UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ - DEPARTAMENTUL DE INFORMATICĂ
DOMENIUL INFORMATICĂ - AN II IF
VARIANTA 1

EXAMEN LA DISCIPLINA "PROGRAMARE AVANSATĂ PE OBIECTE ÎN JAVA"
– SESIUNEA IUNIE 2025 –

I. Pentru fiecare dintre cele 5 întrebări de mai jos, indicați varianta de răspuns pe care o considerați corectă:

1. Fie următorul program Java:

```
class Automobil {
    private String marca;
    public Automobil(String marca) { this.marca = marca; }
    public int hashCode() { return marca.length() / 2; }
    public boolean equals(Object obj) {
        return marca.charAt(0) != ((Automobil)obj).marca.charAt(0); }
}

public class Test {
    public static void main(String[] args) {
        HashSet<Automobil> ma = new HashSet<>();
        ma.add(new Automobil("Dacia")); ma.add(new Automobil("Bugatti"));
        ma.add(new Automobil("Ford"));  ma.add(new Automobil("Borgward"));
        ma.add(new Automobil("Fiat"));  ma.add(new Automobil("Delage"));
        System.out.println(ma.size());
    }
}
```

După executarea programului, va fi afișată valoarea:
a) 1 b) 2 c) 3 d) 4

2. Considerăm următorul program Java:

```
class Sir{
     private String sir;
     public Sir(String sir) { this.sir = sir; }
     public void modificaSir(String sir) { this.sir = sir; }
     public void modificaSir(Sir sir){ sir = new Sir("Mihai"); }
     public String getSir(){ return sir; }
 }

 public class Test {
     public static void main(String[] args) {
         Sir s = new Sir("Alex");
         Sir t = new Sir("Ion");
         s.modificaSir("Matei");
         t.modificaSir(new Sir("Dan"));
         s.modificaSir(t);
         System.out.println(s.getSir() + " " + t.getSir());
     }
 }
```

După executarea programului, se va afișa:
a) Matei Alex b) Ion Alex c) Alex Dan d) Matei Ion

3. Considerăm următorul program Java:

```
public class Test {
    public static void main(String args[])
                                        throws FileNotFoundException {
        Scanner fin = new Scanner(new File("exemplu.txt"));
        int s = 0;
        try {
            while(fin.hasNext()) {
                int x = fin.nextInt();
                s = s + x;
            }
        }
        catch(InputMismatchException e) { System.out.println("Număr incorect!");}
        finally { System.out.println(s); }
    }
}
```

Dacă în fișierul text *exemplu.txt* există cel puțin două numere scrise incorect (de exemplu: 2a17, a217, abc, 2-17 etc.), atunci programul:

a) va afișa suma numerelor din fișier aflate înaintea primului număr scris incorect, după care programul se va termina forțat din cauza unei excepții
b) va afișa o singură dată mesajul `Număr incorect!` și suma numerelor din fișier aflate înaintea primului număr scris incorect, după care programul se va termina normal
c) va afișa câte un mesaj `Număr incorect!` pentru fiecare număr din fișier scris incorect, apoi va afișa suma tuturor numerelor din fișier scrise corect, după care programul se va termina normal
d) va afișa doar mesajul `Număr incorect!`, după care programul se va termina forțat din cauza unei excepții

4. Considerăm următorul program Java:

```
class A { public void print() { System.out.print("A "); } }

class B extends A { }

class C extends B { public void print() { System.out.print("B "); } }

public class Test {
    public static void main(String args[]) {
        A p = new B();
        B q = new C();
        A r = new C();
        p.print();
        q.print();
        r.print();
    }
}
```

După executarea programului, se va afișa:
a) A B A b) A B B c) B B B d) B B A

5. Ştiind faptul că şirul de caractere s conţine doar paranteze rotunde, deschise sau închise, ce se va afişa pe ecran după executarea secvenţei de mai jos?

```
int n = s.length();
int p = s.indexOf("()");
while (p != -1) {
        s = s.substring(0, p) + s.substring(p + 2);
        n -= 2;
        p = s.indexOf("()");
}
System.out.println(n);
```

a) numărul minim de paranteze închise sau deschise care trebuie inserate în şirul s astfel încât toate parantezele din şir să se închidă corect
b) numărul perechilor de paranteze care se închid corect în şirul s
c) lungimea maximă a unei secvenţe de forma ()()...() din şirul s
d) lungimea maximă a unei secvenţe de forma (((...))) din şirul s

II. Se consideră definită complet o clasă `Imobil` cu datele membre `tipImobil`, `nrCamere`, `suprafaţă` şi `localitate`. Clasa este utilizată pentru a memora informaţii despre imobile gestionate de o agenţie imobiliară. Datele membre `tipImobil` şi `localitate` sunt de tip `String`, `nrCamere` este de tip `int`, iar suprafaţa de tip `double` . Clasa încapsulează constructori, metode de tip `set/get` pentru toate datele membre, precum şi metodele `toString()`, `equals()` şi `hashCode()`. Creaţi o lista care să conţină cel puţin 3 obiecte de tip `Imobil` şi, folosind stream-uri bazate pe lista creată şi lambda expresii, rezolvaţi următoarele cerinţe:
   a) afişaţi toate imobilele aflate în Bucureşti, sortate crescător după numărul de camere;
   b) afişaţi lista localităţilor distincte în care agenţia gestionează imobile;
   c) creaţi o listă care să conţină imobilele care au suprafaţa cuprinsă între 50 m^2 si 100 m^2;
   d) să se afişeze numărul de imobile care au două camere din fiecare localitate.

III. Informaţiile despre imobilele gestionate de către o agenţie imobiliară sunt păstrate în fişiere text. Fiecare linie dintr-un astfel de fişier conţine informaţii referitoare la un imobil, sub forma `tipImobil,nrCamere, suprafata,localitate.` Scrieţi o clasă Java care să calculeze, pe baza informaţiilor dintr-un fişier de tipul precizat anterior, câte imobile gestionează agenţia într-o anumită localitate, folosind un fir de executare dedicat. Scrieţi un program care, utilizând clasa definită anterior, citeşte de la tastatură denumirea unei localităţi, după care afişează numărul total de imobile gestionate de agenţie în localitatea respectivă, pe baza informaţiilor din fişierele text `imobile_1.txt` şi `imobile_2.txt`.

IV. Se consideră definită complet clasa mutabilă `Adresa` care permite memorarea unei adrese din România: *stradă, număr, bloc, scară, etaj, apartament, judeţ, localitate* şi *cod_poştal*. Definiţi o clasă imutabilă `Imobil` care să permită memorarea următoarelor informaţii despre un imobil: *adresă* (referinţă spre un obiect de tip `Adresa`), *numărul de locatari* (număr natural) şi *locatari* (listă cu elemente de tip `StringBuilder` care conţine numele locatarilor).

NOTĂ:

- Datele de intrare se consideră corecte.
- Nu se vor trata excepţiile.
- Punctaj: 2.5p. (5 x 0.5p.) + 2.5p. + 2p. + 2p. + 1p. (din oficiu)