Examen "Programare avansată pe obiecte" - seria 23

Restanță seriile 33 și 34

...

Puncte: 85/90

1. Numele complet: *

Majeri Constantin-Gabriel

2. Grupa: *

232

3. Dacă o clasă B extinde clasa abstractă A și clasa C extinde clasa B, care dintre instanțierile următoare este corectă? *
(5/5 puncte)

- (●) A ab = new B(); ✓
- ( ) C cb = new B();
- ( ) B ba = new A();
- ( ) C ca = new A();

4. Precizați care dintre următoarele afirmații sunt adevărate pentru o colecție de tip HashMap: *
(5/5 puncte)

- [ ] nu se poate asocia aceeași valoare mai multor chei
- [x] este permisă utilizarea valorii null atât pentru cheie, cât și pentru valoare ✓
- [x] perechile nu pot fi stocate într-o ordine a cheilor sau a valorilor dată de o funcție comparator ✓
- [ ] se menține ordinea de inserare a perechilor

5. Știind faptul că șirul de caractere s conține doar paranteze rotunde, deschise sau închise, ce se va afișa pe ecran după executarea secvenței de mai jos?

```
int n = s.length();
int p = s.indexOf("()");
while (p != -1)
{
        s = s.substring(0, p) + s.substring(p + 2);
        n -= 2;
        p = s.indexOf("()");
}

System.out.println(n);
```

*
(5/5 puncte)

- ( ) numărul perechilor de paranteze de forma () din șirul s
- ( ) lungimea maximă a unei secvențe de forma ()()...() din șirul s
- ( ) lungimea maximă a unei secvențe de forma (((...))) din șirul s
- (•) numărul minim de paranteze închise sau deschise care trebuie inserate în șirul s astfel încât toate parantezele din șir să se închidă corect ✓

6. Crearea unei clase singleton trebuie să respecte următoarele reguli: *
(5/5 puncte)

- [x] constructorii clasei sunt privați ✓

- [ ] clasa trebuie să conțină doar date membre non statice
- [x] referința obiectului este memorată printr-o dată membră statică ✓
- [ ] constructorii clasei sunt publici

7. Care dintre următoarele expresii poate fi utilizată pentru a prelua valoarea câmpului Salariu de tip float dintr-un obiect rst de tip ResultSet? * (5/5 puncte)

- ( ) float s = rst.getField("Salariu", "float");
- (•) float s = rst.getFloat("Salariu"); ✓
- ( ) float s = rst.getField("Salariu", Float);
- ( ) float s = rst.get("Salariu", Float);

8. Ce se va afișa în urma executării secvenței de cod de mai jos?

```
int x = 0;
if (Double.isInfinite(2/x))
        System.out.println("Infinit");
else
        System.out.println("2/0"); *
```

(5/5 puncte)

- ( ) eroare la compilare din cauza împărțirii la 0
- (•) eroare la executare din cauza împărțirii la 0 ✓
- ( ) programul este corect și va afișa Infinit
- ( ) programul este corect și va afișa NaN

✕

9. Se consideră următorul cod Java:

```
public class Fir implements Runnable
{
      void run()  { ... }
}
```

Care dintre următoarele linii de cod conduce la pornirea unui fir de executare? * (5/5 puncte)

- new Runnable(Fir).start()
- new Thread(Fir).start()
- new Thread(new Fir).start()
- new Fir().start() ✓

✗

10. Dacă se apelează o metodă statică sincronizată pentru un obiect, atunci: * (5/5 puncte)

- alte fire nu mai pot apela, pentru nici un alt obiect al clasei respective, nicio altă metodă sincronizată ✓
- atunci alte fire nu mai pot apela, pentru același obiect, nicio altă metodă sincronizată
- o metodă statică nu se poate sincroniza
- alte fire mai pot apela, pentru alte obiect al clasei respective, alte metodă sincronizate

11. Care dintre următoarele metode verifică dacă într-un obiect de tip ResultSet mai există tupluri care nu au fost încă prelucrate (accesate)? * (5/5 puncte)

- hasNext
- afterLast
- next ✓

- last

12. Precizați care dintre următoarele afirmații nu sunt adevărate pentru mecanismul de serializare: *
(5/5 puncte)

- [x] datele membre private nu sunt serializate ✓
- [x] serializarea include datele membre statice ✓
- [ ] dacă un obiect care trebuie serializat conține referințe către obiecte neserializabile, atunci va fi generată o excepție
- [ ] datele membre de tip transient nu sunt serializate

13. Precizați care dintre următoarele afirmații sunt adevărate pentru un bloc try – catch: *
(5/5 puncte)

- [x] blocurile catch se exclud reciproc, respectiv o excepție nu poate fi tratată de mai multe blocuri catch ✓
- [ ] blocul finally nu poate să lipsească
- [ ] ordinea în care blocurile catch sunt implementate nu contează
- [x] blocul finally nu are parametrii ✓

14. Care dintre următoarele variante reprezintă o supraîncărcare corectă pentru metoda protected int getNota(String curs)? *
(5/5 puncte)

- protected int getNota (String curs) throws IOException
- private int getNota (String curs)
- protected long getNota (String curs)
- (●) public long getNota (int IDStudent) ✓

15. Fie lp o listă de tipul ArrayList<Persoana> și următoarea secvență de instrucțiuni (considerăm definită complet clasa Persoana cu datele membre nume, vârsta și salariu):

```
BiFunction<Persoana, Double, Double> f = (p, m) -> p.getSalariu()*(1 + m/100.0);
BiPredicate<Double, Double> p = (s, m) -> s <= m;
BiConsumer<Persoana, Double> c = (t, m) -> t.setSalariu(t.getSalariu()*(1 + m/100.0));

for(Persoana pc : lp)
        .....
            if (p.test(f.apply(pc, 50), 5000) {
                c.accept(pc, 50);
            }
```

Elementele listei lp trebuie prelucrate astfel: se va mări salariul unei persoane cu 50% doar în cazul în care noul său salariu nu ar depăși 5000 RON. Pentru a efectua această prelucrare, punctele de suspensie de mai sus trebuie înlocuite cu: *
(5/5 puncte)

- (●) if(p.test(f.apply(pc,50.0),5000.0)) c.accept(pc,50.0); ✓
- ( ) f.apply(pers,50.0).andThen(p.test(50.0,5000.0)).andThen(c.accept(pers,50.0));
- ( ) if(p.test(f.apply(pc,5000.0),50.0)) c.accept(pc,50.0);
- ( ) if(f.apply(pc,50.0).test(pc, 50.0)) c.accept(pc,50.0);

16. Precizați care dintre următoarele afirmații referitoare la un stream asociat unei colecții sunt adevărate: *
(5/5 puncte)

- [ ] un stream stochează elementele unei colecției în scopul de a le prelucra
- [x] un stream nu este reutilizabil, respectiv poate fi prelucrat o singură dată ✓
- [ ] pentru același stream se pot efectua mai multe operații intermediare și mai multe operații de închidere
- [x] un stream poate să returneze un alt stream ✓

17. Pentru asigurarea excluderii reciproce se utilizează cuvântul cheie synchronized la nivel de: *
(5/5 puncte)

- [ ] clasă
- [x] metodă ✓
- [x] bloc de instrucțiuni ✓
- [ ] dată membră

✗

18. Fie următorul program Java:

```
class Fir implements Runnable{
    int x;

    public Fir(int x){ this.x = x; }

    public void run(){
        new Thread(() -> {System.out.print(x);}).start();
    }

    public static void main(String args[]) throws InterruptedException{
        Fir obj1 = new Fir(1);
        Fir obj2 = new Fir(2);
        Thread t1 = new Thread(obj1);
        Thread t2 = new Thread(obj2);
        t1.start();
        t2.start();
        t1.join();
        t2.join();
        System.out.print(3);
    }
}
```

După executarea programului, poate fi afișat unul dintre numerele: * (0/5 puncte)

- (•) 123 sau 213
- ( ) 312 sau 321
- ( ) 123, 132, 213 sau 231
- ( ) 123, 132, 213, 231, 312 sau 321 ✓

19. Precizați de câte ori se realizează mecanismul de suprascriere și de câte ori mecanismul de supraîncărcare în următoarele clase:

```
class Angajat {
    int salariu;
    void afiseazaSalariu() {
        System.out.println(this.salariu);
    }
}

class Economist extends Angajat {
    int spor;
    void afiseazaSalariu() {  suprascriere
        System.out.println(this.salariu + this.spor);
    }

    void afiseazaSalariu(int bonus) {  supraincarcare
        System.out.println(this.salariu + this.spor + bonus);
    }

    int afiseazaSalariu(String mesaj) {  supraincarcare
        int aux = this.salariu + this.spor;
        System.out.println(mesaj + aux);
        return aux;
    }
} *
```

(5/5 puncte)

- o suprascriere și o supraîncărcare
- două suprascrieri și o supraîncărcare
- o suprascriere și două supraîncărcări ✓
- trei suprascrieri și o supraîncărcare

20. Dacă după introducerea într-un browser a adresei http://localhost:8080/WebApplication1/Persoana?varsta=30 (http://localhost:8080/WebApplication1/Persoana?varsta=30) se obține ca răspuns o pagină validă și știm faptul că Persoana este un servlet, atunci în cadrul servlet-ului sigur este implementată metoda: *
(5/5 puncte)

- doPost

- (•) doGet ✓
- ( ) init
- ( ) destroy