TEHNICI AVANSATE DE PROGRAMARE

LUCRARE DE LABORATOR 3

Clase şi obiecte în Java. Crearea obiectelor. Constructori. Variabile clasă. Metode statice. Moştenirea

I. SCOPUL LUCRĂRII

Lucrarea de față are rolul de a prezenta şi familiariza studentul cu noțiuni privind obiectele şi clasele în limbajului Java: clase şi crearea obiectelor, rolul constructorilor, semnificația şi utilizarea variabilelor clasă, metodele clasă, subclasele şi moştenirea; cu importanța şi situațiile de utilizare a acestora.

La sfârşitul acestei lucrări, studentul va avea posibilitatea să scrie programe Java în care să folosească noțiunile menționate anterior.

II. NOȚIUNI TEORETICE

Observație: Vezi noțiunile prezentate în cursul 2.

1. Crearea obiectelor

- Să considerăm următorul exemplu:

```
public class Vehicul
{
      public String categoria;
      public String marca;
      public int nrRoti;
      public boolean aerCond;
      public boolean volan;

      public Vehicul(String nume,boolean aerCond) {
        categoria="automobil";
        marca=nume;
        nrRoti=4;
        this.aerCond=aerCond;
        volan=true;
      }
      public Vehicul(String nume) {
        categoria="bicicleta";
        marca=nume;
        nrRoti=2;
        aerCond=false;
        volan=false;
      }
      public void afisare() {
         System.out.println("Categorie vehicul "
                            +categoria);
          System.out.println("Denumire "+categoria+": "+marca);
```

```
            System.out.println("Nr roti: "+nrRoti);
            System.out.print("Aer conditionat: ");
            if(aerCond)
                        System.out.println("da");
            else
                        System.out.println("nu");
            System.out.print("Volan: ");
            if(volan)
                        System.out.println("da");
            else
                        System.out.println("nu");
            System.out.print("\n");
      }

      public static void main(String args[ ]) {

            Vehicul b1,m1,m2;
            b1=new Vehicul("Pegas");
            m1=new Vehicul("Dacia",false);
            m2=new Vehicul("Cielo",true);
            b1.afisare();
            m1.afisare();
            m2.afisare();
      }
}
```

Observații:

1. ***Un fişier sursă java care conține o clasă publică trebuie salvat cu numele clasei respective la care se adaugă extensia*** *.java.*
2. ***Într-un fişier sursă java poate exista cel mult o clasă declarată publică.*** Astfel programul de mai sus se va salva într-un fişier cu numele **Vehicul.java**. Altfel, la compilare se va semnala eroare.
3. Clasa **Vehicul** conține doi constructori. Utilizarea mai multor constructori pentru aceeaşi clasă se recomandă atunci când se urmărește inițializarea obiectelor clasei în moduri diferite. Astfel, în **main( )** se vor crea obiecte (vehicule) ce reprezintă biciclete şi automobile, în funcție de utilizarea unui constructor sau al altuia.
4. Unul din contextele în care utilizarea notației bazate pe **this** (**this** este referința către obiectul curent) este necesară, este atunci când se definește un parametru sau o variabilă locală cu acelaşi nume ca o dată membru a clasei. Astfel, pentru a se face diferența între variabila locală sau parametru şi data membru a clasei, aceasta din urmă este accesată explicit prin referința **this (**vezi primul constructor al clasei **Vehicul**: `this.aerCond=aerCond`**).**
5. Într-un constructor se poate referi ca primă instrucțiune un constructor al clasei curente. Este singura poziție din definiția unui constructor în care poate să apară o referire la un alt constructor. Un astfel de apel apare sub forma:

```
this(listaDeArgumente); // constructor din aceeasi clasa
```

unde **listaDeArgumente** reprezintă lista valorilor parametrilor constructorului invocat.

- Exemplul 2:

```
public class Floare
{
      public String denumire;
      public int nrPetale;
      public String zonaClima="-";

      public Floare(String denum,int nrP) {
        denumire=denum;
        nrPetale=nrP;
      }
      public Floare(String denum,int nrP,String clima) {
        this(denum,nrP);
        zonaClima=clima;
      }
      public void afisare() {
          System.out.println("Denumire: "+denumire);
          System.out.println("Numar petale : "+nrPetale);
          System.out.println("Zona climatica: "+zonaClima+"\n");
      }

      public static void main(String args[ ]) {

          Floare floare1=new Floare("narcisa",7);
          Floare floare2=new Floare("orhidee",30,"tropicala");
          floare1.afisare();
          floare2.afisare();
      }
}
```

Observații:

1. Variabilele membru se pot inițializa când sunt declarate. Se recomandă ca inițializarea să se facă în constructor (în principal, pentru ca fiecare obiect al clasei să poată avea valori diferite pentru datele membru).
2. În cazul în care o dată membru membru este inițializată la declarare iar apoi şi în constructor, atunci valoarea sa va fi cea pe care o primeşte în constructor. Dacă în constructor nu mai primeşte nici o valoare, atunci valoarea sa va rămâne cea de la declarare. Astfel, obiectul **floare1** va avea pentru data membru **zonaClima** valoarea "-" deoarece în primul constructor (cel care este utilizat pentru crearea acestui obiect), aceasta dată membru nu primeşte nici o valoare. Obiectul **floare2**, în schimb, este creat cu ajutorul celui de-al doilea constructor, care inițialiazează data membru **zonaClima** la valoarea celui de-al treilea parametru al sau – care este "tropicala".

2. Variabile clasă

Să considerăm clasa **Vehicul** prezentată în secțiunea anterioară. O vom modifica adăugându-i o dată membru statică numită **contor** care numără obiectele clasei ce sunt create în program. Această nouă variabilă, fiind declarată **static**, este asociată clasei, ci nu instanțelor clasei. O variabilă clasă poate fi accesată prin numele clasei, deci nu este nevoie de crearea unei instanțe a clasei doar pentru a o accesa. O variabilă statică este creată o singură dată, la încărcarea clasei.

```
public class Vehicul
{
      static int contor=0;
      public String categoria;
      public String marca;
      public int nrRoti;
      public boolean aerCond;
      public boolean volan;

      public Vehicul(String nume,boolean aerCond)
      {
        categoria="automobil";
        marca=nume;
        nrRoti=4;
        this.aerCond=aerCond;
        volan=true;
        contor++;
      }
      public Vehicul(String nume)
      {
        categoria="bicicleta";
        marca=nume;
        nrRoti=2;
        aerCond=false;
        volan=false;
        contor++;
      }
      public void afisare()
      {
          System.out.println("Categorie vehicul "+contor+": "
                              +categoria);
          System.out.println("Denumire "+categoria+": "+marca);
          System.out.println("Nr roti: "+nrRoti);
          System.out.print("Aer conditionat: ");
          if(aerCond)
                    System.out.println("da");
          else
                    System.out.println("nu");
          System.out.print("Volan: ");
          if(volan)
                    System.out.println("da");
          else
                    System.out.println("nu");
          System.out.print("\n");
      }

      public static void main(String args[ ]) {

          Vehicul b1,m1,m2;
          b1=new Vehicul("Pegas");
          b1.afisare();
          m1=new Vehicul("Dacia",false);
          m1.afisare();
          m2=new Vehicul("Cielo",true);
          m2.afisare();
          System.out.println("\nExista "+Vehicul.contor+" obiecte
                              ale clasei Vehicul");
      }
}
```

Să considerăm un alt exemplu:

```
class Salariat
{
      public static final double impozit=0.15;
      public String nume;
      public String functia;
      public int varsta;
      public double salariuBrut;

      public Salariat(String num,String func,int ani,double sal)
      {
        nume=num;
        functia=func;
        varsta=ani;
        salariuBrut=sal;
      }
      public double salariuNet()
      {
          return salariuBrut*(1-impozit);
      }
      public void afisare()
      {
       System.out.println("Nume salariat: "+nume);
       System.out.println("Functia: "+functia);
       System.out.println("Varsta: "+varsta);
       System.out.println("SalariuBrut: "+salariuBrut+" lei");
       System.out.println("SalariuNet: "+salariuNet()+" lei \n");
      }
}


public class UtilSal
{
  public static void main(String args[ ])
  {
    Salariat s1=new Salariat("Popa Ion","contabil",36,7300000);
    Salariat s2=new Salariat("Rada Alina","secretara",28,6000000);
    System.out.println("Impozit: "+(int)(Salariat.impozit*100)+"% \n");
    s1.afisare();
    s2.afisare();
  }
}
```

Observație:

Am utilizat în clasa **Salariat** o dată membru **impozit** care este declarată şi static şi final, ceea ce înseamnă că este considerată o constantă de către compilator (valoarea sa nu poate fi modificată).

3. Metode clasă (statice)

Să considerăm următorul exemplu:

```
public class Film
{
      static int pretBilet=30000;
```

```
        static String cinematograf="Patria";
        public String nume;
        public int durata; // in minute
        public boolean luatOscar;

        public Film(String num,int nrMin,boolean oscar)
        {
          nume=num;
          durata=nrMin;
          luatOscar=oscar;
        }

        static int getPretBilet()
        {
          return pretBilet;
        }

        static void printCinema(Film f)
        {
          System.out.println("Filmul "+f.nume+" este difuzat la
                              cinematograful "+cinematograf);
        }
        public void afisare()
        {
           System.out.println("Nume: "+nume);
           System.out.println("Durata (in minute): "+durata);
           if(luatOscar)
             System.out.println("Filmul a primit premiul Oscar\n");
           else
             System.out.println("Filmul nu a primit premiul
                                 Oscar\n");
        }

        public static void main(String args[ ]) {
           System.out.println("Pretul unui bilet este: "
                               +Film.getPretBilet()+" lei");
           Film f1=new Film("Titanic",180,true);
           f1.afisare();
           Film.printCinema(f1);
        }
    }
```

Observații:

1. O metodă statică este considerată că aparține clasei şi nu instanțierilor clasei. O metodă statică poate să refere numai variabile sau metode statice (pentru că numai acestea există fără a se fi instanțiat un obiect din clasa respectivă), dar poate să fie apelată din orice metodă a clasei.
2. Metoda **main( )** care reprezintă punctul de plecare pentru orice program Java, este declarată ca fiind statică şi deci poate să fie referită fără instanțierea unui obiect.
3. Limbajul Java permite declararea unei secvențe de cod ca fiind statică în modul următor:

```
        static {
            secventa de cod
    }
```

4. Astfel de secvențe se pot declara numai în afara metodelor. Corespunzător, în momentul în care clasa respectivă ajunge să fie încărcată (pentru că a fost referită) se vor executa toate secvențele de cod declarate în acest mod la nivelul clasei. Să considerăm următoarea aplicație Java:

```
class Floare
{
   int nrPetale;
   static
   {
       System.out.println("floarea");
   }
   public Floare(int nrPetale)
   {
       this.nrPetale=nrPetale;
   }
   public void afiseaza()
   {
       System.out.println(nrPetale);
   }
}
public class Exemplu
{
  static
  {
     Floare f=new Floare(10);
     f.afiseaza();
  }
  public static void main(String args[])
  {
  }
}
```

Programul afişează:

```
Floarea
10
```

deşi în metoda **main( )** nu sunt prevăzute prelucrări, s-au executat secvențele de cod indicate cu atributul static din ambele clase definite.

Să considerăm un alt exemplu:

```
public class Test
{
   int x;
   static int y=0;

   static void modificaY() {
       y+=10;
   }
   void modificaY(int y) {
       this.y=y;
   }
   void modificaX(int x) {
      this.x=x;
   }
   void func() {
      Test t=new Test();
      y=1;
      t.y=1;
      Test.y=1;
```

```
        Test.modificaY();
        t.modificaY(2);
        System.out.println(y);
        x=1;
        t.modificaX(2);
        System.out.println(x);
    }

    public static void main(String args[]) {
         Test t=new Test();
         t.func();
    }
}
```

În metoda **func( )** a clasei **Test** a fost creat un obiect care reprezintă o instanțiere a clasei. Se observă că în timp ce pentru variabila statică **y** există 5 variante pentru atribuirea unei valori (toate referindu-se la variabila globală **y**), pentru variabila **x** cele două modificări ale valorii lui **x** se referă la variabile diferite – în primul caz (**x=1**) este vorba despre data membru **x** a obiectului pentru care se execută metoda, iar în cel de-al doilea caz (**t.modifica(x)**) este vorba despre data membru **x** a obiectului de tip **Test** creat în metoda **func()**.

4. Moştenirea

Să considerăm următorul exemplu (programul Vehicule.java):

```
class Vehicul
{
      public String denumire;
      public int nrRoti;

      public Vehicul(String denumire,int nrRoti)
      {
        this.denumire=denumire;
        this.nrRoti=nrRoti;
      }
      public void afisareVehicul()
      {
          System.out.println("\n");
          System.out.println("Denumire: "+denumire);
          System.out.println("Nr roti: "+nrRoti);
      }
}

class Bicicleta extends Vehicul
{
    public boolean combustibil;

    public Bicicleta(String denumire,int nrRoti)
    {
        super(denumire,nrRoti);
        combustibil=false;
    }
    public void afisareBicicleta()
    {
       afisareVehicul();
       System.out.println("Bicicleta nu consuma benzina");
    }
}
```

```
class Motocicleta extends Bicicleta
{
    public Motocicleta(String denumire,int nrRoti)
    {
        super(denumire,nrRoti);
        combustibil=true;
    }
    public void afisareMotocicleta()
    {
       afisareVehicul();
       System.out.println("Motocicleta consuma combustibil");
    }
}

class Automobil extends Vehicul
{
  public String combustibil;
  public boolean volan;



  public    Automobil(String    denumire,int    nrRoti,String
combustibil)
    {
        super(denumire,nrRoti);
        this.combustibil=combustibil;
        volan=true;
    }
  public void afisareAutomobil()
    {
       afisareVehicul();
       System.out.println("Combustibil: "+combustibil);
       System.out.println("Volan: da");
    }
}

public class Vehicule
{
      public static void main(String args[ ])
      {

          Vehicul v;
          Bicicleta b=new Bicicleta("Pegas",2);
          b.afisareBicicleta();
          Automobil a=new Automobil("BMW",4,"benzina");
          a.afisareAutomobil();
          v=a;
          v.afisareVehicul();
       // v.afisareAutomobil(); eroare
          Motocicleta m=new Motocicleta("Honda",2);
          m.afisareMotocicleta();
          b=m;
//corect          b.afisareBicicleta();
          v=m; //corect
      }
}
```

III. MODUL DE LUCRU

- Clasic:

1. Se editează codul sursă al programului Java folosind un editor de text disponibil (de ex., se poate utiliza Notepad).
2. Se salvează fişierul cu extensia **.java**.
3. Compilarea mini-aplicaţiei Java se va face din linia de comandă:
   javac nume_fişier_sursă.java
   În cazul în care programul conţine erori acestea vor fi semnalate şi afişate.
4. Pentru rularea aplicaţiei Java, se lansează interpretorul Java:
   java nume_clasă_care_conţine_main

- Se foloseşte utilitarul disponibil în laborator J2SDK Net Beans.

IV. TEMĂ

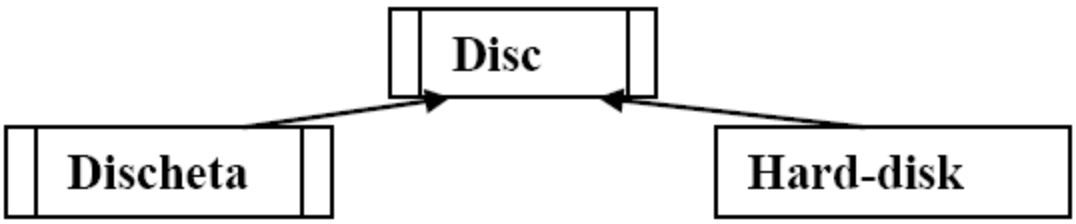
- Se vor parcurge toate exemplele prezentate în platforma de laborator testându-se practic (acolo unde este cazul).

- Modificaţi clasa Salariat astfel încât salariul fiecărui angajat să poată fi exprimat şi în dolari. De asemenea, programul să afişeze şi cursul de schimb al dolarului utilizat în calcule.

- Scrieţi clasa Student. Fiecare student are un nume, an, grupă, şi două note obţinute la o anumită materie - una pe semestrul 1 iar cealaltă pe semestrul 2. Clasa conţine un constructor ce iniţializează datele membru ale clasei la valorile parametrilor săi, o funcţie care calculează şi returnează media celor două note, şi o funcţie de afişare ce afişează valorile datelor membru şi valoarea mediei. Scrieţi un program complet în care să utilizaţi obiecte ale clasei Student.

- Modificaţi clasa **Film** astfel:
  - adăugaţi clasei o metodă statică care să returneze valoarea datei membru **durata**.
  - încercaţi să apelaţi metoda **afisare( )** în cadrul metodei **printCinema( )** şi invers, apelaţi metoda **printCinema( )** în cadrul metodei **afisare( )**.
  Explicaţi la fiecare caz în parte rezultatele obţinute.

- Creaţi următoarea ierarhie de clase:



Un disc are un nume şi o capacitate. O discheta are în plus o stare (1 dacă este “write-protected”, 0 altfel). Un hard-disk are în plus un controler (de tip sir de caractere; exemplu: “IDE”, “SCSI”). Superclasa are un constructor (cu parametri) şi o funcţie de afişare (afişează valorile datelor membru). Clasa discheta are un constructor, o funcţie de afişare şi o funcţie care setează (modifică) starea dischetei. Clasa hard-disk are un constructor şi o funcţie de afişare.
Scrieţi un program Java care lucrează cu obiecte de tipul celor 3 clase.