TEHNOLOGII JAVA
PENTRU DEZVOLTAREA APLICAȚIILOR
LUCRARE DE LABORATOR 6

Clase şi obiecte în Java.
Metode clasă (statice). Moştenirea.

I. SCOPUL LUCRĂRII

Lucrarea de față are rolul de a prezenta şi familiariza studentul cu noțiuni de bază ale limbajului Java: metodele clasă, subclasele şi moştenirea.

La sfârşitul acestei lucrări, studentul va avea posibilitatea să scrie programe Java în care să folosească noile noțiuni.

II. NOȚIUNI TEORETICE

1. Metodele clasă (statice)

Vezi Curs 6, secțiunea 4.3.

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:

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.

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.

Limbajul Java permite declararea unei secvențe de cod ca fiind statică în modul următor:

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

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()**.

2. Moştenirea

Vezi Curs 6, secțiunea 4.5.

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 combustibil");
    }
}
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

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**. Fişierul trebuie salvat la următoarea locaţie: **c:\JBulider7\jdk1.3.1\bin**
3. Compilarea mini-aplicaţiei Java se va face din linia de comandă. Se poate proceda astfel. Se deschide o fereastră MS-Dos: Start ->Run, se tipăreşte **command** în căsuţa de text şi se apasă butonul OK. Printr-o schimbare de directoare şi subdirectoare se trece la locaţia: **c:\JBulider7\jdk1.3.1\bin.** Sau, se lansează WindowsCommander. Se trece la locaţia **c:\JBulider7\jdk1.3.1\bin.** Se deschide o fereastră MS-Dos: Commander ->Run Dos.
4. Pentru compilare, se tipăreşte la prompter **javac nume_fişier_sursă.java** şi se apasă Enter. De ex., dacă fişierul se numeşte Test.java, se va scrie **javac Test.java**. În cazul în care programul conţine erori acestea vor fi semnalate şi afişate în fereastra MS-Dos, după care va apare iar prompter-ul. Dacă programul nu conţine erori şi compilarea se face cu succes, atunci va apare numai prompter-ul.
5. Pentru rularea aplicaţiei Java, se lansează interpretorul Java. Se tipăreşte la prompter următoarea comandă **java nume_clasă_care_conţine_main** şi se apasă Enter. De ex., dacă clasa din program care conţine metoda *main( )* se numeşte Test, se va scrie **java Test**.

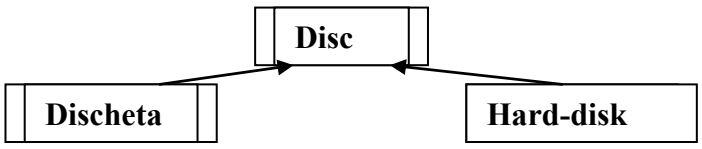
IV. TEMĂ

1. Se vor parcurge toate exemplele prezentate în platforma de laborator testându-se practic.
2. 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.

3. 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.