Poliedrul

Presupunem că obiectul grafic 3D se **aproximează** printr-un poliedru, iar acest poliedru se precizează prin:

- vârfuri
- poligoane (fețe) plane

Memorarea poliedrelor

Pentru a memora un poliedru vom folosi următoarea clasă:

```
public class Poliedru:
  Integer NrPuncte;            //numarul de puncte (varfuri) din poliedru
  Vector3D[] Puncte;           //punctele (virfurile) poliedrului
  Integer NrPoligoane;         //numarul de poligoane
  Poligon[] Poligoane;         //poligoanele poliedrului
  public int TipAfisare;
    //tipul de afisare: 2-plin, 1-cu linii, 0-cu puncte
    //alte proprietati: normale, textura, proprietati ale materialului,
    //      transformari asociate poliedrului
  public Poliedru(int m, int n)
    //construieste un poliedru cu m virfuri si n poligoane
  public void AdaugPunct(int i, float x,float y, float z)
  public void AdaugPoligon(int i, int s, int x1, int x2, int x3, int x4, int x5)
    //Adaugarea unui poligon (fata) la poliedru. Poliedrul are s (3<=s<=5)
    //   varfuri, precizate in x1,...,x5
  public void AdaugPoligonM(int i, int s, Integer[] V)
    //Adaugarea unui poligon (fata) la poliedru. Poliedrul are s varfuri
    //   precizate in vectorul V (vector cu elemente de tipul Integer)
  public void Centrare(float r)
    //centrul obiectului -> origine
    //obiectul se inscrie intr-un cub de latura r
  public void CuloarePoligon(int i, float r, float g, float b)
    //culoarea poligonului i
  public void CuloareUnica(float r, float g, float b)
    //culoarea unica pentru fiecare poligon
  public void CuloareAleatoare()
    //culoarea fiecarui poligon este stabilita aleator
  public void Afisare(GL gl)
    //afisarea poliedrului
  public Poliedru Stelare(float h){
    //din poliedrul curent (centrat) se obtine poliedrul stelat,
    // h=inaltimea piramidei ce inlocuieste fiecare poligon
```

Clasa Poligon folosită este:

```
class Poligon:
  //poligonul (fata) unui poliedru
  Integer Nrp;         //numarul de puncte (virfuri) din poligon
  Culoare Culoare;     //coloarea poligonului
  Integer[] Ep;
        //extremitatile poligonului (indexuri in lista de virfuri)
  //alte date pt. poligon (textura, proprietatile materialului)
  public Poligon(int Nrp)  //constructorul clasei
```

Poliedre particulare

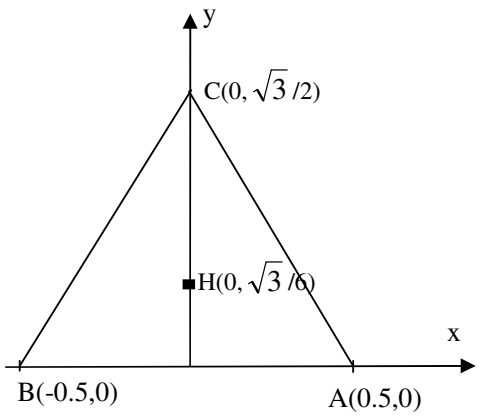
- **convex** dacă segmentul care uneşte oricare două puncte ale poliedrului este inclus complet în interiorul poliedrului.
- **regulat** dacă toate vârfurile, muchiile şi feţele sale sunt la fel, în sensul că există o simetrie care transformă orice faţă în altă faţă.

Poliedre convexe şi regulate (http://ro.wikipedia.org/wiki/Poliedru)**:**

Nume	Vârfuri	Muchii	Feţe	Muchii la o faţă	Muchii la un vârf
Tetraedru	4	6	4	3	3
Cub sau hexaedru	8	12	6	4	3
Octaedru	6	12	8	3	4
Icosaedru	12	30	20	3	5
Dodecaedru	20	30	12	5	3

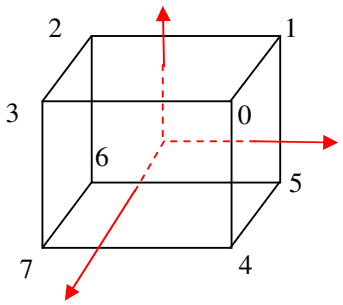
Tetraedrul

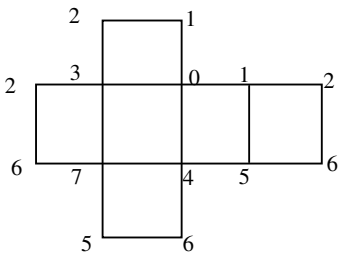
```
public class Tetraedru extends Poliedru{
  //tetraedru de latura 1
  public Tetraedru(){
    super(4,4);  //apel constructor
    AdaugPunct(0, 0.5f,0f,0f);
    AdaugPunct(1,-0.5f,0f,0f);
    AdaugPunct(2,0f,(float)Math.sqrt(3)/2f,
       0f);
    AdaugPunct(3,0f,(float)Math.sqrt(3)/6f,
       (float)Math.sqrt(6)/3f);
    AdaugPoligon(0,3,0,2,3,0,0);
    AdaugPoligon(1,3,1,3,2,0,0);
    AdaugPoligon(2,3,0,1,2,0,0);
    AdaugPoligon(3,3,0,3,1,0,0);
    Centrare(1);
  }
}
```



Cubul

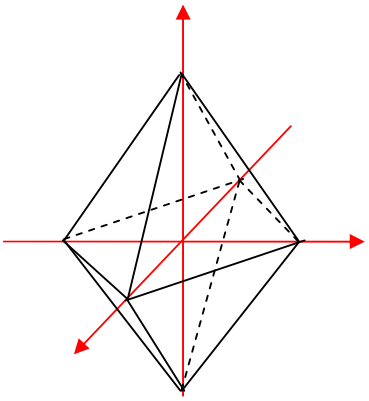
```
public class Cub extends Poliedru{
  public Cub(){
    super(8,6);
    AdaugPunct(0,1,1,1);
    AdaugPunct(1,-1,1,1);
    AdaugPunct(2,-1,-1,1);
    AdaugPunct(3,1,-1,1);
    AdaugPunct(4,1,1,-1);
    AdaugPunct(5,-1,1,-1);
    AdaugPunct(6,-1,-1,-1);
    AdaugPunct(7,1,-1,-1);
    AdaugPoligon(0,4,0,1,2,3,0);
    AdaugPoligon(1,4,4,0,3,7,0);
    AdaugPoligon(2,4,0,4,5,1,0);
    AdaugPoligon(3,4,1,5,6,2,0);
    AdaugPoligon(4,4,2,6,7,3,0);
    AdaugPoligon(5,4,5,4,7,6,0);
    Centrare(1);
  }
}
```





Octoedrul

```
public class Octoedru extends Poliedru{
```



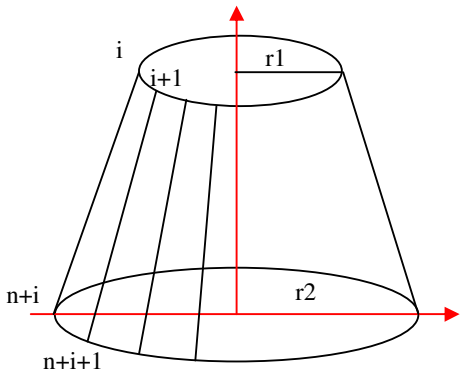
```
  public Octoedru(){
    super(6,8);
    AdaugPunct(0,0,0,1);
    AdaugPunct(1,0,0,-1);
    AdaugPunct(2,1,0,0);
    AdaugPunct(3,-1,0,0);
    AdaugPunct(4,0,1,0);
    AdaugPunct(5,0,-1,0);
    AdaugPoligon(0,3,0,2,4,0,0);
    AdaugPoligon(1,3,0,4,3,0,0);
    AdaugPoligon(2,3,0,3,5,0,0);
    AdaugPoligon(3,3,0,5,2,0,0);
    AdaugPoligon(4,3,1,2,5,0,0);
    AdaugPoligon(5,3,1,4,2,0,0);
    AdaugPoligon(6,3,1,5,3,0,0);
    AdaugPoligon(7,3,1,3,4,0,0);
    Centrare(1);
  }
}
```

Icosaedrul

Dodecaedru

Trunchi de con gol la baze

```
public class extends Poliedru{
  public TrConGol(float r1, float r2, float h, int n) {
    super(2*n,n);
    int i,i1;
    for (i=0;i<n;i++){
      AdaugPunct(i,(float)(r1*Math.cos(2*Math.PI*i/n)),
          (float)(r1*Math.sin(2*Math.PI*i/n)),h);
      AdaugPunct(n+i,(float)(r2*Math.cos(2*Math.PI*i/n)),
          (float)(r2*Math.sin(2*Math.PI*i/n)),0);
    }

    for (i=0;i<n;i++){
      i1=i+1;
      if (i==n-1) i1=0;
      AdaugPoligon(i,4,n+i,n+i1,i1,i,0);
    }
  }
}
```

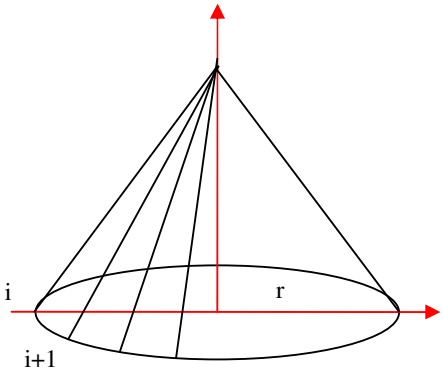


Cilindrul gol la baze

```
public class CilindruGol extends TrConGol{
  public CilindruGol(float r, float h, int n){
    super(r,r,h,n);
  }
}
```

Con gol la bază

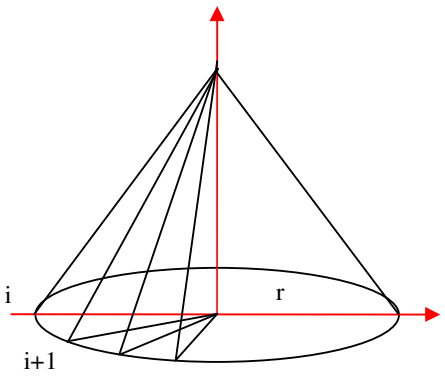
```
public class ConGol extends Poliedru{

  public ConGol(float r, float h, int n) {
    super(n+1,n);
    int i,i1;
    for (i=0;i<n;i++)

          AdaugPunct(i,(float)(r*Math.cos(2*Math.
          PI*i/n)),
          (float)(r*Math.sin(2*Math.PI*i/n)),0);
```



```
    AdaugPunct(n,0,0,h);

    for (i=0;i<n;i++){
      i1=i+1;
      if (i==n-1) i1=0;
      AdaugPoligon(i,3,n,i,i1,0,0);
    }
  }
}
```

Con plin

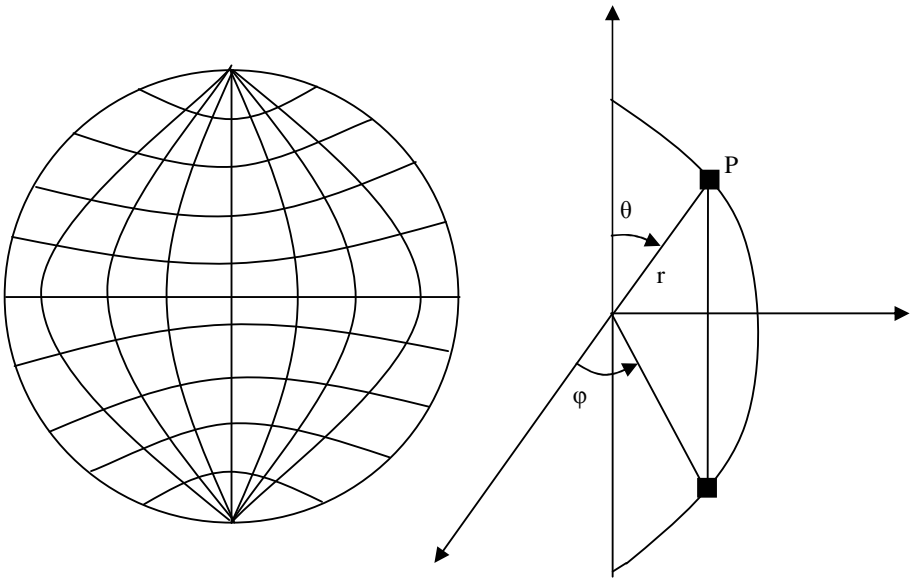
```
public class ConPlin extends Poliedru{

  public ConPlin(float r, float h, int n) {
    super(n+2,2*n);
    int i,i1;
    for (i=0;i<n;i++)
      AdaugPunct(i,(float)(r*Math.cos(2*Math.PI*i/n)),
        (float)(r*Math.sin(2*Math.PI*i/n)),0);
    AdaugPunct(n,0,0,h);
    AdaugPunct(n+1,0,0,0);

    for (i=0;i<n;i++){
      i1=i+1;
      if (i==n-1) i1=0;
      AdaugPoligon(i,3,n,i,i1,0,0);
      AdaugPoligon(n+i,3,n+1,i,i1,0,0);
    }
  }
}
```



Sfera

```
public class Sfera extends Poliedru{

  public Sfera(float r, int m, int n) {
    super((m-1)*n+2,m*n);

    int i,j,j1;

    for (i=1;i<=m-1;i++)
      for (j=0;j<n;j++)
        AdaugPunct((i-1)*n+j,
          (float)(r*Math.sin(Math.PI*i/m)*Math.cos(2*Math.PI*j/n)),
          (float)(r*Math.sin(Math.PI*i/m)*Math.sin(2*Math.PI*j/n)),
          (float)(r*Math.cos(Math.PI*i/m)));
    AdaugPunct((m-1)*n,0,0,r);
    AdaugPunct((m-1)*n+1,0,0,-r);
    for (i=1;i<=m-2;i++){
      for (j=0;j<n;j++){
        if (j==n-1){
          j1=0;
        }else j1=j+1;
        AdaugPoligon((i-1)*n+j,4,(i-1)*n+j,i*n+j,i*n+j1,(i-1)*n+j1,0);
      }
    }
    for (j=0;j<n;j++){
      if (j==n-1){
            j1=0;
      }else j1=j+1;
      AdaugPoligon((m-2)*n+j,3,(m-1)*n,j,j1,0,0);
      AdaugPoligon((m-1)*n+j,3,(m-1)*n+1,(m-2)*n+j1,(m-2)*n+j,0,0);
    }
  }
```

}



P(r*sin(θ)*cos(φ), r*sin(θ)*cos(φ), r*cos(θ))

Sunt **(m-1)*n+2** vârfuri

- P_{ij}, θ_i, i=1..m-1; φ_j, j=0..n-1 => (m-1)*n vârfuri
- A(0,0,r); B(0,0,-r) => "polii"

m*n poligoane:

- **2n** triunghiuri
- **n*(m-2)** patrulatere