EXAMEN LA DISCIPLINA "PROGRAMARE AVANSATĂ PE OBIECTE"
– SESIUNEA MAI/IUNIE 2017 –

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

1. Fie următorul program Java:

```
class ABC{
    public static int x;
    public int y;

    public ABC() { met(); }

    public void met(){ x = x + 3; y = y + 1; }
}
public class Test{
    public static void main(String[] args){
        ABC t = new ABC();
        ABC u = new ABC();
        System.out.println(t.x + u.y);
    }
}
```

După executarea programului, va fi afișată valoarea:

a) 4 b) 5 c) 7 d) 8

2. Fie următorul program Java:

```
class Test{
    String str = "a";

    void A(){
        try {
            str += "b"; B();
        } catch (Exception e){ str += "c"; }
    }
    void B() throws Exception{
        try{
            str += "d"; C();
        } catch (Exception e){ throw new Exception();}
        finally{ str += "e"; }
        str += "f";
    }
    void C() throws Exception { throw new Exception(); }

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

abdefc

După executarea programului, va fi afișată valoarea:

a) abdec b) abdefc c) abdef d) abcdef

3. După executarea secvenței de cod

```
String s = new String("Examen");
if (s == "Examen") System.out.print("A");
else System.out.print("B");
if (s.equals("Examen")) System.out.print("C");
else System.out.print("D");
```

se va afișa:

a) AD b) BC c) AC d) BD

4. Fie următorul program Java:

```
class A{
    public int x = 1;
    void afisare() { System.out.println(x); }
}
class B extends A{
    public int x = 2;
    void afisare() { System.out.println(x); }
}
public class Test{
    public static void main(String[] args){
        A ob = new B();
        System.out.println(++ob.x);
    }
}
```

După executarea programului, se va afișa:

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

5. Fie următorul program Java:

```
class Tablou{
    static void met(int[] a, int b){
        int aux;
        aux = a[0]; a[0] = b; b = aux;
    }
}
public class Test{
    public static void main(String[] args){
        int a[] = {1, 2, 3, 4, 5}, b = 6;
        Tablou.met(a, b);
        int s = b;
        for (int i = 0; i < a.length; i++) s = s + a[i];
        System.out.println(s);
    }
}
```

După executarea programului, se va afișa:

a) 15 b) c) 21 d) 26

II. Se consideră definită o clasă `Log` având datele membre `user`, `date`, `login_time` și `logout_time`. Clasa este utilizată pentru a memora informații despre conectările la un server efectuate de către mai mulți utilizatori. Datele membre `user` și `date` sunt de tip `String`, iar `login_time` și `logout_time` sunt de tip `long` (numărul de secunde trecute de la ora 00:00 a zilei respective). Clasa încapsulează 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 `Log` și, folosind stream-uri bazate pe lista creată și lambda expresii, rezolvați următoarele cerințe:

- afișați utilizatorii care au fost conectați la server cel puțin o oră;
- afișați datele distincte în care s-a conectat cel puțin un utilizator;
- creați o colecție care să conțină utilizatorii care s-au conectat la server în ziua de 01.01.2017, până la ora 12:00, în ordine alfabetică;
- afișați pentru fiecare utilizator zilele în care acesta s-a conectat la server.

III. Informațiile despre conectările efectuate de către mai mulți utilizatori la un anumit server sunt păstrate în fișiere text de tip `log`. Fiecare linie dintr-un astfel de fișier conține informații referitoare la o conectare la un server, sub forma `user,date,login_time,logout_time`. Scrieți o clasă Java care să calculeze, pe baza informațiilor dintr-un fișier de tip `log`, de câte ori s-a conectat într-o zi un utilizator la un server, folosind un fir de executare dedicat. Scrieți un program care, utilizând clasa definită anterior, citește de la tastatură numele unui utilizator și o dată, după care afișează numărul total de conectări ale utilizatorului respectiv la două servere, pe baza informațiilor din fișierele `log` cu numele `server_1.log` și `server_2.log`. Datele calendaristice sunt șiruri de forma `zz.ll.aa`, iar orele sunt șiruri de forma `hh:mm:ss`.

IV. Se consideră baza de date `Angajati`, având următorul URL: *jdbc:derby://localhost:1527/Angajati*. Baza de date conține tabela `Logs`, care păstrează informații despre conectările efectuate de către mai mulți utilizatori la baza de date. Tabela `Logs` are câmpurile `user`, `date`, `login_time` și `logout_time`. Scrieți un servlet care să preia prin 2 parametri (transmiși folosind metoda POST) numele unui utilizator și o dată calendaristică, după care să generează o pagină HTML care să conțină intervalele orare din ziua respectivă în care acel utilizator s-a conectat la baza de date sau un mesaj corespunzător în cazul în care utilizatorul nu s-a conectat niciodată în ziua respectivă.

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()`
- Clasa `ResultSet` conține metodele `getDate` și `getTime`.