UNIVERSITATEA DIN BUCUREŞTI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
DEPARTAMENTUL DE INFORMATICĂ
DOMENIUL DE STUDII INFORMATICĂ
SERIILE 23, 24 ŞI 25

EXAMEN LA DISCIPLINA "ELEMENTE AVANSATE DE PROGRAMARE"
– SESIUNEA MAI/IUNIE 2022 –

I. Pentru fiecare dintre cele 5 întrebări de mai jos, indicaţi variantele de răspuns pe care le consideraţi 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 afişată valoarea:

a) 2 **b)** 3 c) 1 d) nicio valoare, deoarece programul este incorect sintactic şi 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 afişa dacă metoda `met()` va lansa excepţia `IllegalArgumentException`?

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

3. 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));
    }
}
a.f(1) -> static = dupa tipul declarat (superclasa) 2
a.g(3) ->istanta = dupa tipul real (subcalsa) 6
```

După executarea programului, se va afișa:

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

4. Fie următorul program Java:

```
class Persoana implements Serializable {
    String nume;
    int varsta;

    public Persoana(String nume, int varsta) {
        this.nume = nume;
        this.varsta = varsta;
        System.out.println("Constructor");
    }
}
public class Test {
    public static void main(String[] args) throws Exception {
        ObjectOutputStream oos = new ObjectOutputStream(
                                 new FileOutputStream("persoana.ser"));
        Persoana p = new Persoana("Popescu Ion", 40), q = p;
        oos.writeObject(q);
        oos.close();
..............................
        ObjectInputStream ois = new ObjectInputStream(
                                new FileInputStream("persoana.ser"));
        Persoana r = (Persoana)ois.readObject();
        ois.close();
    }
}
```

De câte ori va fi afișat mesajul *Constructor*, după executarea programului dat?
e) niciodată f) o dată g) de două ori h) de trei ori

5. Fie următorul program Java:

```
class Automobil{
    private String marca;
    public Automobil(String marca) { this.marca = marca; }
    public int hashCode() { return 0; }
    public boolean equals(Object obj) { return false; }
}

public class Test {
    public static void main(String[] args) {
        HashSet<Automobil> la = new HashSet<>();
        la.add(new Automobil("Audi"));
        la.add(new Automobil("BMW"));
        la.add(new Automobil("Audi"));
        la.add(new Automobil("Opel"));
        System.out.println(la.size());
    }
}
```

După executarea programului, va fi afișată valoarea:
e) 4 f) 3 g) 2 **h) 1**

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 lista 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;

  la.stream().filter(ob->ob.getpret>=5000).sorted(Comprator.comparing(Aoto::getPret).reversed()).forEach()

- afișați mărcile distincte de automobile;

  map
- creați o listă formată din automobilele care au capacitatea cilindrică cuprinsă între 2000 și 3000 cm^3;
  collect()
- afișați pentru fiecare marcă modelele existente. - >groupBY (marca), mappare (modele)

III. Scrieţi o clasă Java care să calculeze de câte ori apare un cuvânt dat într-un fişier text, folosind un fir de executare. Scrieţi un program care citeşte de la tastatură un cuvânt şi, utilizând clasa definită anterior, afişează numărul total al apariţiilor cuvântului respectiv în fişierele text *exemplu_1.txt*, *exemplu_2.txt* şi *exemplu_3.txt*. Cuvintele din fişierele text de intrare sunt despărţite între ele prin spaţii şi semnele de punctuaţie uzuale.

```
class Fir exteds Thread
{
    String path;
    String cuvantul;
    int frecventa;

Fir(String path;
    String cuvantul)
)
{

}

   void run()
{
     Scanner fin = new Scanner(new File(this.path));
     String linie;

while(fin.hasNextLine())
{
   linie = fin.nextLine();
  String cuv[] = linie.split(„[ ,.!';]+”);

     for(String item : cuv)
     if(this.cuvantul.equls(item))
                this.frecventa++;
}
}

getFrecventa(){retun this.frecventa;}

}


Fir fir1 = new Fir(path, cuv)
Fir fir2 = new Fir(path, cuv)
Fir fir3 = new Fir(path, cuv)

Fir1.start();
Fir1.join();
Fir2.start();
Fir2.join();
```

```
Fir3.start();
Fir3.join();


int total = fir1.getNrAp + fir2.getNrAp + fir3.getNrAp;

}
```

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 program care să citească de la tastatură o valoare *min* și afișează informațiile despre angajații având salariul mai mare sau egal decât *min* sau un mesaj corespunzător dacă nu există niciun astfel de angajat.

Statement, PreparedStatement
ResultSet -> next

Clasa imutabila (curs 4), Clasa utilitara (curs 2), Clasa sing (curs 3)

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)