Subiect POO 2006-2007 tip A. Se da urmatorul cod Java

```
1  public class Masina {
2     private int pasageri;
3     protected Masina inlocuita;
4     protected static int numarExemplare=0;
5
6     public Masina(int pas, Masina inl) {
7       this.pasageri = pas;
8       this.inlocuita = inl;
9       numarExemplare++;
10    }
11    public Masina(int pas) {
12      this(pas, null);
13    }
14    public boolean inlocuiesteMasina() {
15      return (inlocuita!=null);
16    }
17    public boolean equals(Object o) {
18      if ((o!=null)&&(o instanceof Masina)){
19        Masina m = (Masina) o;
20        if (!inlocuiesteMasina())
21          return (m.inlocuita==null);
22        return (pasageri==m.pasageri);
23      }
24      return false;
25    }
26 }
```

```
1  public class Motocicleta {
2     private int pasageri;
3     protected String model;
4     protected static int numarExemplare=0;
5
6     public Motocicleta(String mod, int pas){
7       this.pasageri = pas;
8       this.model = mod;
9       numarExemplare++;
10    }
11    public Motocicleta(String mod) {
12      this(mod, 1);
13    }
14    public boolean equals(Object o) {
15      if (o instanceof Motocicleta){
16        Motocicleta m = (Motocicleta) o;
17        if (modelulESpecificat())
18          return (pasageri==m.pasageri);
19        return (m.model==null);
20      }
21      return false;
22    }
23    public boolean modelulESpecificat() {
24      return (model!=null);
25    }
26                                          }
```

Refacere partial:

1. **Comentati liniile de cod 2, 6, 11, 12, 14, 17, 18, 19, 20, 21** (insistand pe ceea ce este special in fiecare linie)**. (10 pct)**

2. **Adaugati clasei `Masina` un camp** numit **`anFabricatie`** de tip **`Integer`**. **(1 pct)**

3. **Modificati constructorii clasei `Masina`** astfel incat **sa initializeze si noul camp**. **(2+2 pct)**

4. **Modificati metoda equals() a clasei `Masina`** astfel incat **comparatia sa priveasca si noul camp**. **(2 pct)**

5. **Adaugati clasei `Masina` o metoda care sa returneze valoarea campului `anFabricatie` (doar referinta). Cum trebuie modificata metoda pentru a returna o copie a obiectului in loc de referinta la obiect? (2+1 pct)**

Examen final:

1. **Creati o clasa `Vehicul` care sa contina elementele comune claselor `Masina` si `Motocicleta`**, clasa care sa fie extinsa prin mostenire de cele doua clase. Ce anume se modifica in codurile claselor **`Masina` si `Motocicleta`** in acest caz? (5 pct.)

2. Creati o clasa numita **`MotocicletaCuAtas`** care extinde clasa **`Motocicleta`** prin adaugarea unui camp de tip **`boolean`** numit **`atasMontat`**. Clasa va avea **doi constructori** care vor initializa si noul camp si vor reutiliza codul constructorilor clasei **`Motocicleta`**. (5 pct.)

3. Creati o clasa numita **`UtilizareVehicul`** si codul metodei principale a acestei clase, care:
- creaza doua obiecte ale clasei **`Masina`, `ms1`** si **`ms2`** utilizand in fiecare caz alt constructor, (2 pct)
- creaza doua obiecte ale clasei **`Motocicleta`, `mt1`** si **`mt2`** utilizand in fiecare caz alt constructor, (2 pct)
- compara obiectele **`ms1`** cu **`ms2`** (cum trebuie ele initializate pentru ca rezultatul sa fie **`true`**?) (3 pct.)
- creaza doua referinte ale clasei **`Vehicul`** numite **`veh1`** si **`veh2`**. (1 pct.)

4. **Explicati care dintre urmatoarele linii de cod vor avea probleme la compilare sau executie** si **care este cauza in fiecare caz**:

a) (3 pct.)

```
1       veh1 = ms1;
2       ms2 = veh1;
3       veh2 = mt1;
4       veh1 = null;
5       veh2 = veh1;
6
```

b) (3 pct.)

```
1       boolean x;
2
3       veh1 = ms1;
4       ms2 = (Masina) veh1;
5       x = ms2.inlocuiesteMasina();
6       mt1 = (Motocicleta) veh1;
```

c) (3 pct.)

```
1       boolean x;
2
3       veh1 = (Masina) ms1;
4       veh2 = (Motocicleta) mt2;
5       x = veh1.equals(veh2);
6       x = veh1.inlocuiesteMasina();
7       x = mt1.equals(mt2);
```

d) (3 pct.)

```
1       boolean x;
2       MotocicletaCuAtas mt3 = new
3             MotocicletaCuAtas("A1", true);
4       veh1 = (Motocicleta) mt3;
5       veh2 = (MotocicletaCuAtas) mt2;
6       x = veh1.equals(ms2);
7       x = veh1.modelulESpecificat();
```

```
 1  public class UtilizareVehicul {
 2
 3    public static void main(String[] args){
 4
 5        Masina  ms1 = new Masina(4, null);
 6        Masina  ms2 = new Masina(5);
 7
 8        Motocicleta  mt1 = new Motocicleta("x", 2);
 9        Motocicleta  mt2 = new Motocicleta("y");
10
11        Vehicul veh1, veh2;
12
13        boolean x;
14
15  /*
16        // (a)
17        veh1 = ms1; // OK
18
19        // ms2 = veh1;
20        // produce eroarea de compilare
21        //     incomp. types - found Vehicul but expected Masina
22
23        veh2 = mt1; // OK
24        veh1 = null; // OK
25        veh2 = veh1; // OK
26  */
27
28
29  /*
30        // (b)
31        veh1 = ms1;
32        ms2 = (Masina) veh1; // OK
33        x = ms2.inlocuiesteMasina(); // OK
34
35        // mt1 = (Motocicleta) veh1;
36        // produce exceptia la executie
37        //     java.lang.ClassCastException: Masina
38  */
39
40
41  /*
42        // (c)
43        veh1 = (Masina) ms1; // OK
44        veh2 = (Motocicleta) mt2; // OK
45        x = veh1.equals(veh2); // OK
46
47        // x = veh1.inlocuiesteMasina();
48        // produce eroarea de compilare
49        // cannot find symbol - method inlocuiesteMasina()
50
51        x = mt1.equals(mt2); // OK
52  */
53
54
55  /*
56        // (d)
57        MotocicletaCuAtas mt3 = new MotocicletaCuAtas("A1", true); // OK
58        veh1 = (Motocicleta) mt3; // OK
59
60        // veh2 = (MotocicletaCuAtas) mt2;
61        // produce exceptia la executie
62        //     java.lang.ClassCastException: Motocicleta
63
64        x = veh1.equals(ms2); // OK
65
66        // x = veh1.modelulESpecificat();
67        // produce eroarea de compilare
68        // cannot find symbol - method modelulESpecificat()
69  */
70
71    }
72
73  }
```