

## 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 corecte:

1. Fie următorul program Java:

```
class C { public static int a=1; }  
public class teste_grila {  
    public static void main(String[] args) {  
        C ob1 = new C();  
        C ob2 = new C();  
        ob1.a++;  
        System.out.println(++ob2.a);  
    }  
}
```

După executarea programului, va fi afișată valoarea:

- a) 2      b) 3      c) 1      d) nicio valoare, deoarece programul este incorect  
sintactic și nu va putea fi executat

2. Considerăm următoarea metodă:

```
void test(){  
    try{  
        met();  
    }  
    catch (NullPointerException ex){  
        System.out.print("NPE ");  
    }  
    catch (Exception ex){  
        System.out.print("EX ");  
    }  
    finally{  
        System.out.print("FIN ");  
    }  
    System.out.println("END");  
}
```

După apelarea metodei test(), ce se va afișa dacă metoda met() va lansa excepția IllegalArgumentException?

- a) NPE FIN END                      b) EX END  
c) NPE EX FIN END                  d) EX FIN END

**3. După executarea secvenței de cod**

```
String s = "abcbcd";  
String t = "Programare";  
int p = t.indexOf(s.charAt(0));  
t = t.substring(0, p) + t.substring(p+1);  
System.out.println(t);
```

se va afișa:

- a) Progrmare      b) rogramare      c) Programre      d) Progmre

**4. Fie următorul program Java:**

```
class A {  
    public static int f(int x) { return x+1; }  
    public int g(int x) { return x+2; }  
}  
  
class B extends A {  
    public static int f(int x) { return x+4; }  
    public int g(int x) { return x+3; }  
}  
  
public class Test {  
    public static void main(String[] args) {  
        A a = new B();  
        System.out.println(a.f(1) + a.g(3));  
    }  
}
```

După executarea programului, se va afișa:

- a) 7      b) 11      c) 8      d) 10

**5. Fie următorul program Java:**

```
class A {  
    String sir = "";  
    public A(String sir) { this.sir = this.sir + sir + "A"; }  
}  
  
class B extends A {  
    public B(String sir) { super(sir); this.sir = this.sir + sir + "B"; }  
}  
  
class C extends B {  
    public C(String sir) { super(sir); this.sir = this.sir + sir + "C"; }  
}  
  
public class Test {  
    public static void main(String[] args) {  
        System.out.println(new C("D").sir);  
    }  
}
```

După executarea programului, se va afișa:

- a) DCDBDA      b) DCBA      c) DADBDC      d) DADABDABC

**II.** Se consideră definită o clasă `Automobil` având datele membre `marca`, `model`, `capacitate` și `pret`. Clasa încapsulează 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 `Automobil` și, folosind `stream-uri` bazate pe lista creată și `lambda expresii`, rezolvați următoarele cerințe:

- afișați automobilele care costă cel puțin 5000€, în ordinea descrescătoare a prețurilor;
- afișați mărcile distincte de automobile;
- creați o listă formată din automobilele care au capacitatea cilindrică cuprinsă între 2000 și 3000 cm<sup>3</sup>;
- afișați pentru fiecare marcă modelele existente.

**III.** Scrieți o clasă Java care să calculeze de câte ori apare un cuvânt dat într-un fișier text, folosind un fir de executare. Scrieți un program care citește de la tastatură un cuvânt și, utilizând clasa definită anterior, afișează numărul total al aparițiilor cuvântului respectiv în fișierele text *exemplu\_1.txt*, *exemplu\_2.txt* și *exemplu\_3.txt*. Cuvintele din fișierele text de intrare sunt despărțite între ele prin spații și semnele de punctuație uzuale.

**IV.** Se consideră baza de date *Angajati*, având următorul URL: *jdbc:derby://localhost:1527/Angajati*. Baza de date conține tabela *DateAngajati*, având câmpurile *CNP*, *Nume*, *Varsta* și *Salariu*. Scrieți un servlet care să preia valoarea unui parametru *min* (transmis folosind metoda GET) și generează o pagină HTML care să conțină informațiile despre angajații cu salariul mai mare sau egal decât *min*.

**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 `HttpServletRequest`:
  - **protected void** `doGet(HttpServletRequest request, HttpServletResponse response)` **throws** `ServletException, IOException;`
  - **protected void** `doPost(HttpServletRequest request, HttpServletResponse response)` **throws** `ServletException, IOException;`
  - **public void** `init()` **throws** `ServletException;`
  - **public void** `destroy();`