrebări.

Pentru neclarități privind enunțurile sau răspunsurile puteți să vă adresați celor care au propus întrebările pentru fiecare secțiune.

Limbajul C:

- Victoria Iordan (victoria.iordan@e-uvv.ro)
- Cosmin Bonchiș (cosmin.bonchis@e-uvv.ro)

Limbajul C++:

- Daniel Pop (daniel.pop@e-uvv.ro)
- Flavia Micotă (flavia.micota@e-uvv.ro)

Limbajul Java:

- Victoria Iordan (victoria.iordan@e-uvv.ro)
- Flavia Micotă (flavia.micota@e-uvv.ro)

Inginerie software:

- Cristina Mîndruță (cristina.mindruta@e-uvv.ro)

1 Limbajul C

1. Care din următoarele acțiuni au loc la compilare?

- (a) analiza lexicală;
- (b) includerea fișierelor header;
- (c) definirea constantelor simbolice;
- (d) analiza sintactică;
- (e) generarea codului obiect;
- (f) editarea de legături

2. În condițiile

```
#define swap(a,b) {int aux; aux=a; a=b; b=aux;}  
float x=10.5, y=3.75;
```

în urma apelului `swap(x, y)`; valorile lui `x`, respectiv `y` vor fi:

- (a) `x=3.75, y=10.5`;
- (b) `x=3.0, y=10.5`;
- (c) `x=3.75, y=10.0`;
- (d) `x=3.0, y=10.0`;

3. Care din variante definește corect și complet ce anume se specifică prin tipul unei date?

- (a) Numărul de octeți ocupați
- (b) Spațiul necesar reprezentării și modul de reprezentare;
- (c) Operațiile permise;

4. Ce reprezintă domeniul unei variabile:

- (a) plaja de valori pe care le poate lua;
- (b) locul unde se creează;
- (c) locul din textul sursă unde poate fi folosită;
- (d) dacă are semn sau nu;

5. Domeniul identificatorilor de variabile globale ține din locul declarației până la:

- (a) sfârșitul blocului;
- (b) sfârșitul programului;
- (c) sfârșitul textului sursă;
- (d) sfârșitul funcției;

6. La execuția programului următor se tastează 20. Cese va afișa după execuție?

```
#include <stdio.h>
void main() {
    char a;
    scanf("%c",&a);
    printf("%c",a); }
```

- (a) 20;
 - (b) 2;
 - (c) 0;
7. Care este durata de viață a variabilelor locale?
- (a) cât timp sunt utilizate;
 - (b) câteva minute;
 - (c) cât durează execuția subprogramului;
 - (d) depinde de clasa lor de memorare;
8. Când au loc conversii implicite de tip?
- (a) la citirea datelor de intrare și la afișarea rezultatelor;
 - (b) când operandii unei expresii au tipuri diferite;
 - (c) când argumentul actual cu care se apelează o funcție are alt tip decât argumentul formal corespunzător;
9. Date declarațiile
- ```
int n=10, m=4;
float x;
```
- valoarea expresiei  $x = 1.5 + n/m$  este
- (a) 4.0;
  - (b) 3.5;
  - (c) de tip double;
  - (d) de tip float;
10. Dacă a este o variabilă întreagă, care este valoarea expresiei
- ```
(a < 'a') && (a > 'z')
```
- (a) 1;

- (b) 0;
- (c) depinde de valoarea lui a

11. Date declarațiile:

```
static int i, t[10];
```

și presupunând că atât i, cât și t nu sunt inițializați explicit, valoarea expresiei

```
( i==0 ) || ( t[i]<0 )
```

- (a) 1;
 - (b) 0;
 - (c) depinde de context.
12. Care este rezultatul numeric al evaluării expresiei

```
a < b < c
```

daca a=-2, b=-1 si c=0?

- (a) 1;
 - (b) TRUE;
 - (c) 0;
 - (d) FALSE;
13. In condițiile în care c este declarată

```
char c;
```

valoarea expresiei

```
c = getchar() != EOF
```

- (a) este valoarea returnată de funcția getchar;
 - (b) este 0;
 - (c) este 1;
 - (d) este 0 sau 1;
14. Ce se va afișa în urma execuției secvenței următoare:

```
#include <stdio.h>
```

```
void main() {
```

```
    printf("%d",10>20 )
```

- (a) false;

- (b) error;
(c) 0;
(d) 1;
15. Ce se va afișa în urma execuției secvenței următoare:
- ```
#include <stdio.h>
void main() {
 unsigned char x=25;
 x=x<<2;
 printf("%d", (int)x); }
```
- (a) 27;  
(b) 100;  
(c) 23;
16. Fie declarațiile:
- ```
int x=10, y=20;
```
- Care este valoarea expresiei !x-y?
- (a) 0;
(b) -10;
(c) -20;
17. Fie declarațiile:
- ```
int x=-1, y=1;
```
- Care este valoarea expresiei (x&& y)==!(|x||y)?
- (a) 0;  
(b) 1;  
(c) -1;  
(d) 2;
18. Care dintre variabilele din secvența următoare are valoarea finală -1?
- ```
int x=-2,y=0,z=1,v=-1;
x++; y-=2; z-=y+++2; v='a'-'b';
```
- (a) toate;

- (b) x;
(c) y;
(d) x,v;
(e) x,y,v;
(f) y,v;
19. Care dintre următoarele expresii condiționale este o transcriere corectă a textului: dacă x este pozitiv, atunci y ia valoarea lui a, altfel y ia valoarea lui b?
- (a) $y=(x>0)?a:b$
(b) $y=!x>0?b:a$
(c) $!(x>0)?y=a:y=b$
(d) $x>0?y=a:y=b$
20. Ce afișează programul?
- ```
#include <stdio.h>
void main() {
 int x=5,y=2,z=3;
 printf("%d", (z=x-2,x=y/z,y-=x,x/2));}
```
- (a) 3 0 2 0  
(b) 1 0 0 2  
(c) 0  
(d) 3 2.5 -0.5 1.75  
(e) programul este eronat
21. Ce afișează programul?
- ```
#include <stdio.h>
void main() {
    int x,y,z;
    x=y=z=4;
    printf("%d", (x<<z)-(x|y)+(z&y));}
```
- (a) 64
(b) 4
(c) 0
(d) 5

- (e) 33
(f) 32
22. Care din următoarele variante reprezintă modalități de comunicare între funcții?
- (a) apel;
 - (b) prin variabile locale;
 - (c) prin argumentele actuale;
 - (d) prin valoarea returnată;
 - (e) prin variabile globale
 - (f) prin includere
23. Ce este contextul de apel al unei funcții?
- (a) lista argumentelor formale
 - (b) o zona de memorie (de pe stivă)
 - (c) locul din textul sursă în care se apelează
 - (d) instrucțiunile (definiția funcției)
24. Care este efectul secvenței:
- ```
for(i=0; i<N; i++);
 printf("\n %d",i);
```
- (a) se afișează pe linii separate valorile de la 1 la N
  - (b) se afișează pe linie nouă valoarea N-1
  - (c) se afișează pe linii separate valorile de la 0 la N-1
  - (d) se afișează pe linie nouă valoarea N
25. Ce afișează următorul program?
- ```
#include <stdio.h>  
int x;  
void y(int z) {printf("%d",++z);}  
void main() {x=1; y(x); printf("%d",x);}
```
- (a) 2 1
 - (b) 1 1
 - (c) 21
 - (d) 11

- (e) 22
(f) 2 2
26. Ce afișează programul următor:
- ```
#include <stdio.h>
int a; int u(){ int a=2; return a; }
void w(int a){ printf("%d",++a); }
void v(){ int a=4; printf("%d",a); }
void main()
 {a=1;
 printf("%d",a); v(); w(a);
 printf("%d%d", u(),a); }
```
- (a) 14221  
(b) 11111  
(c) 14121  
(d) 14212
27. Care este valoarea lui n după execuția secvenței:
- ```
char t[ ]="timisoara", *p,*q,n;
p=q=t;
while(*(q++));
n=q-p;
```
- (a) n=0
(b) n=9
(c) n=10
(d) n='\0'- 't'
28. Ce reprezintă declarația: `int *(*f)(int *)` ?
- (i) funcție ce primește argument pointer la întreg și întoarce pointer la întreg
(ii) o declarație greșită
(iii) pointer către o funcție care așteaptă ca argument un pointer la int și întoarce un pointer la int
(iv) pointer către o funcție care întoarce un int

- (a) i
- (b) ii
- (c) iii
- (d) iv
- (e) nici una
- (f) toate
- (g) i și iv

29. Ce se afișează?

```
#include <stdio.h>
void main() {
    char *u[2]={ "abc", "def" }, **v;
    v=&u[0];
    printf("%c",(*v)[1]); }
```

- (a) a
- (b) b
- (c) c
- (d) d
- (e) e
- (f) f
- (g) abc

30. Care din afirmațiile următoare, referitoare la structuri, sunt adevărate:

- (a) sunt tipuri de date agregate
- (b) sunt tipuri de date definite de programator
- (c) sunt tipuri de date predefinite
- (d) sunt tipuri de date scalare

2 Limbajul C++

1. Un constructor se caracterizează prin următoarele proprietăți în C++:
 - (a) Este o funcție membră care are același nume ca și clasa în care este declarată
 - (b) Este o funcție membră care întoarce o valoare
 - (c) Este o funcție membră care nu are valoare de return
 - (d) Este o funcție membră care întotdeauna nu are parametri
 - (e) Este o funcție membră utilizată pentru a inițializa un obiect
 - (f) Este o funcție membră utilizată pentru a dealoca spațiu de memorie
 - (g) Este o funcție membră care se folosește împreună cu operatorul new
 - (h) O clasă nu poate avea mai mult de un constructor
 - (i) Este o metodă care se folosește împreună cu operatorul delete
 - (j) Poate fi o funcție membră virtuală
2. Care dintre următoarele facilități sunt suportate în limbajul C++ dar nu sunt suportate în limbajul C?
 - (a) Funcții care au valori implicite ale parametrilor
 - (b) Macro definiții
 - (c) Tipul de dată referință
 - (d) Operatorul de rezoluție
 - (e) Supraîncărcarea funcțiilor
 - (f) Cuvântul cheie const
3. Dacă *a* și *b* sunt două obiecte de tipul clasei *Text* și clasa conține o metodă cu următorul prototip:
static bool equals(const Test&, const Test&);? Care este apelul corect pentru funcția *equal()*?
 - (a) equals(a,b);
 - (b) a.equals(b);
 - (c) Test.equals(a,b);
 - (d) a.equals(b,a);
 - (e) equals(b);
4. Care dintre următoarele afirmații sunt adevărate?
 - (a) Operatorii pot fi supraîncărcați în limbajul C++.
 - (b) Limbajul Java rulează într-o mașină virtuală.
 - (c) Limbajele Java și C++ sunt independente de platformă.

- (d) Variabile referință sunt caracteristice pentru limbajele C++ și Java.
- (e) Limbajul C++ nu are implementat un mecanism de garbage collection.
- (f) Limbajul C++ nu permite moștenire multiplă.
5. Care dintre următoarele afirmații sunt false?
- (a) În limbajele Java și C++ nu contează ordinea în care sunt definite și apelate metodele.
- (b) În limbajele Java și C++ specificarea comentariilor se realizează la fel.
- (c) În cazul tratării excepțiilor prin mecanismul try ... catch în limbajele Java și C++ există o clauză care se execută în orice caz (chiar dacă s-a aruncat sau nu o excepție).
- (d) În limbajele Java și C++ există o clasă care este supraclasă pentru toate clasele.
- (e) Declararea claselor în Java și C++ trebuie să se termine cu punct și virgulă (;).
- (f) În limbajul C++ nu toate metodele trebuie definite în interiorul clasei.
6. Care dintre următoarele afirmații sunt adevărate despre template-uri în C++?
- (a) Template-urile sunt o facilitare a limbajului C++ care permit folosirea același cod pentru diferite tipuri de date
- (b) Permit scrierea unei funcții care se folosește pentru toate tipurile de date inclusiv pentru tipurile definite de utilizator. Ca de exemplu sort(), max(), min(), etc.
- (c) Permit scrierea unei clase sau structurii care poate fi folosită pentru orice tip de date inclusiv tipuri de date definite de utilizator. De exemplu: liste dublu înlănțuite, stive, cozi ...
- (d) Template-urile sunt exemple de polimorfism la execuție.
7. Care este rezultatul următorului program:
- ```
#include <iostream>
using namespace std;
class Test {
 int x;
 Test() { x = 5;}
};
int main() {
 Test *t = new Test;
 cout << t->x;
}
```
- (a) Eroare de compilare
- (b) 5
- (c) O valoare aleatoare
- (d) 0
8. Dacă o clasă X are ca membri variabile pointer este recomandat să conțină:
- (a) Doi constructori
- (b) Un destructor care nu face nimic `X(){}`
- (c) O funcție friend care să permită copierea obiectului
- (d) Supraîncărcarea operatorului =

- (e) Constructorul de copiere
- (f) Supraîncărcarea operatorului ==
- (g) Un destructor în care se dealocă memoria adresată de membrii de tip pointer ai clasei
9. Care este rezultatul execuției următoarei secvențe de cod:
- ```
#include <iostream>
using namespace std;

template <typename T>
    T max(T x, T y) {
        return (x > y)? x : y;
    }
int main() { {
    cout << max(3, 7) << std::endl;
    cout << max(3.0, 7.0) << std::endl;
    cout << max(3, 7.0) << std::endl;
    return 0;
} }
```
- (a) 7
- 7.0
- 7.0
- (b) Eroare de compilare la toate liniile care conțin streamul de ieșire cout spunând că tipul datelor nu este specificat
- (c) Eroare de compilare la ultima linie care conțin streamul de ieșire cout spunând că apelul funcției max este ambiguu.
- (d) Nici una din afirmațiile de mai sus.
10. Care dintre următoarele recomandări sunt bine de urmat pentru scrierea unei aplicații:
- (a) Scrierea de funcții care conțin multe linii de cod
- (b) Menținerea codului simplu și eliminarea complexității necesare
- (c) Codul trebuie scris astfel încât să fie logic și ușor de înțeles de oricine
- (d) Scrierea de comentarii pentru fiecare linie de cod
- (e) Evitarea codului care se repetă și extragerea lui în clase, funcții
- (f) Separarea codului în module, fiecare modul concentrându-se pe o anumită cerință
- (g) Crearea de interfețe generale care servesc tuturor cerințelor și neîmpărțirea lor în interfețe mai mici care deservește cerințe specifice
11. Pentru următoarele clase care este ordinea corectă de apelare a constructorilor și destructorilor:
- ```
class B { public: B(){} };
class M : public B { public: M(){} };
class N: protected B { public: N(){} };
class D: public N, protected M { public: D() : M(), N() {} };

int main() {
 D obj;
 return 0;
} }
```

- (a) Constructor: B N B M D  
Destructor: B N B M D
- (b) Constructor: B M B N D  
Destructor: D N B M B
- (c) Constructor: B N B M D  
Destructor: D M B N B
- (d) Constructor: D  
Destructor: D
- (e) Constructor: B N M D  
Destructor: D M N B
- (f) Constructor: B N B M D  
Destructor: D N B M B
12. Care dintre următoarele afirmații sunt adevărate?
- ```

1. #include <iostream>
2. #include <vector>
3. #include <algorithm>
4. using namespace std;
5. void myfct(int i){
6.     cout << " " << i*10;
7. }
8. int main() {
9.     vector<int> v;
10.    v.push_back(10);
11.    v.push_back(20);
12.    v.push_back(30);
13.    vector<int>::iterator it;
14.    for_each(v.begin(), v.end(),myfct);
15.    it=find(v.begin(), v.end(), 20);
16.    cout <<endl << *++it << endl;
17.    cout <<binary_search(v.begin(), v.end(), 20);
18.    return 0;
19. }
```
- (a) După executarea liniei 14 se va afișa 10 20 30
- (b) După executarea liniei 14 se va afișa 100 200 300
- (c) După executarea liniei 14 se va afișa 0 0 0
- (d) După executarea liniei 16 se va afișa 20
- (e) După executarea liniei 16 se va afișa 30
- (f) După executarea liniei 17 se va afișa 2
- (g) După executarea liniei 17 se va afișa 1
- (h) După executarea liniei 17 se va afișa 20
13. Care este rezultatul următorului program în C++?

```

#include<iostream>
class Baza {};
class Derivata: public Baza {};
int main() {
    Derivata d;
    try {
        throw d;
    } catch(Baza b) {
        std::cout<<"Am prins o exceptie de tipul suprac clasei";
    } catch(Derivata d) {
        std::cout<<"Am prins o exceptie de tipul subclasei";
    } catch(...) {
        std::cout<<"Alta exceptie";
    } return 0;
}

```

- (a) Am prins o exceptie de tipul subclasei
- (b) Am prins o exceptie de tipul suprac clasei
- (c) Eroare de compilare
- (d) Alta exceptie

14. Care este rezultatul următorului program?

```

#include <iostream>
class Forma {
public:
    virtual Forma* duplica() {
        return new Forma;
    }
    virtual void afisare() {
        std::cout << "Forma" << std::endl;
    }
};
class Patrat : public Forma {
public:
    virtual Patrat* duplica() {
        return new Patrat;
    }
    virtual void afisare() {
        std::cout << "Patrat" << std::endl;
    }
};
int main(int argc, char** argv) {
    Forma* s1 = new Patrat;
    Patrat* b1 = s1->duplica();
    b1->afisare();
    delete s1; delete b1;
    return 0;
}

```

- (a) Patrat
- (b) Forma
- (c) Codul nu se compilează
- (d) Programul arunca o excepție
- (e) FormaPatrat
- (f) PatratForma

15. Rezultatul executării următoarei secvențe de cod este:

```
class X {
public:
    X(int i=0) {
        p=new int;
        if(p) *p=i;
    }
    X(const X &r) {
        p=r.p;
    }
    ~X(){
        if(p)
            delete p;
    }
private:
    int *p;
};
void main() {
    X *o1=new X(1), *o2=new X(*o1);
    delete o1;
    delete o2;
}
```

- (a) Codul nu se compilează.
- (b) Eroare la runtime deoarece o locație de memorie este eliberată de mai multe ori.
- (c) Eroare la runtime deoarece se eliberează o locație de memorie pentru care nu a fost alocată memorie prin program.
- (d) Totul este corect și codul se compilează și rulează fără probleme.

16. Care este rezultatul executării următoarei secvențe de cod?

```
class B {
public:
    B() { cout<<"B::B()"<<endl; }
    B(const B& r) { cout << "B::B(B&)"<<endl; }
};
class A {
public:
    A(const A& r) { cout<<"A::A(A&)"<<endl; }
    A(B& bb) { cout<<"A::A(B&)"<<endl; }
};
void f(A) {
    // ...
}
int main(int, char*[]) {
    B b;
    f(b);
}
```

- | | |
|------------------------------------|------------------------------------|
| (a) B::B()
A::A(A&)
A::A(A&) | (b) B::B()
A::A(B&)
A::A(B&) |
| (c) B::B()
A::A(B&) | (d) B::B()
B::B(B&)
A::A(A&) |
| (e) B::B()
B::B(B&)
A::A(B&) | |

17. Care dintre liniile următoare apelează constructorul clasei de bază Angajat în C++?

- (a) Angajat::Angajat(const char* s, int d) : Angajat(s,d) {}
 (b) Angajat::Angajat(const char* s, int d) { Angajat(s,d); }
 (c) Angajat::Angajat(const char* s, int d) { super(s,d); }?

18. Ce va afișa următorul program?

```
#include <iostream>

template<class T> class Foo {
public:
    int B(T t) { return 1; }
    int B(int i) { return 2; }
    int B(int i) const { return 3; }
};

template<> class Foo<int> {
public:
    int B(int i) { return 4; }
    int B(int i) const { return 5; }
};

int doB() {
    Foo<int> g;
    const Foo<int> f=g;
    return f.B(10); // 6
}

int main() {
    std::cout << doB();
    return 0;
}
```


- (a) 1
 - (b) 2
 - (c) 3
 - (d) 4
 - (e) 5
 - (f) Nu se compilează
 - (g) Nu afișează nimic
19. Dacă B este clasă de bază publică pentru clasa A, care dintre următoarele două declarații sunt corecte: (1) `A *p = new B;` sau (2) `B *p = new A;`?
- (a) ambele, deoarece pointerii la clase de bază/derivate pot fi inițializați în ambele moduri
 - (b) niciuna, deoarece tipul pointerului nu corespunde cu tipul obiectului spre care pointează acesta
 - (c) doar (1)
 - (d) doar (2)
20. Care dintre următoarele afirmații sunt adevărate în limbajul C++?
- (a) O funcție statică nu poate arunca o excepție.
 - (b) O funcție statică nu poate returna o variabilă membru non-statică a clasei.
 - (c) O funcție statică nu poate returna o variabilă membru statică a clasei.
 - (d) O variabilă membru statică a clasei nu poate fi modificată într-o funcție membră constantă non-statică.
21. Fie declarația de mai jos. Cum poate fi accesat membrul Counter fără a crea o instanță a clasei Foo?
- ```
struct Foo {
 static int Counter;
 Foo(const Foo &f);
 Foo();
};
```
- (a) `Foo().Counter`
  - (b) `Foo.Counter`
  - (c) `Foo-> Counter`
  - (d) `Foo::Counter`
  - (e) `::Counter`
22. Considerând declarația clasei Foo și funcția foo definită mai jos, care dintre următoarele afirmații este adevărată?

```

struct Foo {
 static int Counter;
 Foo(const Foo &f);
 Foo();
};

void foo() {
 Foo f;
 Foo f2 = f;
}

```

- (a) Ar fi fost invocat constructorul de copiere doar dacă f (adica valoarea din dreapta) ar fi un obiect constant.
- (b) Un operator de asignare este generat implicit și este folosit.
- (c) Este invocat constructorul de copiere la inițializarea obiectului f2.

23. Atunci când funcțiile membre ale unei clase nu au sens în contextul unei clase derivate, cauza cea mai probabilă este violarea următorului principiu al OOP:

- (a) OCP
- (b) SRP
- (c) LSP
- (d) DRY

24. Ce va afișa secvența de cod de mai jos?

```

void f(int a) {
 static int x = 100;
 cout << x << " ";
 if(a>0) {
 static int y = 5;
 cout << y+x << " ";
 }
 x = 200;
 cout << x << " ";
}

void main(int, char*[]) {
 f(0);
 f(1);
}

```

- (a) 100 200 200 205 200
- (b) 100 200 100 105 200
- (c) 100 200 100 200
- (d) 100 200 200 105 200

25. Fie clasa Angajat o clasă de bază pentru mai multe subtipururi de angajați ai unei instituții. Fiind dată funcția de mai jos, în linia etichetată cu (1):

```

void f(vector<Angajat*> v) {
 for(int i=0; i<v.size(); i+)
 v[i]->print(); (1)
}
// vector - reprezinta clasa
// din biblioteca standard C++

```

- (a) Se va apela întotdeauna corect funcția print() din clasa derivată
- (b) Se va apela corect funcția print() din clasa derivată doar dacă funcția print() este virtuală în clasa Angajat
- (c) Se va apela întotdeauna funcția print() din clasa Angajat deoarece obiectul pentru care se apelează este de tipul Angajat\*
- (d) Codul nu se va compila deoarece operatorul [] nu este redefinit pentru clasa vector

26. Excepțiile sunt:

- (a) erori care apar la compilarea programului
- (b) situații speciale tratate în program prin teste de tipul if(variabila == NULL)
- (c) erori care apar la rularea programului
- (d) 'aruncate' folosind instrucțiunea try și 'tratate' folosind instrucțiunea catch
- (e) 'aruncate' folosind instrucțiunea throw și 'tratate' folosind instrucțiunile try si catch

27. Principiul OCP se referă la:

- (a) responsabilitățile pe care trebuie să le implementeze o clasă
- (b) realizarea de ierarhii de clase consistente
- (c) problemele care apar din cauza codului duplicat
- (d) posibilitatea extinderii claselor și evitarea modificărilor claselor și codului existent

28. Care dintre următoarele afirmații sunt adevărate?

- (a) În cazul claselor șablon, compilatorul va genera cod pentru toate șabloanele declarate, indiferent dacă sunt instanțiate cu un tip concret sau nu.
- (b) În cazul funcțiilor șablon, se va genera cod doar în cazul apelurilor realizate.
- (c) Erorile în declarațiile șabloanelor pot fi descoperite toate la compilare.
- (d) Pot exista erori în declarațiile șabloanelor care sunt descoperite abia la instanțierea șablonului.
- (e) Un parametru non-type al unui șablon, de ex. `template < int i > class X {}`, poate fi instanțiat doar cu valori sau expresii constante de acel tip (in exemplu, int).

29. O clasă de bază virtuală

- (a) este inițializată o singură dată în cazul unei relații de moștenire multiplă de tip diamant.
- (b) are toate funcțiile membre virtuale

- (c) are toate funcțiile membre virtuale pure
- (d) este o clasa abstractă din care este derivată o clasa concretă.

30. Polimorfismul

- (a) se referă la mecanismul de alocare/dealocare de memorie pentru obiecte în limbajele orientate obiect.
- (b) se referă la invocarea metodelor folosind mecanismul de legare dinamică la rularea programului
- (c) se obține în C++ prin intermediul funcțiilor membre virtuale
- (d) este posibilitatea să avem mai multe funcții cu același nume în cadrul unei clase.

### 3 Limbajul Java

1. Se consideră următoarele definiții în limbajul Java:

```
public interface I{
 int counter = 0;
}
public class A implements I{
 public A(){
 counter++;
 }
}
```

Care dintre următoarele afirmații este adevărată?

- (a) Valoarea variabilei counter este 0 după crearea unui obiect de tipul A;
  - (b) Valoarea variabilei counter este 1 după crearea unui obiect de tipul A;
  - (c) Definiția interfeței este incorectă pentru că interfețele nu pot avea atribute;
  - (d) Clasa A nu se compilează deoarece membrul counter nu poate fi modificat;
  - (e) Interfața I nu se compilează deoarece nu poate defini atribute ci doar operații.
2. Se consideră următoarele definiții în limbajul Java:

```
public interface I1{
 int method();
 public void method1();
}
public interface I2{
 public int method();
 public void method2();
}
public abstract class C implements I1, I2{
 public int method() {
 return 0;
 }
 public void method1(){};
}
```

Care dintre următoarele afirmații este adevărată?

- (a) Clasa C se compilează
- (b) Clasa C nu se compilează deoarece nu implementează complet interfața I1;
- (c) Clasa C nu se compilează deoarece nu implementează complet interfața I2;
- (d) Nu se pot crea obiecte de tipul C;
- (e) Clasa C nu se compilează deoarece nu are metode abstracte.

3. Se consideră următorul cod Java:

```
public interface I{
}
public class C0{
 public void m(){
}
public class C1 extends C0 implements I{
 public void m(){
 System.out.println("m called");
 }
}
public class C2 extends C0 implements I{
}
public class C{
 private C0[] array = new C0[]{new C1(), new C2()};
 public void m(){
 array[0].m();
 array[1].m();
 }
}
```

Care dintre următoarele afirmații este falsă?

- (a) Clasa C nu se compilează;
- (b) Clasa C2 nu se compilează deoarece nu are metode;
- (c) Clasa C beneficiază de polimorfism deoarece atât C1 cât și C2 extind C0;
- (d) Clasa C beneficiază de polimorfism deoarece atât C1 cât și C2 implementează I;
- (e) Interfața I nu se compilează deoarece nu are operații definite.

4. Un obiect este ..... a unei clase.
- (a) o subclasă
  - (b) o instanță
  - (c) o interfață
  - (d) o încapsulare
  - (e) o funcție membră
5. Se dau trei clase: Computer, AppleComputer, și IBMComputer. Care sunt relațiile dintre aceste clase?
- (a) Computer este clasă de bază, AppleComputer și IBMComputer sunt subclase ale clasei Computer.
  - (b) IBMComputer este clasă de bază, AppleComputer și Computer sunt subclase ale clasei IBMComputer.
  - (c) Computer, AppleComputer și IBMComputer nu sunt relaționate.
  - (d) Computer este clasă de bază, AppleComputer este o subclasă a clasei Computer, și IBMComputer este o subclasă a AppleComputer

6. Se consideră următoarea definiție a unei clase Java:

```
abstract class C{
 abstract void m0();
 final public abstract void m1();
}
```

Care dintre următoarele afirmații este adevărată?

- (a) Clasa nu se compilează deoarece nu are metode care nu sunt abstracte;
  - (b) Clasa se va compila, dar metoda m1 nu poate fi suprascrisă în subclase;
  - (c) Clasa nu se va compila din cauza modificatorilor de vizibilitate ai metodei m1;
  - (d) Clasa nu se va compila deoarece o clasă abstractă trebuie să fie declarată public;
  - (e) Orice clasă ne-abstractă care extinde această clasă trebuie să implementeze cele două metode din clasa C.
7. Se consideră următoarea definiție a unei clase Java:

```
public abstract class C{
 abstract void m0();
 protected abstract void m1();
}
```

Care dintre următoarele afirmații sunt adevărate?

- (a) Vizibilitatea metodei m1 este public deoarece clasa are vizibilitate public;
- (b) Clasele din același pachet pot apela metoda m1;
- (c) Clasele din același pachet și din pachetele părinte pot apela metoda m1;
- (d) Sub-clasele clasei C pot apela metoda m1;
- (e) Sub-clasele clasei C trebuie să implementeze metoda m1.

8. Se consideră codul Java de mai jos:

```
public class C{
 private final static int MAX_LENGTH = 10;
 private Object[] numbers;
 public C(Number[] otherNumbers){
 numbers = otherNumbers;
 }
 public C(Object[] otherNumbers){
 numbers = otherNumbers;
 }
 public double computeSum(){
 int result = 0;
 for (int i = 0; i < MAX_LENGTH; i++){
 result += (Integer)numbers[i];
 }
 return result;
 }
}
```

Care dintre următoarele afirmații este adevărată?

- (a) Primul constructor aruncă o excepție `ClassCastException` la asignare;
  - (b) Metoda `computeSum` poate arunca excepția `ArrayIndexOutOfBoundsException`;
  - (c) Constructorii aruncă excepția `NullPointerException` dacă argumentul este null;
  - (d) Metoda `computeSum` poate arunca excepția `NullPointerException`.
9. Dacă o metodă are ca paramentru o referință la un obiect inmutabil, de exemplu clasa `String`, poate metoda să modifice starea obiectului?
- (a) Da, dacă avem o referință la un obiect inmutabil îi putem modifica starea



- (b) Da, dar trebuie să folosim un operator special "+" ca să realizăm acest lucru
  - (c) Nu, starea obiectelor imulabile nu poate fi modificată de nimeni după ce obiectul a fost creat
  - (d) Nu, doar proprietarul obiectului imutabil poate să îi schimbe starea.
10. Care dintre membrii unei clase nu sunt serializabili în Java?
- (a) variabile membre statice
  - (b) variabile membre transiente
  - (c) variabilele membre publice
  - (d) nici una din variante
11. Se consideră următorul cod Java (compatibil Java 7):

```
public long processList1(List<Integer> list){
 long sum = 0;
 for (Integer item : list){
 sum += item;
 list.remove(item);
 }
 return sum;
}

public long processList2(List<Integer> list){
 long sum = 0;
 Iterator<Integer> it = list.iterator();
 while (it.hasNext()){
 sum += it.next();
 it.remove();
 }
 return sum;
}
```

Care dintre afirmațiile următoare sunt adevărate în relație cu limbajul Java într-un context "single-thread"?

- (a) Ambele metode sunt implementate corect și calculează aceeași sumă pentru același argument;
- (b) Metoda processList2 nu se va compila;
- (c) Metoda processList1 poate arunca excepția ConcurrentModificationException;

- (d) Metoda `processList2` poate arunca excepția `ConcurrentModificationException`;
  - (e) Ambele metode golesc lista primită ca argument, ca și un efect colateral.
12. De ce se setează opțiunea de auto commit, de execuție a interogărilor, pe false în cazul unei aplicații care folosește JDBC API?
- (a) Pentru a crește performanța.
  - (b) Pentru a menține integritatea procesului de business
  - (c) Pentru a folosi tranzacții distribuite
  - (d) Nici una din variantele de mai sus
13. Care dintre afirmațiile următoare este adevărată în raport cu limbajul Java?
- (a) Specificatorul static se aplică la clase, attribute și metode;
  - (b) O metodă declarată static poate apela doar alte metode statice din alte obiecte;
  - (c) O metodă declarată static într-o clasă poate referi doar attribute statice și alte metode statice în aceeași clasă;
  - (d) Unui atribut declarat static i se poate asigna o valoare o singură dată;
  - (e) Un atribut declarat static are aceeași valoare pentru toate obiectele unei clase.
14. Care dintre afirmațiile de mai jos este adevărată în relație cu limbajul Java?
- (a) Specificatorul final se aplică la clase, attribute și metode;
  - (b) O metodă declarată final poate apela doar metode declarate final în aceeași clasă;
  - (c) O metodă declarată final nu poate fi suprascrisă la subclasare;
  - (d) Un atribut declarat final poate fi setat doar o singură dată;
15. Care dintre afirmațiile de mai jos este adevărată în relație cu limbajul Java?
- (a) Moștenirea poate fi întotdeauna înlocuită cu compunerea obiectelor;
  - (b) O clasă poate implementa orice număr de interfețe;
  - (c) O enumerare poate implementa orice număr de interfețe;
  - (d) Folosirea compoziției furnizează o posibilitate de a eluda restricțiile de acces;
16. Se consideră următoarea definiție a unei clase Java:
- ```
class A{  
    public static int i=1;  
    public static void main(String[] args) {  
        A x=new A();
```

```
        A y=new A();
        x.i =x.i+1;
        y.i =y.i+1;
    }
}
```

Care dintre afirmațiile următoare este adevărată:

- (a) Clasa nu se compilează corect;
- (b) După rulare, $x.i = 2$ și $y.i=2$;
- (c) După rulare, $x.i = 2$ și $y.i=3$;
- (d) După rulare, $x.i = 3$ și $y.i=3$;
- (e) După rulare, $x.i$ și $y.i$ au alte valori decât cele de mai sus.

17. Care dintre următoarele declarații sunt incorecte în limbajul Java:

- (a) `Integer numbers= new Integer(100);`
- (b) `int nr[]= new Integer[100];`
- (c) `String matrix [][]= new String[2][];`
- (d) `Integer tab= new int[100];`
- (e) `Integer array[]= new Integer[100];`

18. Ce puteți spune despre programul Java de mai jos ?

```
class A {
    A(int x) { System.out.print("Constructor A called "); }
}
class B extends A {
    B() { System.out.print("Constructor B called "); }
}
public class Test {
    public static void main(String args[]){
        B b = new B(); }
}
```

- (a) Nu va fi compilat
- (b) Va afișa: "Constructor B called"
- (c) Va afișa: "Constructor A called"

- (d) Va afișa: "Constructor A called Constructor B called"
19. Care din următoarele afirmații sunt adevărate relativ la codul Java de mai jos (liniile sunt numerotate):
1. public class Test extends Thread{
 2. public void run() {
 3. System.out.print(" Answer ");
 4. wait(1000);
 5. System.out.print("Question ");
 6. }
 7. public static void main(String args []) {
 8. Test ob = new Test();
 9. ob.start();
 10. } }
- (a) Compilarea va eșua la linia 4 deoarece wait(1000) se poate apela doar în cod sincronizat
- (b) Compilarea va eșua la linia 4 deoarece wait(1000) se poate apela doar într-un bloc try/catch;
- (c) Compilarea se va face cu succes. La execuție, nu se va afișa nimic
- (d) Compilarea se va face cu succes. La execuție, se va afișa "Answer "
- (e) Compilarea se va face cu succes. La execuție, se va afișa "Question Answer "
20. Ce este semnătura unei metode în limbajul Java:
- (a) Numele metodei
 - (b) Numele metodei împreună cu numele argumentelor
 - (c) Numele metodei împreună cu numele și tipul argumentelor
 - (d) Numele metodei împreună cu numele și tipul argumentelor și tipul returnat
21. Constructorul în Java:
- (a) Nu are tip de valoare de retur;
 - (b) Poate fi implicit;
 - (c) Are același nume cu clasa;
 - (d) De fiecare dată când este creat un obiect, este apelat un constructor.

22. Care din următoarele secvențe de cod Java aruncă o excepție când apare o problemă:

1.

```
public void someMethod ()  
{ ...  
    if ( problem ) throw new Exception("Useful Message");  
...}
```
2.

```
public void someMethod () throws Exception  
{ ...  
    if ( problem ) Exception("Useful Message");  
...}
```
3.

```
public void someMethod () throw Exception  
{ ...  
    if ( problem ) throws new Exception("Useful Message") ;  
...}
```
4.

```
public void someMethod () throws Exception  
{ ...  
    if ( problem ) throw new Exception("Useful Message");  
...}
```

- (a) 1;
- (b) 2;
- (c) 3;
- (d) 4.

23. Modificatorii de access ai modificatorilor unei clase Java sunt::

- (a) public, protected, default (când nu se specifică modificador), private
- (b) public, protected, private
- (c) public, abstract, final
- (d) public, static, final

24. Care din următoarele afirmatii sunt false relativ la limbajul Java:

- (a) Orice clasă are ca și clasă de bază clasa Object
- (b) O clasă poate extinde una sau mai multe clase de bază
- (c) O clasă poate implementa una sau mai multe interfețe
- (d) Subclasele moștenesc atributele, metodele și constructorii clasei de bază

25. Ce se afișează în urma rulării următoarei secvențe de cod Java?

```
String s1 = "year" + 20 + 18, s2 = 2000 + 18 + "year";  
System.out.println("s1 = " + s1 + ", s2 = " + s2);
```

- (a) s1 = year38, s2 = 20018year
- (b) s1 = year 2018, s2 = 200018 year
- (c) s1 = year 38, s2 = 20018 year
- (d) s1 = year2018, s2 = 2018year
- (e) Eroare la execuție: este necesară o conversie explicită

26. Un fir de execuție Java este:

- (a) O instanță a unei clase derivate din clasa Thread
- (b) O instanță a unei clase care implementează interfața Runnable
- (c) Un obiect al unei clase a cărei superclasă este clasa Thread
- (d) Un obiect al unei clase care implementează interfața Runnable

27. Ce este adevărat despre clasa String (în Java):

- (a) Conținutul unui obiect String nu mai poate fi modificat după ce a fost creat
- (b) Conținutul a două obiecte String se compară prin operatorul ==
- (c) Clasa String este declarată final
- (d) Clasa String are un atribut size care ne dă lungimea șirului de caractere

28. Ce afirmație de mai jos este adevărată despre o clasă Java abstractă:

- (a) O clasă care nu poate fi instanțiată
- (b) O clasă care are cel puțin o metodă abstractă
- (c) O clasă care se definește folosind cuvântul cheie abstract

29. În fișierul A.java sunt următoarele declarații de clase și interfețe Java:

```
interface X {}  
interface Y {}  
public class A implements X {}  
public class B extends A implements X, Y {}
```

Care este motivul pentru care secvența de mai sus nu va fi compilată:

- (a) În Java nu există moștenire multiplă
- (b) Clasa B face o dublă implementare interfețelor

- (c) In fisier există două clase publice
- (d) Pentru a putea fi implementate, interfețele trebuie să fie publice

30. Să considerăm următoarele definiții în limbajul Java:

```
class A implements Runnable {  
    int counter = 0;  
    public void run() {  
        while (true) counter ++;  
    }  
}  
  
public class Test{  
    public static void main(String [] arg) {  
        A a = new A();  
    }  
}
```

Care din variantele de mai jos instanțiază și lansează un fir de execuție ?

- (a) run();
- (b) a.start();
- (c) new Thread(a).run();
- (d) new Thread(a).start();

4 Inginerie software

1. Bifați afirmațiile adevărate. Un caz de utilizare (use case):
 - (a) definește o funcție a sistemului.
 - (b) reprezintă ce trebuie să facă sistemul.
 - (c) arată cum trebuie realizată o funcție a sistemului.
 - (d) obligatoriu este în relație directă cu un actor.
 - (e) definește comportamentul sistemului.
 - (f) definește o structură a sistemului.

2. Fie următoarele exemple de cerințe ale unei aplicații destinată unui laborator de analize medicale. Bifați cerințele funcționale:
 - (a) Sistemul trebuie să permită administrarea tipurilor de analize.
 - (b) Sistemul trebuie să suporte maximum 20 utilizatori simultan.
 - (c) Sistemul trebuie să permită modificarea buletinului de analize.
 - (d) Se va utiliza sistemul de encriptare DES32 pentru datele transferate.
 - (e) Sistemul trebuie să răspundă în maximum 1 secundă la orice comandă utilizator.
 - (f) Facilitatea de HELP se va organiza ierarhic.
 - (g) Se va păstra un istoric al ultimelor 20 de analize efectuate pentru fiecare pacient.
 - (h) Sistemul va fi implementat în Java și va utiliza SGBD Oracle.

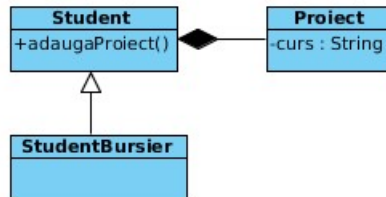
3. Care afirmație caracterizează stilul arhitectural client-server?
 - (a) Subsistemele comunică prin date partajate de dimensiuni mari păstrate într-o bază de date centrală.
 - (b) Sistemul este compus din module funcționale care procesează intrările și produc ieșiri.
 - (c) Sistemul este format din sisteme ce oferă servicii și sisteme ce solicită aceste servicii.
 - (d) Un eveniment este trimis tuturor subsistemelor, acesta fiind tratat de subsistemele interesate.
 - (e) Structurează sistemul pe mai multe nivele de abstractizare.

4. Verificarea software-lui poate implica
 - (a) analiza statică automată
 - (b) evaluarea utilității și utilizabilității software-lui în situații operaționale.
 - (c) depanarea erorilor
 - (d) inspectări ale software-lui
 - (e) testarea în vederea descoperirii existenței erorilor

- (f) testarea faptului că software-ul îndeplinește cerințele utilizator
- 5. Care din următoarele tipuri de produse software intră în categoria CASE
 - (a) Sisteme de operare
 - (b) Medii integrate de dezvoltare
 - (c) Compilatoare
 - (d) Editoare LaTeX
 - (e) Instrumente pentru testare
 - (f) Editoare UML
 - (g) Editoare grafice
 - (h) Software pentru controlul versiunilor
- 6. Diagrama de stări și tranziții reprezintă
 - (a) funcțiile sistemului.
 - (b) răspunsul sistemului la evenimente interne.
 - (c) răspunsul sistemului la evenimente externe.
 - (d) interacțiuni între obiecte din sistem.
 - (e) structura datelor.
 - (f) interacțiunile actorilor cu sistemul.
 - (g) fluxul de prelucrare a datelor în sistem.
- 7. Depanarea codului include
 - (a) Stabilirea existenței erorilor
 - (b) Crearea unei ipoteze despre cauza erorii
 - (c) Localizarea erorilor
 - (d) Creare teste de acceptare
 - (e) Corectarea erorilor
 - (f) Refactorizarea codului
- 8. Metodele agile de dezvoltare de software implică
 - (a) Furnizare incrementală
 - (b) Implicarea clientului pe parcursul procesului de dezvoltare
 - (c) Instituirea de procese normative pentru lucrul în echipă
 - (d) Acțiuni periodice de eliminare a complexității din sistem
 - (e) Modelarea completă a software-lui înainte de scrierea codului

9. Testarea performanței înseamnă:
- (a) aplicarea unei serii de teste în care încărcarea este crescută treptat.
 - (b) procesul de testare a componentelor individuale în regim de izolare.
 - (c) definirea specificațiilor intrărilor testului și a rezultatelor așteptate de la sistem, plus precizarea entității testate.
 - (d) utilizarea sistemului dincolo de încărcarea maximă pentru care a fost proiectat.
 - (e) reaplicarea unui set de teste existent.
10. Cadrele pentru aplicații sunt
- (a) sisteme dezvoltate prin integrarea de sisteme de aplicații existente.
 - (b) un tip de aplicație generalizat în jurul unei arhitecturi comune, astfel încât el poate fi adaptat în moduri diferite pentru clienți diferiți.
 - (c) abstractizări generice care apar în aplicații, reprezentate ca tipare de programare care ilustrează obiecte abstracte și concrete și interacțiuni.
 - (d) colecții de clase abstracte și concrete care pot fi adaptate și extinse pentru a crea sisteme de aplicație.
 - (e) aplicații obținute prin întretesere la compilare, în diferite părți ale unei aplicații, de componente partajate.
11. Bifați situațiile ce pot motiva procesul de mentenanță.
- (a) Sistemul nu trece unul din testele de validare.
 - (b) Apare o modificare la legislația aferentă domeniului aplicației.
 - (c) Apar defecte în timpul testării sistemului.
 - (d) Apar defecte în timpul utilizării sistemului.
 - (e) Utilizatorii produsului software nu sunt mulțumiți de modul de comunicare prin UI.
 - (f) Este identificată o eroare în cursul procesului de inspectare a software-lui.
 - (g) Beneficiarul nu validează prototipul UI.
 - (h) O componentă propusă spre achiziționare nu corespunde specificațiilor sale.
 - (i) Beneficiarul ia decizia utilizării unui alt sistem de gestiune a bazelor de date.
12. Bifați abordările pentru reutilizarea sistemelor de aplicații
- (a) Integrare de produse COTS
 - (b) Utilizare șabloane de programare
 - (c) Programare orientată pe aspecte
 - (d) Dezvoltarea liniilor de produse software

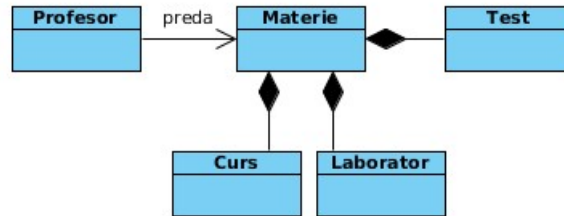
- (e) Extindere cadre pentru aplicații
13. Managementul versiunilor, ca activitate a procesului de management al configurațiilor software, înseamnă
- (a) urmărirea cererilor de modificare a software-lui, analizarea impactul și costului acestora și selectarea modificărilor ce vor fi realizate.
 - (b) urmărirea multiplele versiuni ale componentelor sistemului și asigurarea că modificările asupra componentelor realizate de diferiți dezvoltatori nu interferează între ele.
 - (c) asamblarea componentelor programului, datelor și bibliotecilor, urmată de crearea unui program executabil.
 - (d) pregătirea software-lui pentru lansare externă și urmărirea versiunilor sistemului lansate pentru utilizare la client.
14. Fiabilitatea unui sistem se poate exprima ca cerință verificabilă prin
- (a) numărul de tranzacții procesate pe secundă
 - (b) dimensiunea memoriei
 - (c) rata de reparație a defectelor
 - (d) procentul de instrucțiuni dependente de platforma țintă
 - (e) timpul de instruire al utilizatorilor
 - (f) probabilitatea de disponibilitate a sistemului
15. Fie următoarea diagramă de clase.



Bifați afirmațiile adevărate.

- (a) Clasa **Student** moștenește clasa **StudentBursier**
- (b) Un obiect de tip **Student** conține o colecție de obiecte de tip **Proiect**
- (c) Clasa **Proiect** are un atribut public de tip **String**
- (d) Clasa **Student** are operația publică `adaugăProiect`
- (e) Clasa **Student** are operația privată `adaugăProiect`
- (f) Clasa **Student** este superclasă pentru clasa **StudentBursier**

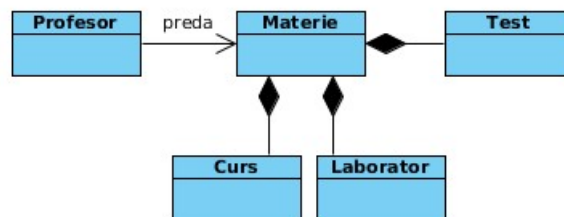
16. Fie următoarea diagramă de clase.



Care secvență de cod Java descrie corect relația dintre clasele Profesor și Materie?

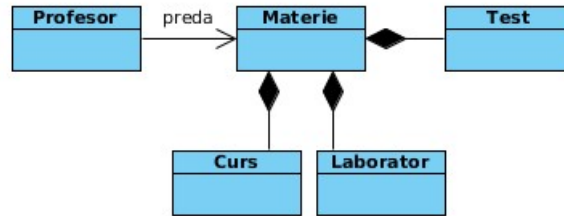
- (a) `class Profesor extends Materie{...}`
- (b) `class Profesor {
 private Materie preda; ...}`
- (c) `class Materie {
 private Profesor preda; ...}`
- (d) `class Materie {
 private Vector<Materie> preda;...}`

17. Fie următoarea diagramă de clase.



Care afirmații sunt adevărate?

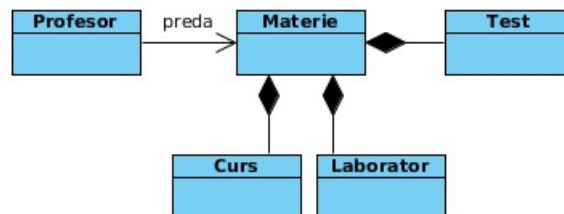
- (a) Clasa `Materie` definește o compoziție de obiecte de tip `Curs`.
 - (b) între clasa `Profesor` și clasa `Materie` există o asociere bidirecțională.
 - (c) Clasa `Test` moștenește clasa `Materie`.
 - (d) Clasa `Materie` definește un agregat de obiecte de tip `Laborator`.
 - (e) Un obiect de tip `Materie` conține o colecție de obiecte de tip `Test`
18. Fie următoarea diagramă de clase.



Care secvență de cod Java descrie corect și complet relația clasei *Materie* cu clasa *Laborator*?

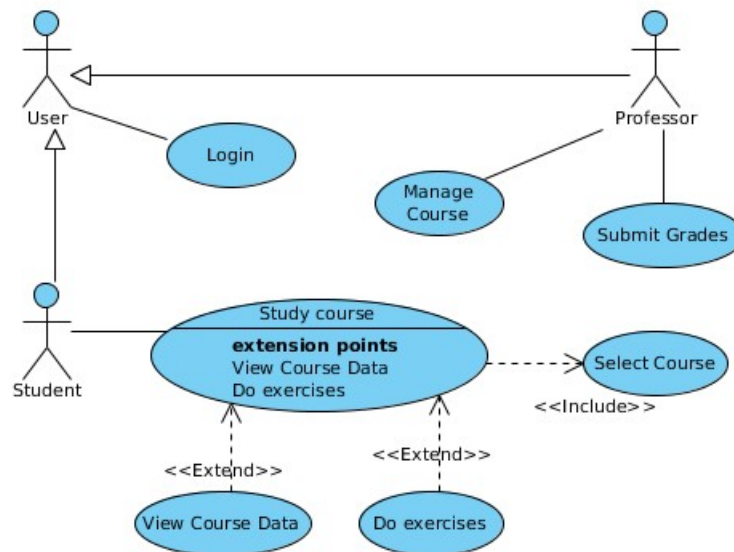
- (a) `class Materie extends Laborator...`
- (b) `class Laborator extends Materie{...}`
- (c) `class Materie {
 private Vector <Laborator> laboratoare = new Vector();...}`
- `class Laborator {
 private Materie materie;
 ...}`
- (d) `class Materie {
 private Laborator laborator;...}`
- `class Laborator {
 private Vector<Materie> materie;
 ...}`
- (e) `class Materie {
 private Vector <Laborator> laboratoare;...}`
- `class Laborator {
 private Materie materie;
 ...}`

19. Fie următoarea diagramă de clase.



Selectați descrierea corectă și completă a relațiilor reprezentate în diagramă:

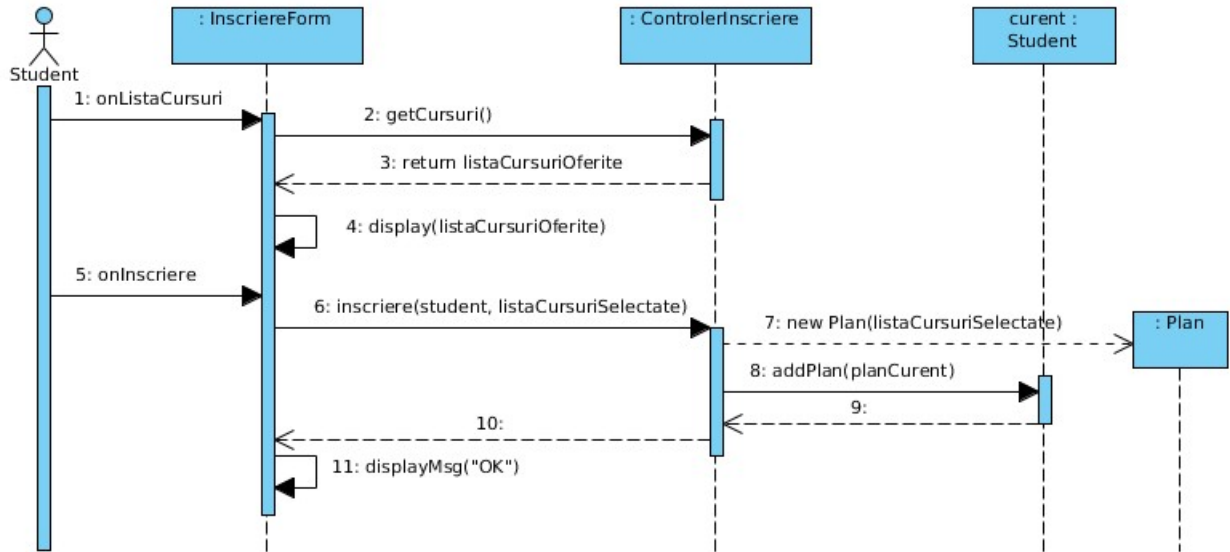
- (a) Asociere între clasele **Profesor** și **Materie**; agregare între clasele **Materie**(agregat) și **Test**(componenta), **Materie**(agregat) și **Laborator**(componenta), **Materie**(agregat) și **Curs**(componenta).
- (b) Asociere unidirecțională, numită *preda*, de la clasa **Profesor** la clasa **Materie**; agregare între clasele **Materie**(agregat) și **Test**(componenta), **Materie**(agregat) și **Laborator**(componenta), **Materie**(agregat) și **Curs**(componenta).
- (c) Asociere unidirecțională, numită *preda*, de la clasa **Profesor** la clasa **Materie**; clasa **Materie** este superclasă pentru clasele **Test**, **Laborator** și **Curs**.
- (d) Asociere unidirecțională, numită *preda*, de la clasa **Profesor** la clasa **Materie**; compoziție între clasele **Materie**(compozit) și **Curs**(componenta); compoziție între clasele **Materie**(compozit) și **Laborator**(componenta); compoziție între clasele **Materie**(compozit) și **Test**(componenta).
- (e) Asociere unidirecțională, numită *preda*, de la clasa **Profesor** la clasa **Materie**; clasele **Curs**, **Laborator** și **Test** moștenesc clasa **Materie**.
20. Fie următoarea diagramă de cazuri de utilizare.



Selecțai toate cazurile de utilizare implicate direct în realizarea funcțiilor accesibile studentului:

- (a) Login
 (b) Manage Course
 (c) Select Course
 (d) Study Course
 (e) View Course Data
 (f) Do exercises
 (g) Submit Grades

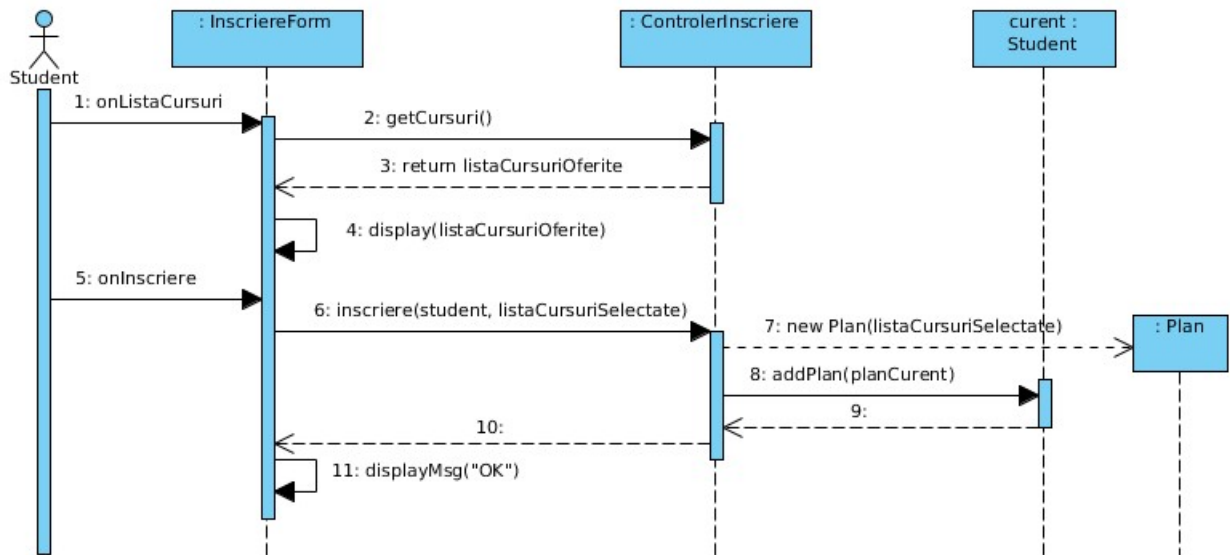
21. Fie următoarea diagramă de secvențe.



Ce operații ale clasei `ControlerInsciere` rezultă din aceasta?

- (a) `getCursuri()`
- (b) `display(listaCursuriOferite)`
- (c) `inscriere(student, listaCursuriSelectate)`
- (d) `Plan(listaCursuriSelectate)`
- (e) `addPlan(planCurent)`
- (f) `displayMsg(OK)`

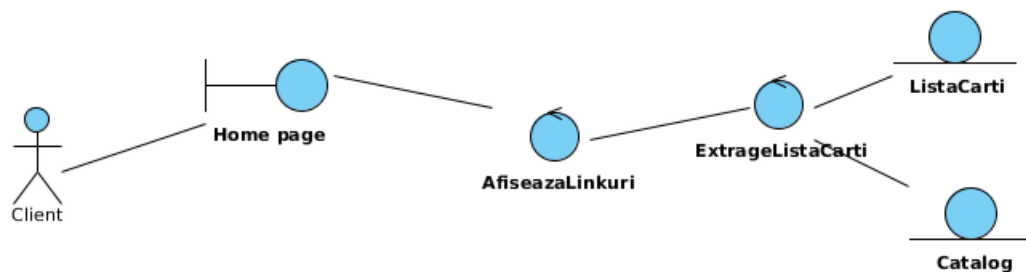
22. Fie următoarea diagramă de secvențe.



Selecțai clasele din care sunt instanțiate obiectele implicate în interacțiune:

- (a) InsciereForm
- (b) ControlerInsciere
- (c) listaCursuriSelectate
- (d) curent
- (e) Student
- (f) Plan
- (g) planCurent

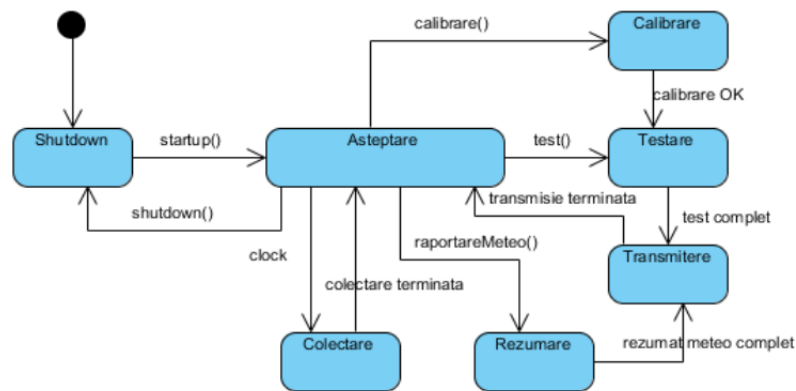
23. Fie următoarea diagramă de robustețe.



Care afirmații sunt adevărate?

- (a) Home page este obiect *boundary*.
- (b) ExtrageListaCarti poate fi obiect persistent.
- (c) AfiseazaLinkuri este obiect de interacțiune cu actor.

- (d) `ExtrageListaCarti` ar putea fi implementat ca metodă a unei clase *entity*.
- (e) `ExtrageListaCarti` poate fi obiect persistent.
- (f) `ListaCarti` este obiect de interacțiune cu actor.
- (g) `Catalog` este obiect *entity*.
24. Selectați afirmațiile care respectă regulile de construire a diagramei de robustețe:
- (a) Actorii trebuie să comunice cu obiecte *boundary*.
- (b) Actorii pot comunica cu obiecte *control*.
- (c) Obiectele *boundary* pot comunica cu obiecte *control*.
- (d) Obiectele *boundary* nu pot comunica cu obiecte *entity*.
- (e) Obiectele *entity* pot comunica cu obiecte *control*.
- (f) Obiectele *entity* pot comunica cu actori.
- (g) Obiectele *control* nu pot comunica cu alte obiecte *control*.
25. Fie următoarea diagramă de stări și tranziții.

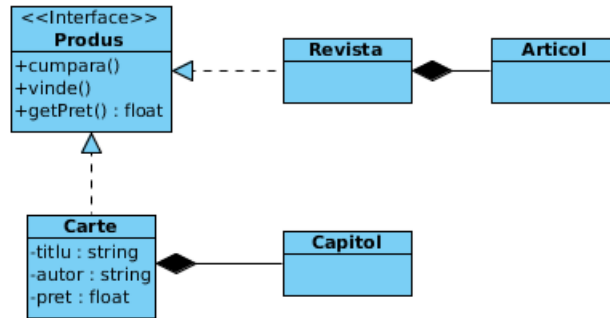


Care afirmații sunt adevărate?

- (a) `Testare` este stare.
- (b) `test()` este un eveniment intern ce declanșează o tranziție de la starea `Așteptare` la starea `Testare`.
- (c) `test complet` este eveniment intern.
- (d) `test complet` este tranziție.
- (e) Revenirea în starea `Așteptare` după apariția evenimentului `calibrare()` se face trecând doar prin stările `Calibrare` și `Testare`.
- (f) `test complet` este eveniment extern ce declanșează o tranziție de la starea `Testare` la starea `Transmitere`.

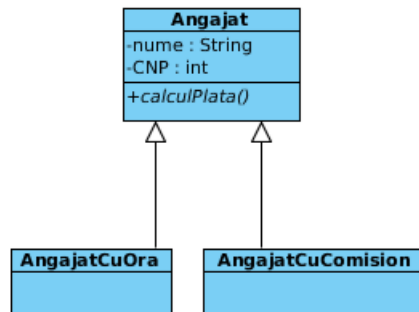
- (g) `test complet` este eveniment intern ce declanșează o tranziție de la starea `Transmitere` la starea `Testare`.
- (h) Revenirea în starea `Așteptare` după apariția evenimentului `calibrare()` se face trecând prin stările `Calibrare`, `Testare` și `Transmitere`.

26. Fie următoarea diagramă de clase.



Care secvențe de cod Java sunt valide pentru clasa `Carte`?

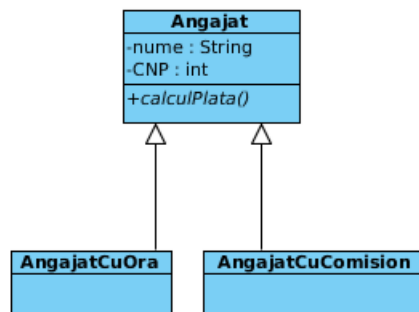
- (a) `class Carte extends Produs{...}`
- (b) `private float pret;`
- (c) `private pret float;`
- (d) `public float getPret();`
- (e) `class Carte implements Produs{...}`
- (f) `private float pret(){...}`
- (g) `public float getPret(){...}`
- (h) `public Produs cumpara(){...}`
- (i) `private Vector<Capitol> capitole = new Vector();`
27. Fie următoarea diagramă de clase.



Care secvență de cod Java definește corect și complet ceea ce rezultă din diagramă pentru clasa `Angajat`?

- (a) `class Angajat {`
 `private String nume;`
 `private int CNP;`
 `public abstract void calculPlata();`
 `...}`
- (b) `abstract class Angajat {`
 `private String nume;`
 `private int CNP;`
 `public abstract void calculPlata();`
 `...}`
- (c) `abstract class Angajat {`
 `private String nume;`
 `private int CNP;`
 `public abstract void calculPlata(){...};`
 `...}`
- (d) `abstract class Angajat {`
 `private String nume;`
 `private int CNP;`
 `public void calculPlata();`
 `...}`

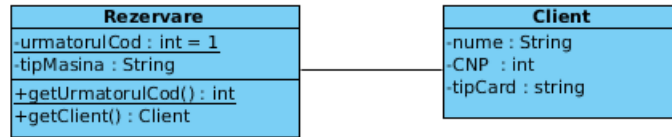
28. Fie următoarea diagramă de clase.



Care secvențe de cod Java sunt valide pentru clasa `AngajatCuOra`?

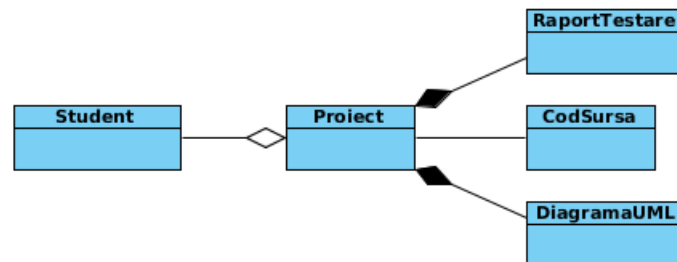
- (a) `class AngajatCuOra extends Angajat{...}`
 (b) `class AngajatCuOra implements Angajat{...}`
 (c) `public calculPlata();`
 (d) `public calculPlata(){...};`

29. Care secvențe de cod Java sunt valide pentru clasa Rezervare?



- (a) `class Rezervare extends Client{...}`
- (b) `private int urmatorulCod = 1;`
- (c) `private static int urmatorulCod = 1;`
- (d) `public static int getUrmatorulCod();`
- (e) `private Client client;`
- (f) `public static Client getClient();`

30. Fie următoarea diagramă de clase.



Selectați afirmațiile valide.

- (a) Clasa `Student` definește un agregat de obiecte de tip `Proiect` iar clasele `RaportTestare` și `DiagramaUML` definesc compoziții de obiecte de tip `Proiect`.
- (b) Clasa `Proiect` definește compoziții de obiecte de tip `Student`, de tip `DiagramaUML` și de tip `RaportTestare`.
- (c) Clasa `Proiect` definește un agregat de obiecte de tip `Student` și compoziții de obiecte de tip `DiagramaUML` și de tip `RaportTestare`.
- (d) Clasa `Proiect` definește o compoziție de obiecte de tip `Student` și agregate de obiecte de tip `DiagramaUML` și de tip `RaportTestare`.
- (e) Clasa `Proiect` este în relație de asociere cu clasa `CodSursa`.

5 Răspunsuri

Limbajul C

- | | |
|-------------|-----------------|
| 1. 1a,1d,1e | 16. 16c |
| 2. 2c | 17. 17b |
| 3. 3b | 18. 18e |
| 4. 4c | 19. 19a,19d |
| 5. 5c | 20. 20c |
| 6. 6b | 21. 21a |
| 7. 7d | 22. 22c,22d,22e |
| 8. 8b,8c | 23. 23b |
| 9. 9b,9d | 24. 24d |
| 10. 10b | 25. 25c |
| 11. 11a | 26. 26a |
| 12. 12c | 27. 27c |
| 13. 13d | 28. 28c |
| 14. 14c | 29. 29b |
| 15. 15b | 30. 30a,30b |

Limbaajul C++

- | | |
|---------------------|-----------------|
| 1. 1a,1c,1e,1g | 16. 16c |
| 2. 2a,2c,2d,2e | 17. 17a |
| 3. 3c,3d | 18. 18e |
| 4. 4a,4b,4d,4e | 19. 19d |
| 5. 5a,5c,5d,5e | 20. 20b |
| 6. 6a,6b,6c,6d | 21. 21d |
| 7. 7a | 22. 22c |
| 8. 8d,8e,8g | 23. 23c |
| 9. 9c | 24. 24a |
| 10. 10b,10c,10e,10f | 25. 25b |
| 11. 11c | 26. 26c,26e |
| 12. 12b,12e,12g | 27. 27d |
| 13. 13b | 28. 28b,28d,28e |
| 14. 14c | 29. 29a |
| 15. 15b | 30. 30b,30c |

Limvajul Java

1. 1d
2. 2a,2d
3. 3a,3b,3d,3e
4. 4b
5. 5a
6. 6c
7. 7b,7d
8. 8b,8d
9. 9c
10. 10a,10b
11. 11c
12. 12a,12b,12c
13. 13a,13c,13e
14. 14a,14c,14d
15. 15b,15c
16. 16d
17. 17b,17d
18. 18a
19. 19b
20. 20c
21. 21a,21b,21c,21d
22. 22d
23. 23a
24. 24b,24d
25. 25d
26. 26a,26c
27. 27a,27c
28. 28a,28c
29. 29c
30. 30d

Inginerie Software

1. 1a,1b
2. 2a,2c,2g
3. 3c
4. 4a,4d,4e
5. 5b,5c,5e,5f,5h
6. 6b,6c
7. 7b,7c,7e
8. 8a,8b,8d
9. 9a
10. 10d
11. 11b,11d,11e,11i
12. 12a,12d
13. 13b
14. 14c,14f
15. 15b,15d,15f
16. 16b
17. 17a,17d,17e
18. 18c
19. 19d
20. 20a,20c,20d,20e,20f
21. 21a,21c
22. 22a,22b,22e,22f
23. 23a,23d,23g
24. 24a,24c,24d,24e
25. 25a,25c,25h
26. 26b,26e,26g,26i
27. 27b
28. 28a,28d
29. 29c,29d,29e
30. 30c,30e