UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
DEPARTAMENTUL DE INFORMATICĂ
DOMENIUL DE STUDII INFORMATICĂ – SERIA 25

# COLOCVIU LA DISCIPLINA "PROGRAMARE AVANSATĂ PE OBIECTE" – SESIUNEA IUNIE 2024 –

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

1. Fie următorul cod Java:

```
class A {
    public A met1() { return new A(); }
    final void met2() { }
    public void met3() { }
    public void met4() { }
    int met5(int i) { return 5; }
}

class B extends A {
    public B met1() { return new B(); }
    public void met2() { }
    void met3() { }
    public static void met4() { }
    int met5() { return 5; }
}
```

Câte suprascrieri (overriding), supraîncărcări (overloading), respectiv erori conține codul de mai sus?
a) două supraîncărcări, trei erori
b) două suprascrieri, o supraîncărcare, două erori
c) o suprascriere, două supraîncărcări, două erori
d) o suprascriere, o supraîncărcare, trei erori

2. Fie următorul program:

```
class A {
    int val;
    public A(int val) { this.val = val; }
    public boolean equals(Object obj) { return val != ((A)obj).val; }
    public int hashCode() { return val / 100; }
}

public class Test {
    public static void main(String[] args) throws IOException {
        HashSet<A> hs = new HashSet();
        hs.add(new A(100));
        hs.add(new A(10));
        hs.add(new A(1));
        hs.add(new A(10));
        hs.add(new A(100));
        System.out.println(hs.size());
    }
}
```

După executarea sa, se va afișa:
a) 2 b) 3 c) 4 d) 5

3. Fie următorul program Java:

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

class B extends A {
    int x = 30;
    int f(int t) { return x + 10*t; }
}

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) 20 b) 130 c) 310 d) 330

4. Fie următorul program Java:

```
class Test {
    static String sir = "E";

    void A() {
        try {
            sir = sir + "A";
            B();
        } catch (Exception e) { sir = sir + "B"; }
    }

    void B() throws Exception {
        try {
            sir = sir + "C";
            C();
        } catch (Exception e) { throw new Exception(); }
        finally { sir = sir + "D"; }
    }

    void C() throws Exception { throw new Exception(); }

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

După executarea programului, se va afișa:
a) EACDB b) EADCB c) EBCAD d) EACD

5. Considerăm următorul cod Java:

```
interface Functie {
    int f(double x);
    default boolean eval(double x) { return f(x) == x; }
}

public class Test {
    Functie rf;
    public Test(Functie rf) { this.rf = rf; }
    public boolean eval(double x) { return rf.f(x) == x; }
}
```

De asemenea, considerăm și următoarele afirmații referitoare la codul Java de mai sus:
a) interfața `Functie` este o interfață funcțională;
b) metoda `eval` din clasa `Test` rescrie metoda implicită `eval` din interfața `Functie`;
c) expresia `boolean b = new Test(x->10).eval(10);` este corectă;
d) expresia `boolean b = new Test(x->x/100).eval(10);` este corectă;
e) expresia `boolean b = new Functie() { public int f(double x) { return (int)(x/10); } }.eval(10);` este corectă;
f) în expresia `boolean b = new Functie() { public int f(double x) { return (int)(x/10); } }.eval(10);` se va apela metoda implicită `eval` din interfața `Functie`.

Precizați câte dintre cele 6 afirmații de mai sus sunt adevărate:
a) 2 b) 3 c) 4 d) 5

II. Se consideră definită o clasă `TenisATP` cu datele membre `numeJucator`, `naționalitateJucator`, `pozitieATP`, `puncteATP`, și `nrTurneeATP`, utilizată pentru a memora informații un jucător de tenis ATP. Datele membre `numeJucator`, `naționalitateJucator` sunt de tip `String`, iar `pozitieATP`, `puncteATP` și `nrTurneeATP` sunt de tip `int`. Clasa încapsulează constructori, metode de tip `set/get` pentru toate datele membre, precum și metodele `toString()`, `equals()` și `hashCode()`. Creați o lista care să conțină cel puțin 3 obiecte de tip `TenisATP` și, folosind stream-uri bazate pe lista creată și lambda expresii, rezolvați următoarele cerințe:

- afișați jucătorii de tenis care au care participat la cel puțin 10 turnee ATP, sortați alfabetic după nume;
- afișați lista naționalităților distincte ale jucătorilor ATP;
- creați o colecție care să conțină jucătorii care au acumulat intre 1000 si 4000 de puncte;
- pentru fiecare naționalitate, să se afișeze numărul jucătorilor având naționalitatea respectivă.

III. Informațiile despre jucătorii de tenis ATP sunt păstrate în fișiere text. Fiecare linie dintr-un astfel de fișier conține informații referitoare la un jucător, sub forma `numeJucator,naționalitateJucator,pozitieATP,puncteATP,nrTurneeATP`. Scrieți o clasă Java care să calculeze, pe baza informațiilor dintr-un fișier de tipul precizat anterior, câți jucători au o naționalitate dată, folosind un fir de executare dedicat. Scrieți un program care, utilizând clasa definită anterior, citește de la tastatură o naționalitate, după care afișează numărul total de jucători având naționalitatea respectivă, pe baza informațiilor din fișierele text `jucatori_1.txt` și `jucatori_2.txt`.

IV. Se consideră definită complet clasa mutabilă `Adresa` care permite memorarea unei adrese din România: *stradă*, *număr*, *bloc*, *scară*, *etaj*, *apartament*, *județ* și *localitate*. Definiți o clasă imutabilă `Facultate` care să permită memorarea următoarelor informații despre o facultate: *denumirea facultății* (șir de caractere), *numărul de studenți* (număr natural), *specializări* (listă cu elemente de tip `String`) și *adresa* (referință spre un obiect de tip `Adresa`).