RESTANȚĂ LA DISCIPLINA "PROGRAMARE AVANSATĂ PE OBIECTE" – SESIUNEA SEPTEMBRIE 2017 –

I. Pentru fiecare dintre cele 5 întrebări de mai jos, indicați variantele de răspuns pe care le considerați corect

1. Fie următorul program Java:

```
class C {
    public static int a = 1;
    public int b = 1;

    public void met() {
        a++;
        b++;
    }
}

public class Test {
    public static void main(String[] args) {
        C ob1 = new C();
        C ob2 = new C();
        ob1.met();
        ob2.met();
        System.out.println(ob1.a + ob2.b);
    }
}
```

După executarea programului, va fi afișată valoarea:

a) 4 b) 3 c) 6 d) 5

2. Fie următorul program Java:

```
class C {
    public static void met() throws Exception {
        try {
            throw new Exception();
        }
        finally {
            System.out.print("M");
        }
    }
}

public class Test {
    public static void main(String args[]) {
        try {
            C.met();
        }
        catch (Exception e) {
            System.out.print("C");
        }
        finally {
            System.out.print("F");
        }
        System.out.println("T");
    }
}
```

După executarea programului, se va afişa:

a) MCFT b) CMFT c) CFMT d) MCT

3. După executarea secvenţei de cod

```
String s = "abracadabra";
s.replace('r', 'R');
int p = s.indexOf("R");
int q = s.lastIndexOf("b");
s = s.substring(0, q - p);
System.out.println(s.length());
```

se va afişa:

a) 10 b) 6 c) 9 d) 7

4. Fie următorul program Java:

```
class B {
    static String x = "";
    public static void met_1() {
        x = x + "A";
    }
    public void met_2() {
        x = x + "B";
    }
}

class C extends B {
    public static void met_1() {
        x = x + "C";
    }
    public void met_2() {
        x = x + "D";
    }
}

public class Test {
    public static void main(String[] args) {
        B ob = new C();
        ob.met_1();
        ob.met_2();
        System.out.println(ob.x);
    }
}
```

După executarea programului, se va afişa:

a) AB b) CD c) CB d) AD

5. Considerăm următoarele afirmaţii:

I. o clasă abstractă poate fi instanţiată
II. o clasă abstractă trebuie să conţină cel puţin o metodă abstractă
III. o clasă abstractă nu poate fi declarată ca fiind de tip `final`
IV. o clasă abstractă poate să aibă constructori

Precizaţi care dintre afirmaţiile de mai sus sunt false:

a) II, III, IV b) III, IV c) II, IV d) I, II, III

II. Se consideră definită o clasă `Asigurare` având datele membre `tip`, `titular`, `valoare` și `localitate`. Clasa este utilizată pentru a memora informații despre asigurările încheiate de o societate de asigurări. Datele membre `tip`, `titular` și `localitate` sunt de tip `String`, iar data membră `valoare` este de tip `double`. Clasa încapsulează constructor cu argumente, 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 `Asigurare` și, folosind stream-uri bazate pe lista creată și lambda expresii, rezolvați următoarele cerințe:

- afișați asigurările de tip RCA, în ordinea descrescătoare a valorilor lor;
- afișați localitățile distincte în care societatea a încheiat asigurări;
- creați o colecție care să conțină asigurările încheiate de societate în București, având valoarea cuprinsă între 10000 RON și 50000 RON;
- afișați pentru fiecare persoană (titular) o listă a asigurărilor încheiate la societatea respectivă.

III. Informațiile despre asigurările încheiate de sucursalele societății de asigurări `SafeLife` sunt păstrate în mai multe fișiere text. Fiecare linie dintr-un astfel de fișier conține informații referitoare la o asigurare, respectiv `tip`, `titular`, `valoare` și `localitate`, despărțite prin virgule. Scrieți o clasă Java care să calculeze, pe baza informațiilor dintr-un fișier de tipul indicat anterior, valoarea totală a asigurărilor de un anumit tip, folosind un fir de executare dedicat. Scrieți un program care, utilizând clasa definită anterior, citește de la tastatură un șir de caractere reprezentând tipul unei asigurări (RCA, CASCO etc.), după care afișează valoarea totală a asigurărilor de tipul respectiv încheiate de două sucursale ale societății, pe baza informațiilor din fișierele text `sucursalaSafeLife_1.txt` și `sucursalaSafeLife_2.txt`.

IV. Se consideră baza de date `SafeLife`, având următorul URL: *jdbc:derby://localhost:1527/SafeLife*. Baza de date conține tabela `Asigurări`, care păstrează informații despre asigurările încheiate de societatea de asigurări `SafeLife`. Tabela `Asigurări` are câmpurile `tip`, `titular`, `valoare` și `localitate`. Scrieți un servlet care să preia printr-un parametru (transmis folosind metoda GET) denumirea unei localități, iar apoi să genereze o pagină HTML care să conțină informații despre asigurările încheiate de societate în localitatea respectivă sau un mesaj corespunzător în cazul în care societatea nu a încheiat nicio asigurare în localitatea dată.

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)
- Metode ale clasei abstracte `HttpServlet`:
  - `protected void doGet(HttpServletRequest request, HttpServletResponse response)`
  - `protected void doPost(HttpServletRequest request, HttpServletResponse response)`
  - `public void init()`
  - `public void destroy()`