

COLOCVIU LA DISCIPLINA "PROGRAMARE AVANSATĂ PE OBIECTE" - SESIUNEA IUNIE 2024 -

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

1. Fie următorul program Java:

```
public class Examen {  
    public static void main(String[] args) throws FileNotFoundException {  
        for (int i = 1; i <= 5; i++)  
            try (Scanner sc = new Scanner(new File("file" + i + ".txt"))) {  
                System.out.print(sc.nextInt());  
            } catch (FileNotFoundException e) {  
                System.out.print("!");  
                throw e;  
            } catch (InputMismatchException e) {  
                System.out.print("?");  
            } finally {  
                System.out.print("F");  
            }  
        }  
    }  
}
```

Presupunem că în directorul unde se execută programul există doar următoarele fișiere text:

file1.txt	vid
file2.txt	conține doar numărul 2
file3.txt	vid
file4.txt	conține doar numărul 4

Precizați care dintre următoarele afirmații referitoare la executarea programului dat sunt adevărate:

- a) se va afișa ?F2F?F4F?F și executarea se va termina cu succes
- b) se va afișa ?F2F?F4F!F și executarea se va termina cu succes
- c) se va afișa ?F2F?F4F!F și executarea se va termina cu o excepție
- d) se va afișa ?2?4!F și executarea se va termina cu o excepție

2. Fie următorul cod Java:

```
class A {  
    Integer intA;  
    public A(Integer i) { intA = i; }  
    public void set(Integer i) { intA = i; }  
    public A(A other) { intA = other.intA; }  
}  
  
class B {  
    A obiect;  
    public B(A init) { obiect = new A(init); }  
    public B(B other) { obiect = other.obiect; }  
    public A getA() { return obiect; }  
}
```

```
class C {
    B obiect;
    public C(B init) { obiect = init; }
    public B getB() { return new B(obiect); }
}
```

Câte dintre cele 3 clase definite mai sus sunt imutabile?

- a) 3 b) 2 c) 1 d) 0

3. Fie următorul program:

```
class A {
    String sir;
    public A(String sir) { this.sir = sir; }
    public boolean equals(Object obj) {
        return sir.substring(1) == ((A)obj).sir.substring(1);
    }
    public int hashCode() { return (int)sir.charAt(0); }
    public String toString() { return sir + " "; }
}

public class Test {
    public static void main(String[] args) throws IOException {
        LinkedHashSet<A> lhs = new LinkedHashSet<>();
        lhs.add(new A("examen"));
        lhs.add(new A("Java"));
        lhs.add(new A("Examen"));
        lhs.add(new A("PAO"));
        lhs.add(new A("Java"));
        lhs.forEach(System.out::print);
    }
}
```

După executarea sa, se va afișa:

- a) examen Java Examen PAO Java
 b) examen Java PAO
 c) Examen PAO Java
 d) examen Java Examen PAO

4. Fie următorul program Java:

```
class A {
    int x = 10;
    public A(int x) { this.x = x; }
    int f(int t) { return x + t; }
}

class B extends A {
    int x = 30;
    public B() { super(20); }
    int f(int t) { return t + super.f(10*x); }
}
```

```
public class Test {
    public static void main(String[] args) {
        A ob = new B();
        System.out.println(ob.f(ob.x));
    }
}
```

După executarea programului, se va afișa:

- a) 320 b) 330 c) 340 d) 350

5. Considerăm următorul cod Java:

```
interface Functie { Function<Integer, Double> f = x -> 1.0/x; }

class A implements Functie{ public double f(int x) { return 2.0 / x; }; }

class Test {
    static void afisare(Functie f, int t) {
        System.out.println(f.f.apply(t));
        System.out.println(f.f(t));
    }

    public static void main(String[] args) {
        afisare(null, 10);
        afisare(new A().f, 20);
        afisare(null, (int)new A().f(1));
        afisare(new Functie() {}, 10);
        afisare(x -> 3.0 / x, 20);
    }
}
```

Precizați câte erori de compilare se vor semnala în clasa Test:

- a) 2 b) 3 c) 4 d) 5

II. Se consideră definită complet clasa *Imobil* având datele membre *tip*, *localitate*, *nrCamere*, *suprafata* și *pret*. Clasa este utilizată pentru a memora informații despre imobilele gestionate de o agenție imobiliară. Datele membre *tip* și *localitate* sunt de tip *String*, data membră *nrCamere* este de tip *int*, iar datele membre *suprafata* și *pret* sunt de tip *double*. Clasa încapsulează constructori, metode de tip *set/get* pentru toate datele membre, precum și metodele *toString()*, *equals()* și *hashCode()*. Creați o listă care să conțină cel puțin 3 obiecte de tip *Imobil* și, folosind stream-uri bazate pe lista creată și lambda expresii, rezolvați următoarele cerințe:

- afișați imobilele care au cel puțin 3 camere și nu costă mai mult de 500000 RON, în ordinea crescătoare a suprafețelor lor;
- afișați localitățile distincte;
- creați o listă care să conțină imobilele aflate în București cu prețul cuprins între 300000 RON și 500000 RON;
- afișați pentru fiecare localitate distinctă imobilele aflate în localitatea respectivă.

III. Informațiile despre imobilele gestionate de către lanțul agențiilor imobiliare *RentSale* sunt păstrate în mai multe fișiere text. Fiecare linie dintr-un astfel de fișier conține informații referitoare la un imobil, respectiv *tip*, *localitate*, *nrCamere*, *suprafata*, *pret*. Scrieți o clasă Java care să calculeze, pe baza informațiilor dintr-un fișier de tipul indicat anterior, numărul imobilelor care au o suprafață mai mare decât o valoare

dată, precum și un număr minim de camere, folosind un fir de executare dedicat. Scrieți un program care, utilizând clasa definită anterior, citește de la tastatură o valoare reală s_{min} și un număr natural c_{min} , după care afișează numărul total al imobilelor având suprafețele mai mari decât s_{min} și cel puțin c_{min} camere existente în două agenții, pe baza informațiilor din fișierele text `AgentieRS_1.txt` și `AgentieRS_2.txt`.

- IV. Se consideră definită complet clasa `Persoana` care permite memorarea următoarelor informații despre o persoană: *nume* (șir de caractere), *vârsta* (număr natural) și *salariul mediu anual* (număr real). Definiți complet o clasă singleton denumită `CitireInformatiiPersoane` care să permită citirea informațiilor despre mai multe persoane dintr-un fișier text de tip CSV, respectiv informațiile despre o persoană sunt scrise pe o linie, despărțite între ele prin câte o virgulă. Informațiile despre persoane se vor citi din fișierul text într-un obiect de tip `ArrayList<Persoana>`.

NOTĂ:

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