Examen de licenţă 2017 - Informatică
Exemple de întrebări - Limbaje de programare şi inginerie software

In atenţia studenţilor:

Proba scrisă a examenului de licenţă din sesiunile iulie-septembrie 2017 va consta din 60 de întrebări similare, ca structură şi nivel de dificultate, celor din această culegere. Pentru fiecare dintre cele trei categorii (Structuri discrete şi algoritmi, Limbaje de programare şi inginerie software, Sisteme de calcul şi baze de date) vor fi câte 20 de întrebări.

Pentru neclarităţi privind enunţurile sau răspunsurile puteţi să vă adresaţi celor care au propus întrebările pentru fiecare secţiune.

Limbajul C:

- Victoria Iordan (victoria.iordan@e-uvt.ro)
- Cosmin Bonchiş (cosmin.bonchis@e-uvt.ro)

Limbajul C++:

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

Limbajul Java:

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

Inginerie software:

- Cristina Mîndruţă (cristina.mindruta@e-uvt.ro)

1 Limbajul C

1. Care din următoarele acţiuni au loc la compilare?

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

2. In condiţiile

   ```
   #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 defineşte corect şi complet ce anume se specifică prin tipul unei date?

   (a) Numărul de octeţi ocupaţi
   (b) Spaţiul necesar reprezentării şi modul de reprezentare;
   (c) Operaţiile 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 ţine din locul declaraţiei până la:

   (a) sfârşitul blocului;
   (b) sfârşitul programului;
   (c) sfârşitul textului sursă;
   (d) sfârşitul funcţiei;

6. La execuţia programului următor se tastează 20. Cese va afişa după execuţie?

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

   (a) 20;
   (b) 2;
   (c) 0;

7. Care este durata de viaţă a variabilelor locale?

   (a) cât timp sunt utilizate;
   (b) câteva minute;
   (c) cât dureaza execuţia subprogramului;
   (d) depinde de clasa lor de memorare;

8. Când au loc conversii implicite de tip?

   (a) la citirea datelor de intrare şi la afişarea rezultatelor;
   (b) când operanzii unei expresii au tipuri diferite;
   (c) când argumentul actual cu care se apelează o funcţie are alt tip decât argumentul formal corespunzător;

9. Date declaraţiile

```
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 declaraţiile:

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

şi presupunând că atât i, cât şi t nu sunt initializaţi 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 condiţiile în care c este declarată

```
char c;
```

valoarea expresiei

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

(a) este valoarea returnată de funcţia getchar;

(b) este 0;

(c) este 1;

(d) este 0 sau 1;

14. Ce se va afişa în urma execuţiei secvenţei următoare:

```
#include <stdio.h>
void main() {
    printf("%d",10>20 }
```

(a) false;

(b) error;

(c) 0;

(d) 1;

15. Ce se va afişa în urma execuţiei secvenţei 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 declaraţiile:

int x=10, y=20;

Care este valoarea expresiei !x-y?

(a) 0;

(b) -10;

(c) -20;

17. Fie declaraţiile:

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 secvenţa 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 condiţionale 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 afişează 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 afişează 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 modalităţi de comunicare între funcţii?

(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 funcţii?

(a) lista argumentelor formale
(b) o zona de memorie (de pe stivă)
(c) locul din textul sursă în care se apelează
(d) instrucţiunile (definiţia funcţiei)

24. Care este efectul secvenţei:

```
for(i=0; i<N; i++);
        printf("\n %d",i);
```

(a) se afişează pe linii separate valorile de la 1 la N
(b) se afişează pe linie noua valoarea N-1
(c) se afişează pe linii separate valorile de la 0 la N-1
(d) se afişează pe linie nouă valoarea N

25. Ce afişează 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 afişează 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 dupa execuţia secvenţei:

```
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 declaraţia: int *(*f)(int *) ?

(i) funcţie ce primeşte argument pointer la întreg şi întoarce pointer la întreg

(ii) o declaraţie greşită

(iii) pointer către o funcţie care aşteaptă ca argument un pointer la int şi întoarce un pointer la int

(iv) pointer către o funcţie care întoarce un int

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

29. Ce se afişează?

```
#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 afirmaţiile următoare, referitoare la structuri, sunt adevıarate:

(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 proprietăţi în C++:
   (a) Este o funcţie membră care are acelaşi nume ca şi clasa în care este declarată
   (b) Este o funcţie membră care întoarce o valoare
   (c) Este o funcţie membră care nu are valoare de return
   (d) Este o funcţie membră care întotdeauna nu are parametri
   (e) Este o funcţie membră utilizată pentru a iniţializa un obiect
   (f) Este o funcţie membră utilizată pentru a dealoca spaţiu de memorie
   (g) Este o funcţie membră care se foloseşte împreună cu operatorul new
   (h) O clasă nu poate avea mai mult de un constructor
   (i) Este o metodă care se foloseşte împreună cu operatorul delete
   (j) Poate fi o funcţie membră virtuală
2. Care dintre următoarele facilităţi sunt suportate în limbajul C++ dar nu sunt suportate în limbajul C?
   (a) Funcţii care au valori implicite ale parametrilor
   (b) Macro definiţii
   (c) Tipul de dată referinţă
   (d) Operatorul de rezoluţie
   (e) Supraîncărcarea funcţiilor
   (f) Cuvântul cheie const
3. Dacă *a* şi *b* sunt două obiecte de tipul clasei *Text* şi clasa conţine o metodă cu următorul prototip: static bool equals(const Test&, const Test&);? Care este apelul corect pentru funcţia *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 afirmaţii sunt adevărate?
   (a) Operatorii pot fi supraîncărcaţi în limbajul C++.
   (b) Limbajul Java rulează într-o maşină virtuală.
   (c) Limbajele Java şi C++ sunt independente de platformă.

(d) Variabile referinţă sunt caracteristice pentru limbajele C++ şi Java.

(e) Limbajul C++ nu are implementat un mecanism de garbage collection.

(f) Limbajul C++ nu permite moştenire multiplă.

5. Care dintre următoarele afirmaţii sunt false?

   (a) În limbajele Java şi C++ nu contează ordinea în care sunt definite şi apelate metodele.

   (b) În limbajele Java şi C++ specificarea comentariilor se realizează la fel.

   (c) În cazul tratării excepţiilor prin mecanismul try ... catch în limbajele Java şi C++ există o clauză care se execută în orice caz (chiar dacă s-a aruncat sau nu o excepţie).

   (d) În limbajele Java şi C++ există o clasă care este supraclasă pentru toate clasele.

   (e) Declararea claselor în Java şi C++ trebuie să se termine cu punct şi virgulă (;).

   (f) În limbajul C++ nu toate metodele trebuie definite în interiorul clasei.

6. Care dintre următoarele afirmaţii sunt adevărate despre template-uri în C++?

   (a) Template-urile sunt o facilitate a limbajului C++ care permit folosirea acelaşi cod pentru diferite tipuri de date

   (b) Permit scrierea unei funcţie care se foloseşte 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 înlănţuite, stive, cozi ...

   (d) Template-urile sunt exemple de polimorfism la execuţie.

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 conţină:

   (a) Doi constructori

   (b) Un destructor care nu face nimic $X(){}$

   (c) O funcţie 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 execuţiei următoarei secvenţe 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 conţin streamul de ieşire cout spunănd că tipul datelor nu este specificat

(c) Eroare de compilare la ultima linie care conţin streamul de ieşire cout spunănd că apelul funcţei max este ambigu.

(d) Nici una din afirmaţile de mai sus.

10. Care dintre următoarele recomandări sunt bine de urmat pentru scrierea unei aplicaţii:

(a) Scrierea de funcţii care conţin multe linii de cod

(b) Menţinerea codului simplu şi eliminarea complexităţii nenecesare

(c) Codul trebuie scris astfel încât să fie logic şi uşor de înţeles de oricine

(d) Scrierea de comentarii pentru fiecare linie de cod

(e) Evitarea codului care se repetă şi extragerea lui în clase, funcţii

(f) Separarea codului în module, fiecare modul concentrându-se pe o anumită cerinţă

(g) Crearea de interfeţe generale care servesc tuturor cerinţelor şi neîmpărţirea lor în interfeţe mai mici care deservesc cerinţe specifice

11. Pentru următoarele clase care este ordinea corectă de apelare a constructorilor şi 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 afirmaţii 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 afişa 10 20 30

(b) După executarea liniei 14 se va afişa 100 200 300

(c) După executarea liniei 14 se va afişa 0 0 0

(d) După executarea liniei 16 se va afişa 20

(e) După executarea liniei 16 se va afişa 30

(f) După executarea liniei 17 se va afişa 2

(g) După executarea liniei 17 se va afişa 1

(h) După executarea liniei 17 se va afişa 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 supraclasei";
  } 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 supraclasei

(c) Eroare de compilare

(d) Alta excepţie

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 excepţie

(e) FormaPatrat

(f) PatratForma

15. Rezultatul executării următoarei secvenţe 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 locaţie de memorie este eliberată de mai multe ori.

(c) Eroare la runtime deoarece se eliberează o locaţie de memorie pentru care nu a fost alocată memorie prin program.

(d) Totul este corect şi codul se compilează şi rulează fără probleme.

16. Care este rezultatul executării urmatoarei secvenţe 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 afişa 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 afişează nimic

19. Dacă B este clasă de bază publică pentru clasa A, care dintre următoarele două declaratii sunt corecte: (1) A *p = new B; sau (2) B *p = new A;?

(a) ambele, deoarece pointerii la clase de bază/derivate pot fi iniţializaţi 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 afirmaţii sunt adevărate în limbajul C++?

(a) O funcţie statică nu poate arunca o excepţie.
(b) O funcţie statică nu poate returna o variabilă membru non-statică a clasei.
(c) O funcţie statică nu poate returna o variabilă membru statică a clasei.
(d) O variabilă membru statică a clasei nu poate fi modificată îintr-o funcţie membră constantă non-statică.

21. Fie declaraţia de mai jos. Cum poate fi accesat membrul Counter fară a crea o instanţă 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 declaraţia clasei Foo şi funcţia foo definită mai jos, care dintre următoarele afirmaţii este adevarată?

```
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 şi este folosit.

(c) Este invocat constructorul de copiere la iniţializarea obiectului f2.

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

    (a) OCP

    (b) SRP

    (c) LSP

    (d) DRY

24. Ce va afişa secvenţa 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 subtipuri de angajaţi ai unei instituţii. Fiind dată funcţia 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 funcţia print() din clasa derivată

(b) Se va apela corect funcţia print() din clasa derivată doar dacă funcţia print() este virtuala în clasa Angajat

(c) Se va apela întotdeauna funcţia 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. Excepţiile sunt:

(a) erori care apar la compilarea programului

(b) situaţii speciale tratate în program prin teste de tipul if(variabila == NULL)

(c) erori care apar la rularea programului

(d) 'aruncate' folosind instrucţiunea try şi 'tratate' folosind instrucţiunea catch

(e) 'aruncate' folosind instrucţiunea throw şi 'tratate' folosind instrucţiunile try si catch

27. Principiul OCP se referă la:

(a) responsabilităţile 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 şi evitarea modificarilor claselor şi codului existent

28. Care dintre următoarele afirmaţii sunt adevarate?

(a) În cazul claselor şablon, compilatorul va genera cod pentru toate şabloanele declarate, indiferent dacă sunt instanţiate cu un tip concret sau nu.

(b) În cazul funcţiilor şablon, se va genera cod doar în cazul apelurilor realizate.

(c) Erorile în declaraţiile şabloanelor pot fi descoperite toate la compilare.

(d) Pot exista erori in declaratiile şabloanelor care sunt descoperite abia la instanţierea şablonului.

(e) Un parametru non-type al unui şablon, de ex. $template < int\ i > \ class\ X\{\}$, poate fi instanţiat doar cu valori sau expresii constante de acel tip (in exemplu, int).

29. O clasă de bază virtuală

(a) este iniţializată o singură dată în cazul unei relaţii de moştenire multiplă de tip diamant.

(b) are toate funcţiile membre virtuale

(c) are toate funcţiile 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 obţine în C++ prin intermediul funcţiilor membre virtuale

(d) este posibilitatea să avem mai multe funcţii cu acelaşi nume în cadrul unei clase.

3 Limbajul Java

1. Se consideră următoarele definiţii în limbajul Java:

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

Care dintre următoarele afirmaţii 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) Definiţia interfeţei este incorectă pentru că interfeţele nu pot avea atribute;
(d) Clasa A nu se compilează deoarece membrul counter nu poate fi modificat;
(e) Interfaţa I nu se compilează deoarece nu poate defini atribute ci doar operaţii.

2. Se consideră următoarele definiţtii 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 afirmaţii este adevărată?

(a) Clasa C se compilează

(b) Clasa C nu se compilează deoarece nu implementează complet interfaţa I1;

(c) Clasa C nu se compilează deoarece nu implementează complet interfaţa 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 afirmaţii 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 şi C2 extind C0;

(d) Clasa C beneficiază de polimorfism deoarece atât C1 cât csi C2 implementează I;

(e) Interfaţa I nu se compilează deoarece nu are operaţii definite.

4. La ce se referă conceptul de "încapsulare a datelor"?

   (a) Incapsularea unei expertize specifice într-o clasă;
   (b) Ascunderea accesului la starea privata a unui obiect;
   (c) Declararea ca private a tuturor atributelor unei clase;
   (d) Includerea numai de atribute în definiţia unei clase, fără a include metode;
   (e) Ascunderea accesului la întreaga stare a unui obiect.

5. Care dintre afirmaţiile următoare sunt false legat de limbajul Java?

   (a) Moştenirea multiplă este obţinută prin aplicarea de mai multe ori a moştenirii simple;
   (b) Moştenirea simplă este echivalentă cu implementarea unei singure interfeţe;
   (c) Doar clasele abstracte pot beneficia de moştenire multiplă;
   (d) Limbajul are moştenire extinsă care poate înlocui moştenirea multiplă;

6. Se consideră următoarea definitie a unei clase Java:

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

   Care dintre următoarele afirmatii 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 sa implementeze cele două metode din clasa C.

7. Se consideră următoarea definiţie a unei clase Java:

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

   Care dintre următoarele afirmaţii sunt adevărate?

   (a) Vizibilitatea metodei m1 este public deoarece clasa are vizibilitate public;

(b) Clasele din acelaşi pachet pot apela metoda m1;

(c) Clasele din acelaşi pachet şi 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 afirmaţii este adevărată?

(a) Primul constructor aruncă o excepţie ClassCastException la asignare;

(b) Metoda computeSum poate arunca excepţia ArrayIndexOutOfBoundsException;

(c) Constructorii aruncă excepţia NullPointerException dacă argumentul este null;

(d) Metoda computeSum poate arunca excepţia NullPointerException.

9. Se consideră codul Java:

```
public class A{
        public static void main(String[] args){
                double d0 = 0.1;
```

```
        double d1 = 1 - 9 * 0.1;
        System.out.println(d0 == d1);
    }
}
```

Ce valoare tipăreşte acest cod?

(a) true;
(b) false;
(c) true sau false, depinzând de compilator;
(d) clasa nu se compilează;
(e) 0.1.

10. Se consideră codul Java:

```
import java.util.HashSet;
public class A{
    private String name;
    public A(String name){
        setName(name);
    }
    public void setName(String name){
        this.name = name;
    }
    public static void main(String[] args){
        HashSet<A> index = new HashSet<A>();
        A a = new A("student");
        index.add(a);
        a = new A("student");
        index.add(a);
    }
}
```

Care dintre afirmaţiile următoare sunt adevărate?

(a) La sfârsitul metodei main, mulţimea set va conţine două obiecte;
(b) La sfârsitul metodei main, mulţimea index va conţine un singur object;

(c) Clasa nu se va compila deoarece clasa obiectelor care se stochează in HashSet sau HashMap trebuie să definească metoda hashCode method iar clasa A nu face acest lucru;

(d) Comportamentul este nedefinit: mulţimea poate conţine atât un obiect cât şi două obiecte, depinzând de compilator;

(e) Clasa A nu este imutabilă.

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 afirmaţiile următoare sunt adevărate in relaţie cu limbajul Java intr-un context "single-thread"?

(a) Ambele metode sunt implementate corect şi calculează aceeasi sumă pentru acelaşi argument;

(b) Metoda processList2 nu se va compila;

(c) Metoda processList1 poate arunca excepţia ConcurrentModificationException;

(d) Metoda processList2 poate arunca excepţia ConcurrentModificationException;

(e) Ambele metode golesc lista primită ca argument, ca şi un efect colateral.

12. Se considera urmatoarea definitie a unei clase Java:

```
public class A{
        private String name;
        public A(String name){
                this.name = name;
        }
        public void setName(String name){
                this.name = name;
        }
        public int hashCode(){
                return name.hashCode();
        }
        public boolean equals(Object o){
                return name.equals(o.toString());
        }
}
```

Care dintre afirmaţiile următoare sunt adevărate?

(a) Plasarea obiectelor clasei A într-un HashMap este corectă deoarece clasa A implementează metodele hashCode şi equals şi ambele sunt bazate pe acelaşi atribut, name;

(b) Metoda equals nu este corect implementată;

(c) Clasa A este imutabilă;

(d) Metoda equals ar fi corectă dacă s-ar include un test adiţional asupra valorii null primită ca şi argument;

(e) Metoda equals apelată cu acelaşi argument pentru acelaşi obiect nu este garantat a returna aceeaşi valoare.

13. Care dintre afirmaţiile următoare este adevarată in raport cu limbajul Java?

(a) Specificatorul static se aplică la clase, atribute si metode;

(b) O metodă declarată static poate apela doar alte metode statice din alte obiecte;

(c) O metoda declarată static într-o clasă poate referi doar atribute statice şi alte metode statice în aceeaşi clasă;

(d) Unui atribut declarat static i se poate asigna o valoare o singură dată;

(e) Un atribut declarat static are aceeaşi valoare pentru toate obiectele unei clase.

14. Care dintre afirmaţiile de mai jos este adevărată in relaţie cu limbajul Java?

    (a) Specificatorul final se aplică la clase, atribute si metode;
    (b) O metodă declarată final poate apela doar metode declarate final în aceeaşi clasă;
    (c) O metodă declarată final nu poate fi suprascrisă la subclasare;
    (d) Un atribut declarat final foate fi setat doar o singură dată;

15. Care dintre afirmaţiile de mai jos este adevărată în relaţie cu limbajul Java?

    (a) Moştenirea poate fi întotdeauna înlocuită cu compunerea obiectelor;
    (b) O clasă poate implementa orice număr de interfeţe;
    (c) O enumerare poate implementa orice număr de interfeţe;
    (d) Folosirea compoziţiei furnizează o posibilitate de a eluda restricţiile de access;
    (e) Compunerea obiectelor este referită şi ca "agregarea obiectelor".

16. Se consideră următoarea definiţie 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 afirmaţiile următoare este adevărată:

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

17. Care dintre următoarele declaraţii 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 puteţi 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 afişa: "Constructor B called"

(c) Va afişa: "Constructor A called"

(d) Va afişa: "Constructor A called Constructor B called"

19. Care din următoarele afirmaţii 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 eşua la linia 4 deoarece wait(1000) se poate apela doar în cod sincronizat

(b) Compilarea va eşua la linia 4 deoarece wait(1000) se poate apela doar într-un bloc try/catch;

(c) Compilarea se va face cu succes. La execuţie, nu se va afişa nimic

(d) Compilarea se va face cu succes. La execuţie, se va afişa "Answer "

(e) Compilarea se va face cu succes. La execuţie, se va afişa "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 şi tipul argumentelor

(d) Numele metodei împreună cu numele şi tipul argumentelor şi tipul returnat

21. Constructorul in Java:

(a) Nu are tip de valoare de retur;

(b) Poate fi implicit;

(c) Are acelaşi nume cu clasa;

(d) De fiecare dată când este creat un obiect, este apelat un constructor.

22. Care din următoarele secvenţe de cod Java aruncă o excepţie 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 modificator), 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 este derivată din clasa Object
(b) O clasă poate extinde una sau mai multe clase de bază
(c) O clasă poate implementa una sau mai multe interfeţe
(d) Subclasele moştenesc atributele, metodele şi constructorii clasei de bază

25. Ce se afişează în urma rulării următoarei secvenţe de cod Java?
String s1 = ”year” + 20 + 13, s2 = 2000 + 13 + ”year”;
System.out.println(”s1 = ” + s1 + ”, s2 = ” + s2);

(a) s1 = year33, s2 = 20013year
(b) s1 = year 2013, s2 = 200013 year
(c) s1 = year 33, s2 = 20013 year
(d) s1 = year2013, s2 = 2013year
(e) Eroare la execuţie: este necesară o conversie explicită

26. Un fir de execuţie Java este:

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

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

(a) Conţinutul unui obiect String nu mai poate fi modificat după ce a fost creat
(b) Conţinutul a două obiecte String se compară prin operatorul ==

(c) Clasa String este declarata final
(d) Clasa String are un atribut size care ne dă lungimea şirului de caractere

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

(a) O clasă care nu poate fi instanţiată
(b) O clasă care are cel puţin o metodă abstractă
(c) O clasă care se defineşte folosind cuvântul cheie abstract

29. In fişierul A.java sunt următoarele declaraţii de clase şi interfeţe Java:

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

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

(a) In Java nu există moştenire multiplă
(b) Clasa B face o dublă implementare interfeţelor
(c) In fisier există două clase publice
(d) Pentru a putea fi implementate, interfeţele trebuie să fie publice

30. Să considerăm următoarele definiţii 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 instanţiază şi lansează un fir de execuţie ?

(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 funcționalitate 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 comadă 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 întrețesere 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

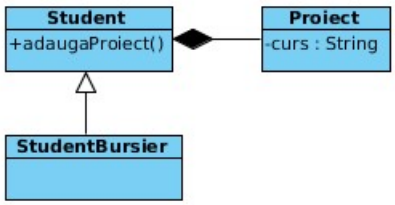
(c) rata de reapariție a defectelor

(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`

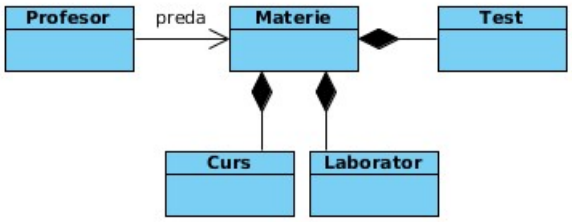
(c) Clasa `Proiect` are un atribut public de tip `String`

(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`?

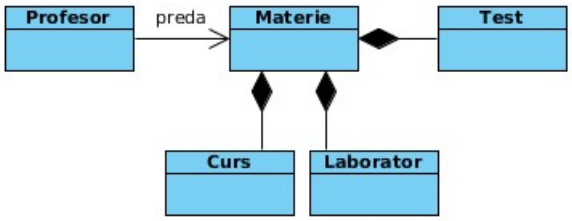
(a) `class Profesor extends Materie{...}`

(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`.

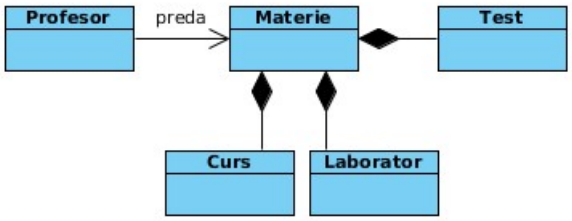
(b) între clasa `Profesor` și clasa `Materie` există o asociere bidirecțională.

(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;
     ...}
```

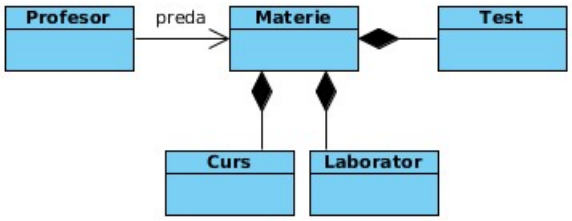
(d)
```
class Materie {
    private Laborator laborator;...}

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).

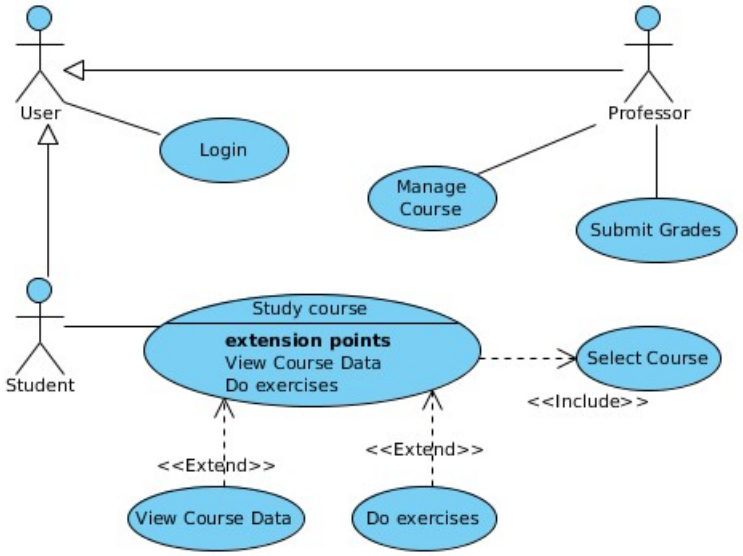
(b) Asociere unidirecțională, numită *preda*, de la clasa `Profesor` la clasa `Materie`; agregare între clasele `Materie`(agregat) și `Test`(componenta), `Materie`(agregat) și `Laborator`(componenta), `Materie`(agregat) și `Curs`(componenta).

(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`.

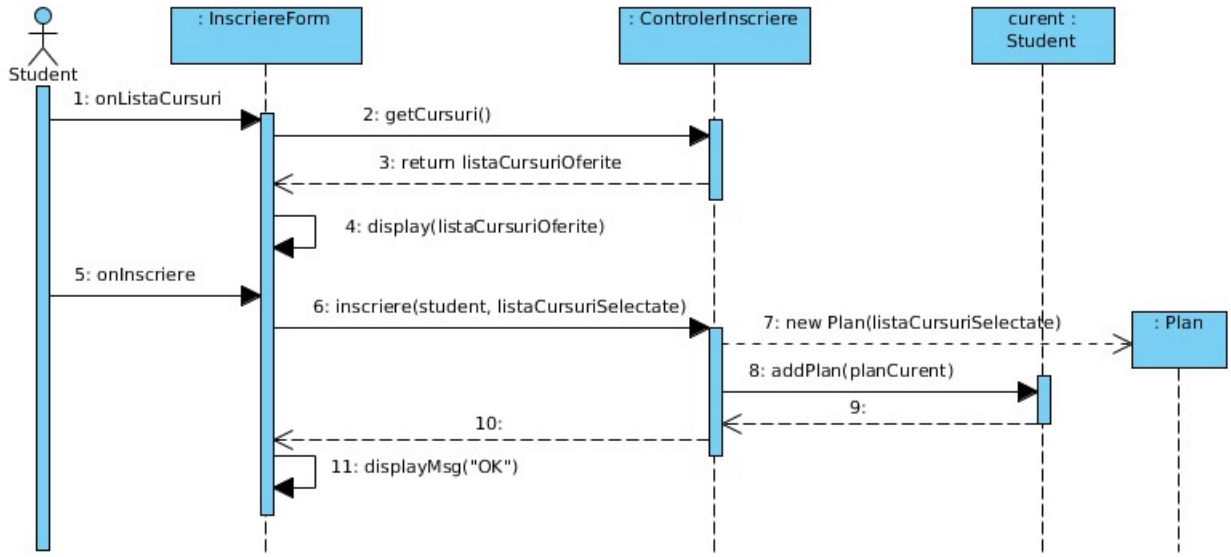
20. Fie următoarea diagramă de cazuri de utilizare.



Selectați 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 `ControlerInscriere` rezultă din aceasta?

(a) `getCursuri()`

(b) `display(listaCursuriOferite)`

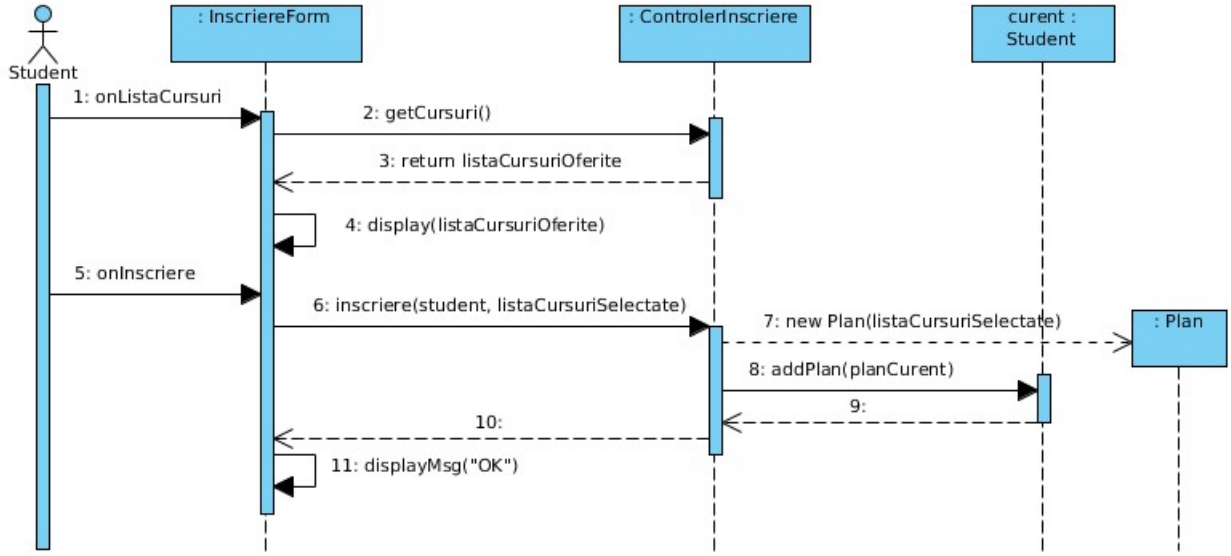
(c) `inscriere(student, listaCursuriSelectate)`

(d) `Plan(listaCursuriSelectate)`

(e) `addPlan(planCurent)`

(f) `displayMsg(OK)`

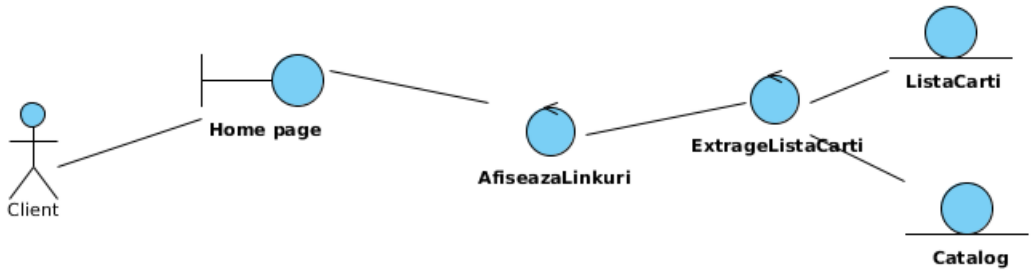
22. Fie următoarea diagramă de secvențe.



Selectați clasele din care sunt instanțiate obiectele implicate în interacțiune:

(a) InscriereForm
(b) ControlerInscriere
(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 clase *boundary*.

(b) Actorii pot comunica cu clase *control*.

(c) Clasele *boundary* pot comunica cu clase *control*.

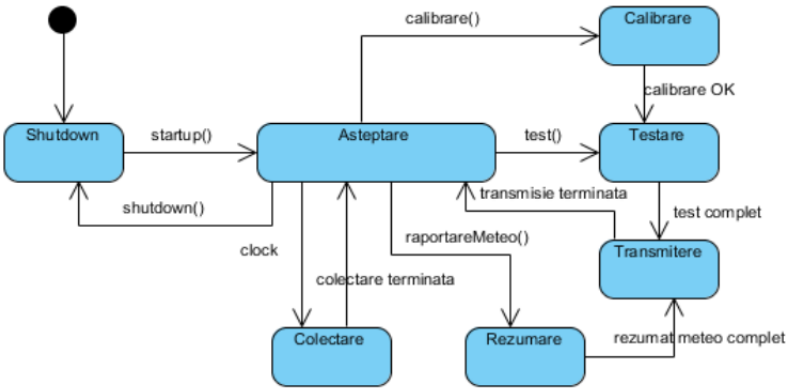
(d) Clasele *boundary* nu pot comunica cu clase *entity*.

(e) Clasele *entity* pot comunica cu clase *control*.

(f) Clasele *entity* pot comunica cu actori.

(g) Clasele *control* nu pot comunica cu alte clase *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 `Asteptare` la starea `Testare`.

(c) `test complet` este eveniment intern.

(d) `test complet` este tranziție.

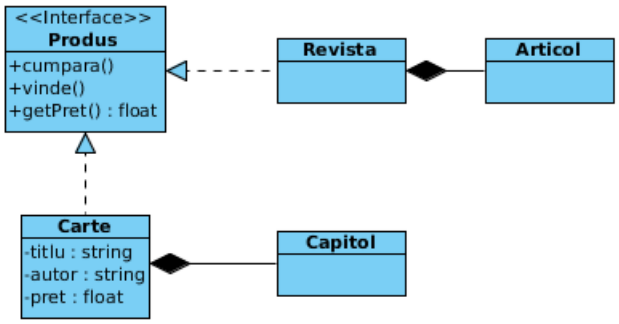
(e) Revenirea în starea `Asteptare` dupa aparitia evenimentului `calibrare()` se face trecând doar prin stările `Calibrare` și `Testare`.

(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 `Asteptare` dupa aparitia 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{...}`

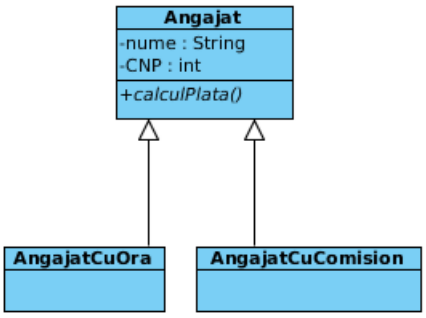
(f) `private float pret(){...}`

(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`?

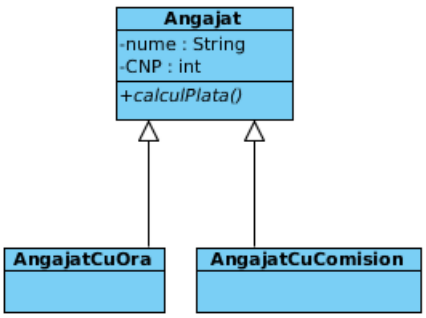
(a)
```
class Angajat {
        private String nume;
        private int CNP;
        public abstract void calculPlata();
         ...}
```

(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{...}`

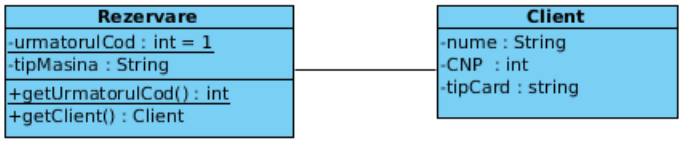
(b) `private int CNP;`

(c) `class AngajatCuOra implements Angajat{...}`

(d) `public calculPlata();`

(e) `public calculPlata(){};`

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



(a) `class Rezervare extends Client{...}`

(b) `private int urmatorulCod = 1;`

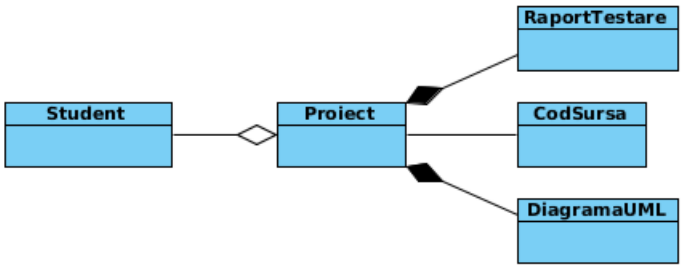
(c) `private static int urmatorulCod = 1;`

(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 `DiagrameUML` definesc compoziții de obiecte de tip `Proiect`.

(b) Clasa `Proiect` definește compoziții de obiecte de tip `Student`, de tip `DiagrameUML` și de tip `RaportTestare`.

(c) Clasa `Proiect` definește un agregat de obiecte de tip `Student` și compoziții de obiecte de tip `DiagrameUML` și de tip `RaportTestare`.

(d) Clasa `Proiect` definește o compoziție de obiecte de tip `Student` și agregate de obiecte de tip `DiagrameUML` și de tip `RaportTestare`.

(e) Clasa `Proiect` este în relație de asociere cu clasa `CodSursa`.

5 Răspunsuri

Limbajul C

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

Limbajul C++

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

Limbajul Java

1. 1d
2. 2a,2d
3. 3a,3b,3d,3e
4. 4b
5. 5a,5b,5c,5d
6. 6c
7. 7b,7d
8. 8b,8d
9. 9c
10. 10a,10e
11. 11c
12. 12b,12e
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,28b,28e
29. 29c,29d,29e
30. 30c,30e