Lucrarea de laborator nr. 2 - Applet-uri şi aplicații Java care lucrează cu obiecte

Breviar teoretic

Obiecte

Reprezentarea lumii înconjurătoare se realizează în memoria sistemelor de calcul utilizând numere, mai exact şiruri de 1 şi 0. Corespondența între aceste şiruri de biți şi lumea reală sau informațiile complexe care descriu obiecte din lumea reală se realizează prin modele de reprezentare ce au ca elemen constituient programele. Obiectele fizice sau noțiunile din lumea reală trebuie reprezentate în memoria calculatorului în așa fel încât informațiile specifice lor să fie păstrate la un loc şi să se poată prelucra ca un tot unitar.

Din punct de vedere al organizării memoriei interne a sistemelor de calcul, un obiect transpus sub forma numerică trebuie stocat într-o zonă de memorie unic definită, care să înglobeze pe lângă nume obiectului şi principalele caracteristici ale acestuia. Această zonă compactă va fi numită imagine calculator a unui ***obiect***. Se remarcă faptul, că între obiectul fizic şi reprezentarea acestuia sunt deosebiri fundamentale:

Un aspect extrem de simplu este oferit de operația de creare a unui nou obiect, identic cu altul deja existent: în cazul sistemelor de calcul acest lucru se realizează prin copierea zonei de memorie folosite de obiect într-o altă zonă în care am dorit dublarea acestuia. În realitate obținerea chiar şi a unei copii Xerox a unei bancnote sau a unei foi de hârtie cu anumite caractere pe ea este un proces extrem de complicat.

Un obiect fizic sau noțiune, pentru a putea fi reprezentat în calculator, impune o analiză atentă nu numai a proprietăților

acestuia, dar şi a modului în care obiectul poate fi utilizat, a operațiilor care sunt premise a fi realizate cu ajutorul acestuia. Toate aceste elemente vor trebui însă definite prin intermediul funcțiilor şi operațiilor numerice permise de calculator. Deoarece toate obiectele fizice ce sunt modelate prin intermediul sistemului de calcul interacționează cu alte obiecte, modelarea numerică a acestora trebuie să permită şi aceste interacțiuni cu stimuli exteriori sau alte obiecte.

Prin urmare, un obiect este caracterizat în primul rând printr-un identificator apelativ, ce constituie ***numele*** obiectului. Caracterizarea perceptivă a obiectului relevă o serie de proprietăți ale acestuia, proprietăți care poartă numele, în cadrul programării orientate pe obiecte, de ***atribute*** ale obiectului.

Setul de operații specifice unui obiect, împreună cu modul în care acesta reacționează la stimuli exteriori poartă numele de ***comportamentul obiectului***.

Din punctul de vedere al programării, un obiect este o reprezentare în memoria calculatorului a proprietăților şi comportamentului unei noțiuni sau ale unui obiect real.

Încapsularea informațiilor în interiorul obiectelor

Dacă fiecare obiect este caracterizat printr-o serie de proprietăți, în realitate, nu este posibil ca toate proprietățile obiectului să fie accesibile. Gândiți-vă ce s-ar întâmpla dacă fiecare dintre noi am auzi ce gândeşte celălalt despre noi!!!!

De aceea este mai bine să lăsăm fiecărui individ posibilitatea decizională relativ la părerea pe care o are despre noi, posibilitate internă, independentă de curiozitatea noastră.

În mod asemănător sunt o serie de proprietăți sau comportamente care sunt ascunse interacțiunii cu alte obiecte, fiind efectiv încapsulate în structura obiectului.

Mai mult, obiectul încapsulează şi modul de funcţionare a operaţiilor lui specifice, din exterior neputându-se observa decât modul de apelare a acestor operaţii şi rezultatele apelurilor. Cu alte cuvinte, ***procesul de încapsulare*** este procesul de ascundere a detaliilor neimportante sau sensibile de construcţie a obiectului.

Pe lângă proprietăţile unui obiect trebuie protejate şi o serie de operaţii definite de către acesta.

În practică este nevoie de o oarecare rafinare a gradului de protejare a fiecărei operaţii sau proprietăţi în aşa fel încât însuşi accesul observatorilor exteriori să poată fi nuanţat în funcţie de gradul de similitudine şi apropiere a observatorului faţă de obiectul accesat. Rafinarea trebuie să meargă până la a putea specifica pentru fiecare proprietate şi operaţie în parte care sunt observatorii care au acces şi care nu.

Această protejare şi încapsulare a proprietăţilor şi operaţiilor ce se pot executa cu ajutorul unui obiect are şi o altă consecinţă şi anume aceea că utilizatorul obiectului respectiv este independent de detaliile constructive ale obiectului respectiv. Structura internă a obiectului poate fi astfel schimbată şi perfecţionată în timp fără ca funcţionalitatea de bază să fie afectată.

Clase de obiecte

În realitate obiectele sunt grupate pe categorii sau familii de obiecte. În programare o familie de obiecte poartă numele de clasă. O clasă conţine caracteristicile privind proprietăţile şi comportamentul unui set de obiecte. În esenţă o clasă este o reprezentare generalizată a unei familii de obiecte, cuprinzând setul de proprietăţi care individualizează respectiva clasă de o altă clasă de obiecte.

Pentru o clarificarea noțiunilor se poate spune că toate obiectele fac parte întodeauna dintr-o familie mai mare de obiecte cu proprietăți şi comportament similar. Aceste familii de obiecte vor fi numite ***clase de obiecte*** sau ***concepte*** în timp ce obiectele aparținând unei anumite clase le vom numi ***instanțe*** ale clasei de obiecte respective sau ***exemple*** ale clasei respective (în engleză - ***instances***). Particularizarea atributelor unei clase conduce la definirea obiectelor respectiv a exemplelor – instanțelor. Această particularizare se datorează faptului că atributele sau proprietățile sunt descrise prin intermediul unor mărimi variabile. Proprietațile vor diferi de la un exemplu la altul, dar fiecare instanță a aceleiaşi clase va avea aceleaşi proprietăți şi aceleaşi operații vor putea fi aplicate asupra ei. În continuare vom numi ***variabile*** aceste proprietăți ale unei clase de obiecte şi vom numi ***metode*** operațiile definite pentru o anumită clasă de obiecte.

Concluzionând:

O clasă de obiecte este o descriere a proprietăților şi operațiilor specifice unui nou tip de obiecte reprezentabile în memorie.

O instanță sau un exemplu al unei clase de obiecte este un obiect de memorie care respectă descrierea clasei.

O variabilă a unei clase de obiecte este o proprietate a clasei respective care poate lua valori diferite în instanțe diferite ale clasei.

O metodă a unei clase este descrierea unei operații specifice clasei respective.

Metodele unei clase sunt memorate o singură dată pentru toate obiectele. Comportarea diferită a acestora este dată de faptul că ele depind de valorile variabilelor.

O structură particulară de clase sunt clasele ce nu se pot instanța direct, de obicei pentru că nu dispunem de destule informații pentru a le putea construi.

Clasele abstracte, neinstanțiabile, servesc în general pentru definirea unor proprietăți sau operații comune ale mai multor clase şi pentru a putea generaliza operațiile referitoare la acestea.

Dacă o clasă de obiecte are cel puțin o metodă abstractă, ea devine în întregime o ***clasă abstractă*** şi nu poate fi instanțiată, adică nu putem crea instanțe ale unei clase de obiecte abstracte.

Altfel spus, o clasă abstractă de obiecte este o clasă pentru care nu s-au precizat suficient de clar toate metodele astfel încât să poată fi folosită în mod direct.

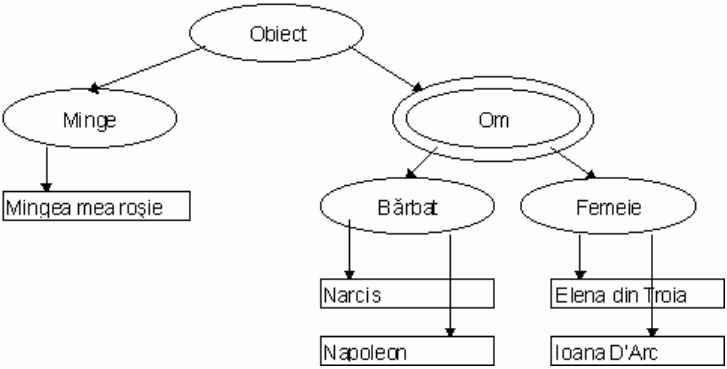
Ierarhizarea claselor de obiecte

Odată cu definirea unei noi clase de obiecte care reprezintă un concept, odată cu specificarea clasei de obiecte din care provine este specificată clasa care reprezintă conceptul original, împreună cu diferențele pe care le aduce noul concept derivat față de cel original. Această operație de definire a unei noi clase de obiecte pe baza uneia deja existente o vom numi ***derivare***. Conceptul mai general se va numi ***superconcept*** sau ***superclasă*** iar conceptul derivat din acesta se va numi ***subconcep****t* sau ***subclasă*.** În acelaşi mod, clasa originală se va numi ***superclasă*** a noii clase în timp ce noua clasă de obiecte se va numi ***subclasă*** a clasei derivate.

Uneori, în loc de derivare se foloseşte termenul de ***extindere***. Termenul vine de la faptul că o subclasă îşi extinde superclasa cu noi variabile şi metode.

În spiritul acestei ierarhizări, putem presupune că toate clasele de obiecte sunt derivate dintr-o clasă inițială, să-i spunem clasa de obiecte generice, în care putem defini proprietățile şi operațiile comune tuturor obiectelor precum ar fi testul de egalitate dintre două instanțe, duplicarea instanțelor sau aflarea clasei de care aparține o anumită instanță.

Ierarhizarea se poate extinde pe mai multe nivele, sub formă arborescentă, în fiecare punct nodal al structurii arborescente rezultate aflându-se clase de obiecte. Desigur, clasele de obiecte de pe orice nivel pot avea instanțe proprii, cu condiția să nu fie clase abstracte, imposibil de instanțiat.Mai jos, avem o ierarhie de clase de obiecte în care clasele sunt reprezentate în câmpuri eliptice iar instanțele acestora în câmpuri dreptunghiulare. Clasele abstracte de obiecte au elipsa dublată.



Construcția unei ierarhii de clase completă, care să conțină clase de obiecte corespunzătoare fiecărui concept cunoscut este o operațiune extrem de complicată. Pentru o problemă dată, conceptele implicate în rezolvarea ei sunt relativ puține şi pot fi uşor izolate, simplificate şi definite. Talentul unui programator este reflectat de restrângerea la minimum a arborelui de concepte necesar rezolvării unei anumite probleme fără a se afecta generalitatea soluției. De alegerea acestor concepte depinde eficiența şi flexibilitatea aplicației.

O clasă de obiecte derivată dintr-o altă clasă păstrează toate proprietățile şi operațiile acesteia din urmă aducând în plus proprietăți şi operații noi. Fiecare dintre clasele de obiecte derivate îşi vor defini propriile lor proprietăți şi operații pentru a descrie diferența dintre ele şi clasa originală.

Unele dintre proprietățile şi operațiile definite în superclasă pot fi redefinite în subclasele de obiecte derivate. Vechile proprietăți şi operații sunt disponibile în continuare, doar că pentru a le putea accesa va trebui să fie specificată explicit superclasa care deține copia redefinită. Operația de redefinire a unor operații sau variabile din interiorul unei clase în timpul procesului de derivare o vom numi ***rescriere***.

Această redefinire ne dă de fapt o mare flexibilitate în construcția ierarhiei unei probleme date pentru că nici o proprietate sau operație definită într-un punct al ierarhiei nu este impusă definitiv pentru conceptele derivate din acest punct direct sau nu.

Revenind pentru un moment la protejarea informațiilor interne ale unui obiect să precizăm faptul că gradul de similitudine de care vorbeam mai sus este mărit în cazul în care vorbim de două clase derivate una din cealaltă. Cu alte cuvinte, o subclasă a unei clase are acces de obicei la mult mai multe informații memorate în superclasa sa decât o altă clasă de obiecte oarecare. Acest lucru este firesc ținând cont de faptul că, uneori, o subclasă este nevoită să redefinească o parte din funcționalitatea superclasei sale.

Interfețe spre obiecte

Un obiect este o entitate complexă pe care o putem privi din diverse puncte de vedere.

Definițiile obiectelor sunt numite ***interfețe***, sunt aplicabile nu numai clasei de obiecte om, dar şi la alte clase de obiecte derivate sau nu din acesta, superclase sau nu ale acesteia. Putem să găsim o mulțime de clase de obiecte ale căror instanțe pot fi privite ca obiecte spațio-temporale dar care să nu aibă mare lucru în comun cu omul. Practic, atunci când construim o interfață, definim un set minim de operații care trebuie să aparțină obiectelor care respectă această interfață. Orice clasă de obiecte care declară că respectă această interfață va trebui să definească toate operațiile.

Operațiile însă, sunt definite pe căi specifice fiecărei clase de obiecte în parte. De exemplu, orice obiect spațial trebuie să definească o operație de modificare a poziției în care se află. Dar această operație este diferită la un om, care poate să-şi schimbe singur poziția, față de o minge care trebuie ajutată din exterior pentru a putea fi mutată. Totuşi, dacă ştim cu siguranță că un obiect este o instanță a unui clase de obiecte care respectă interfața spatio-temporală, putem să executăm liniştiți asupra acestuia o operație de schimbare a poziției, fără să trebuiască să cunoaştem amănunte despre modul în care va fi executată această operație. Tot ceea ce trebuie să ştim este faptul că operația este definită pentru obiectul respectiv.

În concluzie, o interfață este un set de operații care trebuiesc definite de o clasă de obiecte pentru a se înscrie într-o anumită categorie. Vom spune despre o clasă care defineşte toate operațiile unei interfețe că implementează interfața respectivă.

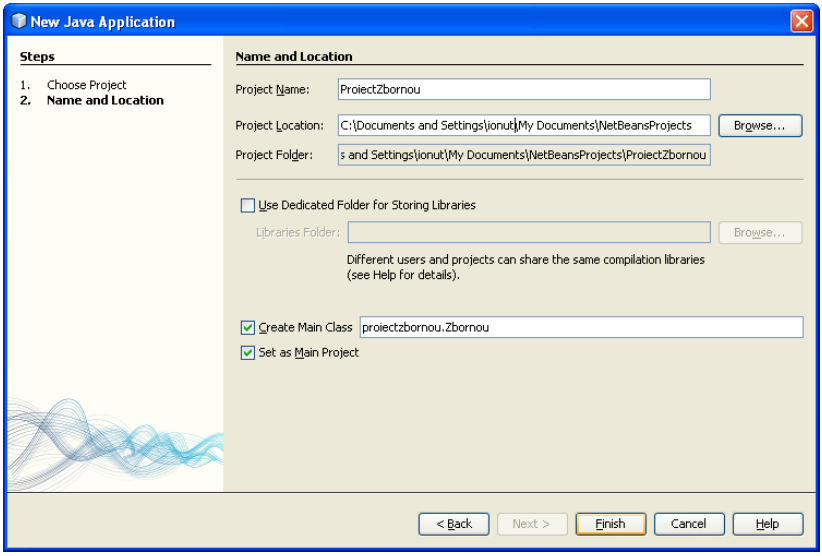
Cu alte cuvinte, putem privi interfețele ca pe nişte reguli de comportament impuse claselor de obiecte. În clipa în care o clasă implementează o anumită interfață, obiectele din clasa respectivă pot fi privite în exclusivitate din acest punct de vedere. Interfețele pot fi privite ca nişte filtre prin care putem privi un anumit obiect, filtre care nu lasă la vedere decât proprietățile specifice interfeței, chiar dacă obiectul în vizor este mult mai complicat în realitate.

Interfețele crează o altă împărțire a obiectelor cu care lucrăm. În afară de împărțirea normală pe clase, putem să împărțim obiectele şi după interfețele pe care le implementează. Şi, la fel cu situația în care definim o operație doar pentru obiectele unei anumite clase, putem defini şi operații care lucrează doar cu obiecte care implementează o anumită interfață, indiferent de clasa din care acestea fac parte.

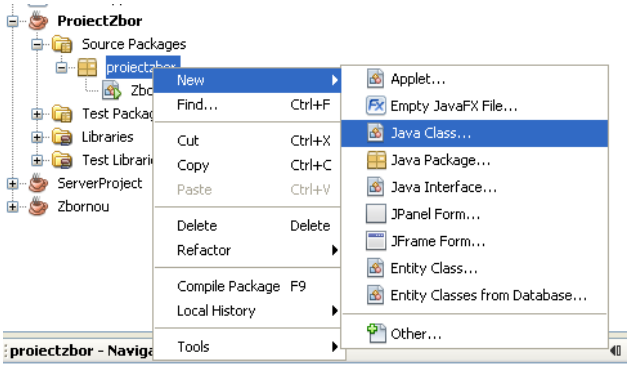
Aplicații

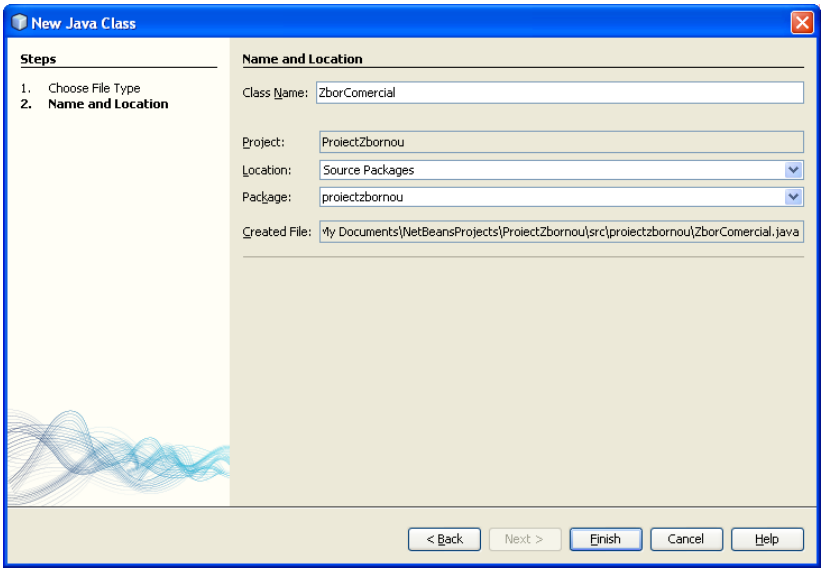
Editați următorul program observând elementele specifice programării orientat pe obiecte:

Creati un proiect nou in NetBeans şi setați opțiunea Create Main Class denumind clasa main Zbornou.

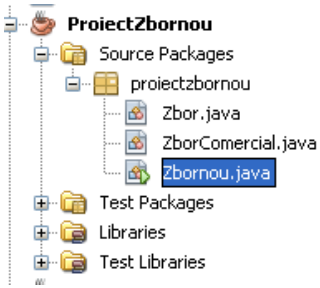


Creați clasa Zbor şi clasa ZborComercial folosind meniul pop-up New->Java Class din NetBeans.





Structura proiectului ar trebui sa arate ca in figura de mai jos:




```
class Zbor {
  int altitudine;
  private int azimut;
  int   viteza;
  float latitudine;
  float longitudine;

  void RotesteAvion (int unghi) {
    azimut = (azimut + unghi) % 360;
// unghiul trebuie sa fie cuprins intre 0-359
de grade
    if (azimut < 0)
      azimut = azimut + 360;
  }
  void SetAzimut (int unghi) {
    azimut= unghi % 360;
    // unghiul trebuie sa fie cuprins intre 0-
359 de grade
    if (azimut < 0)
      azimut = azimut + 360;
  }

  int GetAzimut() {
    return azimut;
  }
  // Afiseaza altitudinea, azimutul si viteza
zborului
  void AfiseazaZbor() {
    System.out.println (altitudine + " m " +
azimut + " grade  " + viteza + " km/ora ");
```

```
  }
}
```

O extensie a clasei **Zbor** se construieşte astfel:

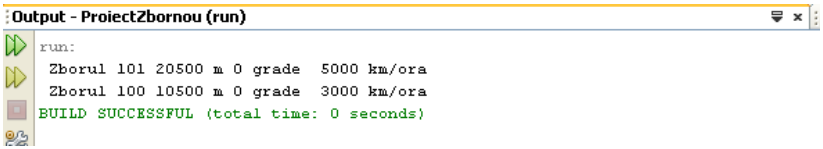
```
class ZborComercial extends Zbor {
//noi atribute specifice clasei ZborComercial

  int NumarZbor;
  int Nrpasageri;
  //se reface rutina de afisare a Zborului
 // utilizând definiţia anterioară

 void AfiseazaZbor() {
   System.out.print(" Zborul " + NumarZbor + " ");
   super.AfiseazaZbor();
  }
}
```

În continuare vor fi utilizate cele două clase formate. Editaţi şi compilaţi următorul program Java:

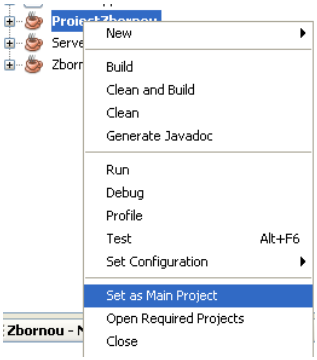
```
// * * * programul Zbornou.java *****

public class Zbornou extends ZborComercial
  {
  public static void main (String args[])
   {
//_____________________________________________
//  Crearea unor exemple sau incidenţe din
//obiectul ZborComercial
 ZborComercial Avion1 = new ZborComercial();

//Este particularizat numele incidenţei
 Avion1.altitudine = 20500;
//Este particularizat atributul altitudine din
//clasa primara Zbor
 Avion1.viteza = 5000;
```

```
//Este particularizat atributul viteza din
//clasa primara Zbor
  Avion1.NumarZbor = 101;
//Este particularizat atributul NumarZbor din
//clasa mostenitoare ZborComercial
  Avion1.Nrpasageri = 120;
//Este particularizat atributul Nrpasageri din
//clasa mostenitoare ZborComercial
//_____________________________________________

ZborComercial Avion2 = new ZborComercial();
                Avion2.altitudine = 10500;
                Avion2.viteza = 3000;
                Avion2.NumarZbor = 100;
                Avion2.Nrpasageri = 100;
//_____________________________________________
//Activarea metodelor mostenite:

                Avion1.AfiseazaZbor();
                Avion2.AfiseazaZbor();
          }
}
```

În urma compilării claselor Zbor, ZborComercial şi Zbornou şi a rulării clasei Zbornou rezultatul este următorul:

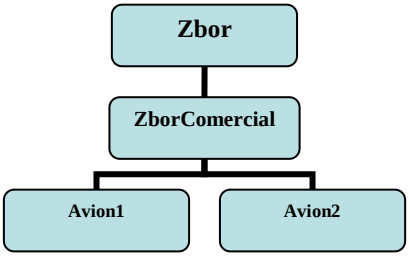


Atenție!
Pentru a fi considerat activ proiectul trebuie sa aiba numele scris cu caractere bold (Set as Main Project)



Rezultatul in consola MsDos ar trebui sa arate ca în figura:

Din punct de vedere al ierarhizării claselor acestea se prezintă în felul următor:



Chestiuni de studiat

1. Utilizați toate metodele disponibile în cadrul claselor formate: modificați azimutul, altitudinea, rotiți avioanele.
2. Construiți o nouă incidență Avion3 cu următoarele caracteristici: altitudine 15000, viteza 8000, latitudine 130, longitudine 200.
3. Pe baza aplicației Zbornou, introduceți şi rulați în cadrul unei noi aplicații următoarele linii de program care particularizează metodele claselor utilizate:

```
for (int i = 1;i<100;i++)
      {Avion1.SetAzimut(67+i);
      Avion1.AfiseazaZbor();
      Avion2.RotesteAvion(15*i);
      Avion2.AfiseazaZbor();
      Avion3.RotesteAvion(15*i);
      Avion3.AfiseazaZbor();
      }
```

4. Construiți o nouă metodă – Aterizare care decrementează viteza si altitudinea, pe un azimut de 15 grade. Faceți apel la metodele construite anterior

Observație: Utilizați structuri for reducerea vitezei şi a altitudinii.

5. Realizați un program care să modeleze circulatia autovehiculelor printr-o intersectie. Vor fi definite clase vehicule cu 2 roți, cu 4 roți, culoarea acestora, capacitatea cilindrică, marca acestora. Metodele folosite vor fi pornirea motorului, oprirea notorului, accelerare, decelerare. Circulatia va fi dirijata de utilizator, rezultatul fiind materializat prin afişarea pe monitor a ordinii în care trec vehiculele, specificându-se în cazul fiecăruia marca şi culoarea. **Toate aceste elemente sunt facultative, identificarea claselor, atributelor şi a metodelor necesare sunt la alegerea dumneavoastră.**
6. Creați un program care să modeleze activitatea dumneavoastra pe parcursul unei zile. Obiectele cu care operați sunt cursurile, laboratoarele ca derivate din cadrul familiei activitate didactica. Atributele sunt dimensiunea temporală a activității: o oră sau două ore, numele activității, poziția activității în cadrul programului orar,etc. Metodele

folosite sunt participarea la activitate, participarea la activitatea didactică, parerea personală legată de respectiva activitate. **Toate aceste elemente sunt facultative, identificarea claselor, atributelor şi a metodelor necesare sunt la alegerea dumneavoastră.**