



**Examen POO – 7 februarie 2024**

1p din oficiu

1. (0,5p) Definiți conceptul de **încapsulare** în Programarea Orientată pe Obiecte. Dați un exemplu simplu în JAVA.
2. (0,5p) Utilizând procedeul de **despachetare** dați un exemplu de secțiune de cod JAVA care reține într-o variabilă de tip primitiv valoarea întreagă împachetată dintr-un obiect instanță a unei clase existente în JAVA.
3. (0,5p) Definiți în JAVA o **clasă** proprie din care **nu pot fi create instanțe de obiecte**.
4. (0,5p) Explicați ce se întâmplă în urma compilării și execuției următorului program JAVA:

```
public class A {  
    public static String x = "Examen";  
    public void method() {  
        x = "Frumos";  
    }  
}  
  
public class B extends A {  
    public String getX() {  
        return x;  
    }  
    public static void main(String[] args) {  
        A obj = new B();  
        obj.method();  
        System.out.println(((B)obj).getX());  
    }  
}
```

5. (0,5p) Dați un exemplu în JAVA de folosire a unei **colecții** în care să rețineți o mulțime de numere reale, astfel încât să nu poată exista valori duplicate, iar inserarea/căutarea elementelor în colecție să se facă cât mai eficient posibil.
6. (0,5p) Explicați ce se întâmplă în urma compilării și execuției următorului program JAVA:

```
public interface A {  
    public int method() throws Exception;  
}  
  
public class B implements A {  
    private int method() throws Exception {  
        return 10;  
    }  
    public static void main(String[] args) {  
        B obj = new B();  
        try {  
            System.out.println(obj.method());  
        } catch (Throwable e) {}  
    }  
}
```

7. (0,5p) Definiți conceptul de **testare regresivă** și precizați cum poate fi implementată în JAVA.

8. (6x0,25p=1,5p) Scrieți pe 6 rânduri separate rezultatele afișate la ieșirea standard obținute în urma execuției următorului program JAVA:

```
public class A {
    protected static int x = init(1);
    String y;

    public A(String msg) {
        y = msg + "3";
    }

    public static int init(int y) {
        return 2*x+3*y;
    }

    public String info() {
        return "ABC";
    }

    public float operatie(byte x) {
        return x/3.f;
    }

    public float operatie(int x) {
        return x/4.f;
    }

    public boolean equals(A obj) {
        return y == obj.y;
    }
}
```

```
public class B extends A{
    static {
        x = init(2);
    }

    public B() {
        super("2019");
        y += "2";
    }

    public String info() {
        return "XYZ";
    }

    public static void main(String[] args) {
        System.out.println(A.x);
        A a = new B();
        System.out.println(a.y);
        System.out.println(a.info());
        System.out.println(a.operatie(27));
        A r = new A(new String("Examen"));
        A s = new A(new String("Examen"));
        System.out.println(r.equals(s));
        A[][] t = new A[25][4];
        System.out.println(t.length);
    }
}
```

9. (4p) O aeronavă de pasageri are o anumită capacitate (rânduri x coloane), având locurile dispuse sub forma unei matrice. O persoană are un bilet cu numărul rândului și coloanei unde ar trebui să stea în aeronavă și are asupra sa exact un document de călătorie (carte de identitate sau pașaport). Pe orice document de călătorie este scris numele și CNP-ul persoanei. În plus, pe o carte de identitate este reținută și adresa completă a persoanei, iar pe un pașaport doar județul și culoarea ochilor.

Respectând paradigma programării orientată pe obiecte, identificați entitățile, relațiile dintre acestea, desenați diagrama de clase UML detaliată (0,5p) și implementați în JAVA un program care să rezolve următoarele cerințe:

- (0,5p) Crearea unei aeronave de o anumită capacitate (rânduri x coloane), care nu are în acel moment niciun pasager
- (0,75p) Crearea de persoane care urmează să călătorească prin citirea tuturor informațiilor (detalii bilet și detalii document călătorie) de la intrarea standard
- (0,75p) Îmbarcarea unei persoane în aeronavă: persoana va fi plasată în aeronavă la rândul și coloana specificate pe bilet; în cazul în care locul respectiv este deja ocupat în aeronavă persoana nu va putea fi îmbarcată și se va arunca o excepție verificată
- (0,5p) Determinarea numărului de persoane din aeronavă care călătoresc având asupra lor un pașaport din județul "Cluj"
- (0,5p) Afișarea, în ordine alfabetică a numelui, fiecărei persoane împreună cu locul (rând/coloană) unde călătorește în aeronavă
- (0,5p) Scrieți o clasă cu o metodă *main* în care creați mai întâi o aeronavă, iar apoi executați toate cerințele specificate mai sus astfel încât să utilizați fiecare funcționalitate cel puțin o dată