

# AS3 – PART II

Cosmin Varlan

# Reamintim ca....

- Introducere in AS3 & Hello World (nu uitati: new->ActionScriptProject, nu altceva).
- Numele fisierului trebuie sa fie la fel cu cel al clasei.
- Numele directorului trebuie sa fie acelasi cu cel al pachetului.
- Pentru a putea utiliza o clasa externa trebuie sa o importam, aplicatia principala mereu va extinde Sprite (sau MovieClip).

# Reamintim ca....

- Constructorul este **metoda** ce va fi apelata la crearea obiectului si are acelasi nume ca si clasa (sau ca fisierul in care se afla).
- Proprietatile si metodele clasei vor fi de tip:
  - ❖ **public**: accesibil de oriunde;
  - ❖ **private**: accesibil numai in clasa curenta;
  - ❖ **protected**: accesibil in clasa curenta si in cele ce o extind;
  - ❖ **internal**: accesibil numai in cadrul pachetului

- **De aici luati moka numar serial de flex  
(pt studenti si profesori):**

**<https://freeriatools.adobe.com/flex/>**

- **De aici luati flexul:**

**<http://www.adobe.com/products/flex/>**

# Variabilele cele mai simple...

- uint – intreg fara semn pe 32 biti
- Int – intreg cu semn pe 32 biti
- Number – numar real pe 64biti
- Boolean – valori booleene: true / false
- String – siruri de caractere

# Proprietate privata sau publica ?!

- Cand vom utiliza o proprietate in interiorul clasei si numai acolo va fi declarata PRIVATA !
- Atunci cand construim o clasa, trebuie sa ne gandim la REUTILIZABILITATE (da, si mie mi-a fost greu sa scriu cuvantu' asta).
- As dori sa am o proprietate publica dar mi-e frica sa nu ma trezesc cu valori anormale in ea... (de exemplu viteza sa nu depaseasca 300 si sa nu fie negativa)

# Proprietate privata sau publica ?!

- Pentru a nu fi negativa, o facem uint.. Simplu
- Pentru a nu depasi 300 ... IDEI CINEVA ?!?!

```
package classes
{
    public class Masina
    {
        private var _viteza : uint = 0;
        public function Masina()
        {
        }
    }
}
```

# Metode speciale de tip set / get...

```
package classes
{
    public class Masina
    {
        private var _viteza : uint = 0;
        public function Masina()
        {
        }
        public function set viteza( i : uint ):void
        {
            _viteza = i < 300 ? i : 300;
            // if (i<300) _viteza = i else _viteza = 300;
        }
    }
}
```

Daca parametrul primit este mai mic decat 300 atunci voi face **\_viteza** sa fie 300... e clasa Masina, nu Racheta...

# Metode speciale de tip set / get...

```
package classes
{
    public class Masina
    {
        private var _viteza : uint = 0;
        public function Masina()
        {
        }
        public function set viteza( i:uint ):void
        {
            _viteza = i < 300 ? i : 300;
            // if (i<300) _viteza = i else _viteza = 300;
        }
    }
}
```

Hei, eu ma simt ignorata...

# Metode speciale de tip set / get...

```
package classes
{
    public class Masina
    {
        private var _viteza : uint = 0;
        public function Masina()
        {
        }
        public function set viteza( i:uint ):void
        {
            _viteza = i < 300 ? i : 300;
            // if (i<300) _viteza = i else _viteza = 300;
        }
    }
}
```

Pai chiar esti, eu fac  
acelasi lucru ca si tine  
dar in mai putine cuvinte  
😊

# Metode speciale de tip set / get...

```
package classes
{
    public class Masina
    {
        private var _viteza : uint = 0;
        public function Masina()
        {
        }
        public function set viteza( i:uint ):void
        {
            _viteza = i < 300 ? i : 300;
            // if (i<300) _viteza = i else _viteza = 300;
        }
    }
}
```

Daca i este mai mic decat 300 ii voi atribui valoarea de dupa semnul intrebarii, altfel ce e dupa doua puncte ... sâc sâc :D

# Metode speciale de tip set / get...

```
package classes
{
    public class Masina
    {
        private var _viteza : uint = 0;
        public function Masina()
        {
        }
        public function set viteza( i:uint ):void
        {
            _viteza = i < 300 ? i : 300;
            // if (i<300) _viteza = i else _viteza = 300;
        }
    }
}
```

Daca cineva va instantia un obiect de tip Masina, eu voi putea fi apelata ca un parametru public al acestui obiect.

# Metode speciale de tip set / get...

```
package classes
{
    public class Masina
    {
        private var _viteza

        public function Masina
        {
        }

        public function set viteza( i:uint ):void
        {
            _viteza = i < 300 ? i : 300;
            // if (i<300) _viteza = i else _viteza = 300;
        }
    }
}
```

Adica am putea avea (in aplicatie):

```
var masina:Masina = new Masina();
masina.viteza = 200;
```

Ultima linie e in loc de `masina.viteza(200);`

# Metode speciale de tip set / get...

```
package classes
```

```
{
```

```
    public class Masina
```

De fapt voi seta valoarea proprietatii

```
        private
```

```
        _viteza : uint = 0;
```

```
    public function Masina()
```

```
    {
```

```
    }
```

```
    public function set viteza( i:uint ):void
```

```
    {
```

```
        _viteza = i < 300 ? i : 300;
```

```
    }
```

```
    public function get viteza():uint
```

```
    {
```

```
        return _viteza;
```

```
    }
```

```
}
```

```
Cosmin }
```

Iar eu voi returna valoarea proprietatii  
private

# Metode speciale de tip set / get...

package classes

Daca apar in stanga unei expresii, voi seta valoarea proprietatii private **\_viteza**.

Daca sunt apelata in dreapta unei expresii, voi returna valoarea proprietatii private **\_viteza**.

```
private uint _viteza;
public function Masina()
{
}
public function set viteza( i:uint ):void
{
    _viteza = i < 300 ? i : 300;
}
public function get viteza():uint
{
    return _viteza;
}
}
```

# Metode speciale de tip set / get...

`package classes`

Ca Setter, mereu voi avea un singur parametru de intrare (poate fi si de tip Object) si voi returna void (nimic)... sunt egoist !

Ca Getter, nu am nevoie de parametri... totusi voi returna mereu valoarea proprietatii ce o reprezinta.

```
public function Masina()  
{  
}  
public function set viteza( i:uint ):void  
{  
    _viteza = i < 300 ? i : 300;  
}  
public function get viteza():uint  
{  
    return _viteza;  
}  
}
```

Funcțiile de tip get / set vor fi MEREU declarate public !

# Metode speciale de tip set / get...

```
package classes
{
    public class Masina
    {
        private var _viteza : uint = 0;
        public function Masina(i : uint)
        {
        }
        public function set_viteza(i : uint)
        {
            _viteza = i < 300 ? i : 300;
        }
        public function get_viteza() : uint
        {
            return _viteza;
        }
    }
}
```

Cu aceste functii as putea considera ca am o proprietate "viteza" publica, si cand cineva incearca sa afle valoarea acesteia voi apela getter, cand o va seta voi apela setterul.

# Si scriptul din care facem apelul...

```
package {  
    import classes.Masina;  
  
    import flash.display.Sprite;  
  
    public class Vitezomanul extends Sprite  
    {  
        private var masina:Masina = new Masina();  
        public function Vitezomanul() {  
            masina.v  
        }  
    }  
}
```

Flex trateaza **viteza** ca o proprietate publica a obiectului.

The screenshot shows a dropdown menu in an IDE. The menu is titled 'masina.v' and contains two items: 'valueOf() Object' and 'viteza'. The 'viteza' item is selected and highlighted. The menu has a scroll bar at the bottom.

# Si scriptul din care facem apelul...

```
package {  
    import classes.Masina;  
  
    import flash.display.Sprite;  
  
    public class Vitezomanul extends Sprite  
    {  
        private var masina:Masina = new Masina();  
        public function Vitezomanul()  
        {  
            masina.viteza = 100;  
            trace( masina.viteza ); // 100  
            masina.viteza = 400;  
            trace( masina.viteza ); // 300  
        }  
    }  
}
```

# Concluzii:

- Cand dorim sa restrictionam sau sa interogam proprietati ale obiectelor este bine sa facem acest lucru prin setteri / getteri.
- Faceti private datele private, faceti private si pe cele publice si puneti-le setteri/getteri 😊

[eu recunosc ca sunt cam puturos si le fac publice]

# Alte metode de a ne proteja proprietatile

- Impotriva valorilor neplacute... :D
- Daca avem o proprietate ce poate avea doar valorile “Rosu” si “Verde” se vor declara constante care sa aiba aceste valori.

# O clasa ce are o functie ce vrea o culoare 😊

```
package classes
{
    public class Culori
    {
        public const ROSU    : String = "Rosu";
        public const VERDE   : String = "Verde";
        public function Culori()
        {
        }
        public function vreauCuloare( culoare:String ):void
        {
            if (culoare=="Rosu") trace "Imi place";
            else
            if (culoare=="Verde") trace "Imi place";
            else
            trace( "Nu stiu culoarea" );
        }
    }
}
```

Aici s-ar putea da si "rosu" si "Rosu". Mie imi place cu R mare :D

# Posibilitatile sunt recunoscute in aplicatie:

```
package {  
    import classes.Culori;  
  
    import flash.display.Sprite;  
  
    public class CuloriConstante extends Sprite  
    {  
        private var c:Culori = new Culori();  
        public function CuloriConstante()  
        {  
            vreauCuloare(culoare:String):v  
            c.vreauCuloare( c.  
        }  
    }  
}
```

Putem pune direct constanta si nu riscam sa gresim...

- ◆ constructor
- ROSU
- VERDE
- hasOwnProperty(name:String):Boolean
- isPrototypeOf(theClass:Object):Boolean
- propertyIsEnumerable(name:String):Boolean
- setPropertyIsEnumerable(name:String, isEnum:Boolean=t
- toLocaleString():String

# O noua aplicatie cu figuri geometrice...

- Figura geometrica va fi implementata la nivel formal (nu vor fi desenate efectiv obiectele in scena).
- Pachetul se va numi “**geometrie**”, clasele vor fi: **Cerc, Dreptunghi, Patrat**. Fiecare va contine metoda “**toString**” ce va returna tipul figurii ca un String si metoda “**aria**” ce va returna un Number reprezentand aria figurii.
- Stie cineva definitia patratului ?!?! (gen proxim: dreptunghi; diferenta specifica: laturi alaturate egale) ... [de ce am intrebam asta ?]

# Am putea face patratul sa fie extins din dreptunghi – cum ?

- Simplu:

```
class Patrati extends Dreptunghi
```

```
package geometrie
```

```
{
```

```
public class Dreptunghi
```

```
{
```

```
protected var _latime:uint;
```

```
protected var _lungime:uint;
```

```
public var a:uint;
```

```
public function Dreptunghi( lat:uint = 0, lung:uint = 0 ):void
```

```
{
```

```
    _latime = lat < lung ? lat : lung;
```

```
    _lungime = lat < lung ? lung : lat;
```

```
}
```

```
public function set latime( dimensiune:uint):void
```

```
{
```

```
    if( dimensiune>0 ) _latime = dimensiune;
```

```
}
```

```
public function set lungime( dimensiune:uint):void
```

```
{
```

```
    if( dimensiune>0 ) _lungime = dimensiune;
```

```
}
```

```
public function aria():Number
```

```
{
```

```
    return ( _latime * _lungime );
```

```
}
```

```
public function toString():String
```

```
{
```

```
    return("Dreptunghi");
```

```
}
```

```
}
```

```
}
```

Am doua dimensiuni... doar sunt dreptunghi

```
package geometrie
```

```
{
```

```
public class Dreptunghi
```

```
{
```

```
protected var _latime:uint;
```

```
protected var _lungime:uint;
```

```
public var a:uint;
```

```
public function Dreptunghi( lat:uint = 0, lung:uint = 0 ):void
```

```
{
```

```
    _latime = lat < lung ? lat : lung;
```

```
    _lungime = lat < lung ? lung : lat;
```

```
}
```

```
public function set latime( dimensiune:uint):void
```

```
{
```

```
    if( dimensiune>0 ) _latime = dimensiune;
```

```
}
```

```
public function set lungime( dimensiune:uint):void
```

```
{
```

```
    if( dimensiune>0 ) _lungime = dimensiune;
```

```
}
```

```
public function aria():Number
```

```
{
```

```
    return ( _latime * _lungime );
```

```
}
```

```
public function toString():String
```

```
{
```

```
    return("Dreptunghi");
```

```
}
```

```
}
```

```
}
```

Valori predefinite (daca nu sunt dati parametri)

```
package geometrie
```

```
{
```

```
    public class Dreptunghi implements IGeom
```

```
    {
```

```
        protected var _latime:uint;
```

```
        protected var _lungime:uint;
```

```
        public var a:uint;
```

```
        public function Dreptunghi( lat:uint = 0, lung:uint = 0 ):void
```

```
        {
```

```
            _latime = lat < lung ? lat : lung;
```

```
            _lungime = lat < lung ? lung : lat;
```

```
        }
```

```
        public function set latime( dimensiune:uint):void
```

```
        {
```

```
            if( dimensiune>0 ) _latime = dimensiune;
```

```
        }
```

```
        public function set lungime( dimensiune:uint):void
```

```
        {
```

```
            if( dimensiune>0 ) _lungime = dimensiune;
```

```
        }
```

```
        public function aria():Number
```

```
        {
```

```
            return ( _latime * _lungime );
```

```
        }
```

```
        public function toString():String
```

```
        {
```

```
            return("Dreptunghi");
```

```
        }
```

```
    }
```

```
}
```

Voi verifica daca sunt  
date cum trebuie  
lungimea/latimea

```
package geometrie
```

```
{
```

```
public class Dreptunghi implements IGeom
```

```
{
```

```
protected var _latime:uint;
```

```
protected var _lungime:uint;
```

```
public var a:uint;
```

```
public function Dreptunghi( lat:uint = 0, lungime:uint = 0 ):void
```

```
{
```

```
    _latime = lat < lungime ? lat : lungime;
```

```
    _lungime = lat < lungime ? lungime : lat;
```

```
}
```

```
public function set latime( dimensiune:uint):void
```

```
{
```

```
    if( dimensiune>0 ) _latime = dimensiune;
```

```
}
```

```
public function set lungime( dimensiune:uint):void
```

```
{
```

```
    if( dimensiune>0 ) _lungime = dimensiune;
```

```
}
```

```
public function aria():Number
```

```
{
```

```
    return ( _latime * _lungime );
```

```
}
```

```
public function toString():String
```

```
{
```

```
    return("Dreptunghi");
```

```
}
```

```
}
```

```
}
```

Verificam daca dimensiunea data este mai mare ca 0.

```
package geometrie
```

```
{
```

```
public class Dreptunghi implements IGeom
```

```
{
```

```
protected var _latime:uint;
```

```
protected var _lungime:uint;
```

```
public var a:uint;
```

```
public function Dreptunghi( lat:uint = 0, lung:uint = 0 ):void
```

```
{
```

```
    _latime = lat < lung ? lat : lung;
```

```
    _lungime = lat < lung ? lung : lat;
```

```
}
```

```
public function set latime( dimensiune:uint):void
```

```
{
```

```
    if( dimensiune>0 ) _latime = dimensiune;
```

```
}
```

```
public function set lungime( dimensiune:uint):void
```

```
{
```

```
    if( dimensiune>0 ) _lungime = dimensiune;
```

```
}
```

```
public function aria():Number
```

```
{
```

```
    return ( _latime * _lungime );
```

```
}
```

```
public function toString():String
```

```
{
```

```
    return("Dreptunghi");
```

```
}
```

```
}
```

```
}
```

Eu calculez aria

```
package geometrie
```

```
{
```

```
public class Dreptunghi implements IGeom
{
    protected var _latime:uint;
    protected var _lungime:uint;
    public var a:uint;
    public function Dreptunghi( lat:uint = 0, lung:uint = 0 ):void
    {
        _latime = lat < lung ? lat : lung;
        _lungime = lat < lung ? lung : lat;
    }
    public function set latime( dimensiune:uint):void
    {
        if( dimensiune>0 ) _latime = dimensiune;
    }
    public function set lungime( dimensiune:uint):void
    {
        if( dimensiune>0 ) _lungime = dimensiune;
    }

    public function aria():Number
    {
        return ( _latime * _lungime )
    }
    public function toString():String
    {
        return("Dreptunghi");
    }
}
```

Iar eu returnez tipul figurii...

```
}
```

Patratal va extinde dreptunghiul.

Cum stie patratal sa extinda dreptunghiul daca nu a facut import la aceasta clasa ??

```
package geometrie
{
    public class Patratal extends Dreptunghi
    {
        private var _latura:uint;
        public function Patratal( latura:uint)
        {
        }
    }
}
```

Clasele din acelasi pachet stiu unele de altele (de aceea nu este nevoie de import)

```
package geometrie
{
    public class Patrat extends Dreptunghi
    {
        private var _latura:uint;
        public function Patrat( latura:uint)
        {
        }
    }
}
```

Patratul are o singura dimensiune  
ce poate fi primita ca parametru al  
constructorului...

```
package geometrie
{
    public class Patrati extends Dreptunghi
    {
        private var _latura:uint;
        public function Patrati( latura:uint)
        {
        }
    }
}
```

```
package geometrie
{
    public class Patrati extends Dreptunghi
    {
        private var _latura:uint;
        public function Patrati( latura:uint)
        {
            super( latura, latura );
        }
        public function set latura(l:uint):void
        {
            if (l>0) {latime = l; lungime = l;}
        }
    }
}
```

Vom apela constructorul superclasei in mod explicit utilizand metoda super. Sa nu uitam ca pentru dreptunghi sunt necesare doua dimensiuni....

```
package geometrie
{
    public class Patrat extends Dreptunghi
    {
        private var _latura:uint;
        public function Patrat( latura:uint)
        {
            super( latura, latura);
        }
        public function set latura(l:uint):void
        {
            if (l>0) {latime = l; lungime = l;}
        }
    }
}
```

Putem face iarasi un setter pentru latura...

1

```
package geometrie
{
    public class Patrat extends Dreptunghi
    {
        private var _latura: uint;
        public function Patrat(latura: uint)
        {
            super(latura);
        }
        public function toString(): String
        {
            if (_latura > 0) {
                return "Patrat cu latura: " + _latura;
            }
            return "Patrat";
        }
    }
}
```

Ar mai trebui metodele **toString** si **aria**. Acestea sunt mostenite din clasa Dreptunghi si de fapt le avem si aici.

Sa trecem la construirea clasei principale in care vom instantia obiectele de tip patrat si dreptunghi.

# Clasa principala...

```
package {  
    import flash.display.Sprite;  
    import geometrie.*;  
  
    public class interfaceExample extends Sprite  
    {  
        public function interfaceExample()  
        {  
            var dreptunghi:Dreptunghi = new Dreptunghi(5,7);  
            var patrat:Patrat = new Patrat(5);  
            trace("Aveam un " + dreptunghi.toString() + " cu aria " + dreptunghi.aria() );  
            trace("Aveam un " + patrat.toString() + " cu aria " + patrat.aria() );  
        }  
    }  
}
```

Eu sunt un  
obiect de tip  
Dreptunghi

Eu sunt un  
obiect de tip  
Patrat

# Clasa principala...

```
package {  
    import flash.display.Sprite;  
    import geometrie.*;  
  
    public class interfaceExample extends Sprite  
    {  
        public function interfaceExample()  
        {  
            var dreptunghi:Dreptunghi = new Dreptunghi(5,7);  
            var patrat:Patrat = new Patrat(5);  
            trace("Avem un " + dreptunghi.toString() + " cu aria " + dreptunghi.aria() );  
            trace("Avem un " + patrat.toString() + " cu aria " + patrat.aria() );  
        }  
    }  
}
```

Constructorul meu are 2 parametri

Al meu doar 1 dar stiu eu sa il dublic ca sa il folosesc pe Dl. Dreptunghi 😊

# Clasa principala...

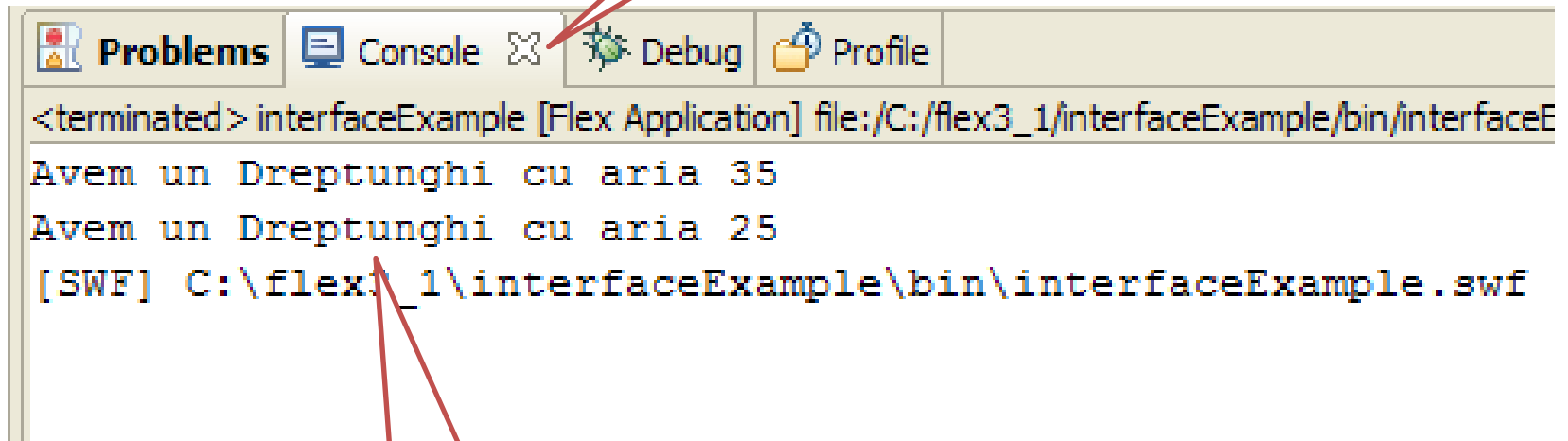
```
package {  
    import flash.display.Sprite;  
    import geometrie.*;  
  
    public class interfaceExample extends Sprite  
    {  
        public function interfaceExample()  
        {  
            var dreptunghi:Dreptunghi = new Dreptunghi(5,7);  
            var patrat:Patrat = new Patrat(5);  
            trace("Avem un " + dreptunghi.toString() + " cu aria " + dreptunghi.aria() );  
            trace("Avem un " + patrat.toString() + " cu aria " + patrat.aria() );  
        }  
    }  
}
```

Ambele  
obiecte au  
implementata  
metoda "aria"

Aria Patratului este mostenita din  
Dreptunghi. De fapt patratul poate fi  
considerat si el tot obiect de tip Dreptunghi  
caruia i-am facut laturile egale

# Clasa principala...

Rezultatul va fi vizibil  
in Consola

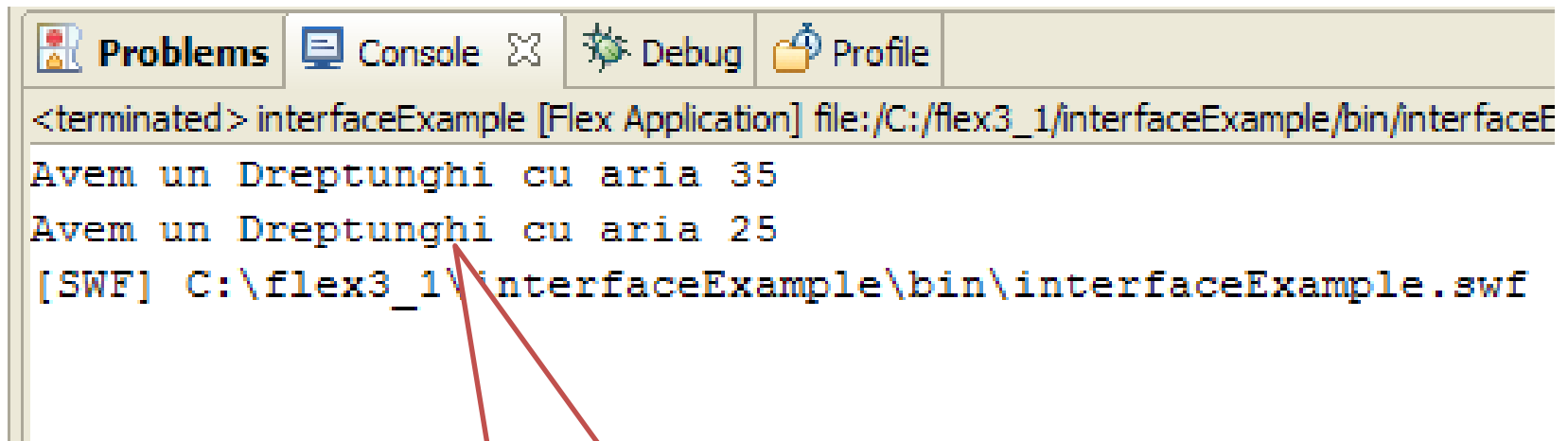


The screenshot shows an IDE interface with a console window. The console title bar includes 'Problems', 'Console', 'Debug', and 'Profile'. The console content displays the following text:

```
<terminated> interfaceExample [Flex Application] file:/C:/flex3_1/interfaceExample/bin/interfaceE  
Avem un Dreptunghi cu aria 35  
Avem un Dreptunghi cu aria 25  
[SWF] C:\flex3_1\interfaceExample\bin\interfaceExample.swf
```

UPS ... asta parca era patrat

# Clasa principala...



The screenshot shows an IDE console window with tabs for Problems, Console, Debug, and Profile. The console output is as follows:

```
<terminated> interfaceExample [Flex Application] file:/C:/flex3_1/interfaceExample/bin/interfaceE
Avem un Dreptunghi cu aria 35
Avem un Dreptunghi cu aria 25
[SWF] C:\flex3_1\interfaceExample\bin\interfaceExample.swf
```

Solutia esteeee....  
... suprascrierea metodei toString();

# Suprascrierea unei metode: override

```
package geometrie
{
    public class Patrati extends Dreptunghi
    {
        private var _latime:uim
        public function Patrati(
        {
            super( latime, latime)
        }
        public function set latime( l):void
        {
            if (l>0) {latime = l; lungime = l;}
        }
        public override function toString():String
        {
            return( "Patrati" );
        }
    }
}
```

Eu pot permite unui obiect ce este extins dintr-o clasa sa suprascrie o functie din clasa initiala.

# Suprascrierea unei metode: override

```
package geometrie
{
    public class Patrati extends Dreptunghi
    {
        private var _latime:ui16
        public function Patrati(l:ui16, l2:ui16):void
        {
            super( latura, latime)
        }
        public function set latura(l:ui16):void
        {
            if (l>0) {latime = l; lungime = l;}
        }
        public override function toString():String
        {
            return( "Patrati" );
        }
    }
}
```

Totusi pot face asta deoarece ma lasa clasa Dreptunghi.  
Daca in Dreptunghi metoda **toString** ar fi fost declarata de tip final, voi produce o eroare...

# Daca ar fi asa... s-ar produce o eroare

```
public final function toString():String  
{  
    return ("Dreptunghi");  
}
```

Dreptunghi

14  
15  
16  
17

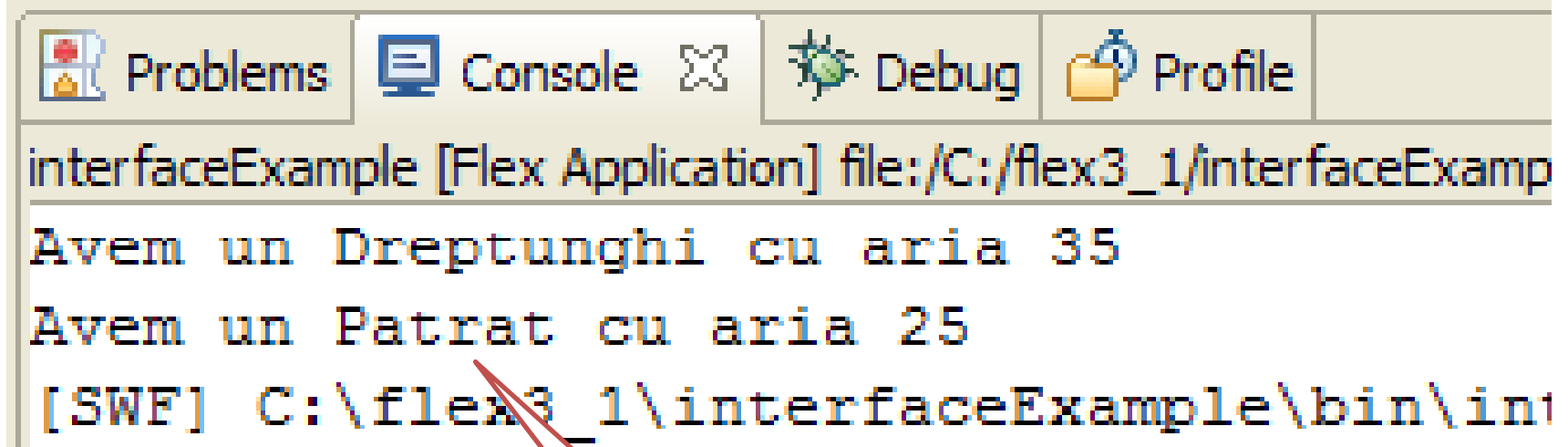
```
public override function toString():String  
{  
    return( "Patrat" );  
}
```

Patrat

The screenshot shows the IDE's interface with the 'Problems' tab selected. It displays a table of errors. A red arrow points from the 'override' keyword in the code above to the error message in the table.

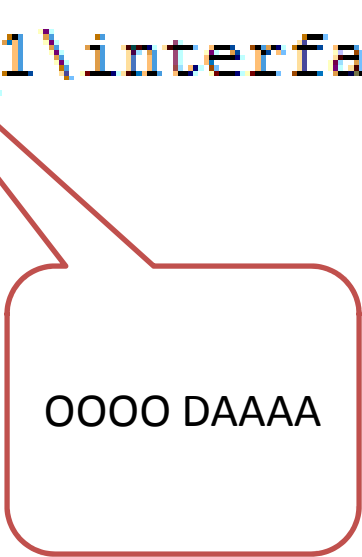
Description	Resource	Path	Location
Errors (1 item)			
1025: Cannot redefine a final method.	Patrat.as	interfaceExample/geometrie	line 14

# Revenim la programul principal...



The screenshot shows an IDE console window with a toolbar at the top containing 'Problems', 'Console', 'Debug', and 'Profile' buttons. The console text is as follows:

```
interfaceExample [Flex Application] file:/C:/flex3_1/interfaceExamp  
Avem un Dreptunghi cu aria 35  
Avem un Patrat cu aria 25  
[SWF] C:\flex3_1\interfaceExample\bin\int
```



Oooo DAAAA

# Clasa Cerc?!?!

```
package geometrie
{
    public class Cerc
    {
        private var _raza:uint;
        public function Cerc( raza:uint = 0 ):void
        {
            _raza = raza;
        }
        public function aria():Number
        {
            return( Math.PI * Math.pow( _raza, 2) );
        }
        public function toString():String
        {
            return( "Cerc" );
        }
    }
}
```

Eu nu mostenesc functiile **aria**,  
**toString**, le voi rescrie...

# Din nou in aplicatie...

```
package {
    import flash.display.Sprite;

    import geometrie.*;

    public class interfaceExample extends Sprite
    {
        public function interfaceExample()
        {
            var dreptunghi : Dreptunghi = new Dreptunghi( 5, 7 );
            var patrat : Patrat = new Patrat( 5 );
            var cerc : Cerc = new Cerc( 5 )
            trace("Avem un " + dreptunghi.toString() + " cu aria " +
                dreptunghi.aria);
            trace("Avem un " + patrat.toString() + " cu aria " + patrat.aria);
            trace("Avem un " + cerc.toString() + " cu aria " + cerc.aria);
        }
    }
}
```

# Problema...

- Toate obiectele au o metoda toString si una aria.
- Daca stiu ca toate obiectele geometrice au aceste metode, as putea face o functie care sa primeasca parametru un obiect geometric si sa poata sa apeleze toString sau aria ?

[ca sa nu scriu de 3 ori `trace( "Obiectul"... );`]

# Problema...

- Probabil ca s-ar putea [altfel nu si-ar bate nimeni capul cu creatul de sliduri inutile:D]
- Se poate... dar atunci ce tip de obiect va primi functia ca parametru ?!

# Problema...

- Nu primește un tip de obiect ci o **Interfata**.
- “**Interfata** este un dispozitiv ce poate face ca două dispozitive ce nu sunt compatibile să poată comunica.”
- De exemplu, monitorul este o interfata între om și calculator. La fel sunt tastatura și mouseul.
- Aici este vorba de o interfata pentru o funcție...

# Interfata...

- Intr-o interfata se declara metodele publice care ar putea fi utilizate de functia ce primeste parametru acea interfata (numai declaratia nu si codul).
- Catre functia ce primeste parametru de tip interfata, nu pot fi trimise decat obiecte ce implementeaza acea interfata

# Interfata...

- O clasa implementeaza o interfata daca contine toate metodele ce sunt precizate in interfata si in plus are antetul de forma:

```
public class <NumeClasa> implements <NumeInterfata>
```

- Numele interfetei va fi scris mereu inceand cu litera I mare (i mare) si va toate cuvintele din nume vor incepe cu litera mare. (De exemplu interfata noastra se va numi IGeom)

# Interfata...

```
package geometrie
{
    public interface IGeom
    {
        function aria():Number;
        function toString():String;
    }
}
```

# Interfata...

NU mai este clasa,  
este interfata.

```
package geometrie
{
    public interface IGeom
    {
        function aria():Number;
        function toString():String;
    }
}
```

# Interfata...

Va fi salvata in  
fisierul Igeom.as

```
package geometrie
{
    public interface IGeom
    {
        function aria():Number;
        function toString():String;
    }
}
```

# Interfata...

Din directorul  
(pachetul)  
geometrie

```
package geometrie
{
    public interface IGeom
    {
        function aria():Number;
        function toString():String;
    }
}
```

# Interfata...

In clasele ce implementeaza interfata, vom modifica antetele. Numai clasele ce implementeaza IGeom pot fi trimise ca parametru...

```
public class Cerc implements IGeom
{
    private var _raza:uint;
```

```
public class Dreptunghi implements IGeom
{
    protected var _latime:uint;
    protected var _lungime:uint;
```

```
package geometrie
{
    public class Patrat extends Dreptunghi implements IGeom
    {
        private var _latura:uint;
        public function Patrat( latura:uint)
        {
```

# Sa rescriem aplicatia principala:

```
package {  
    import flash.display.Sprite;  
    import geometrie.*;  
  
    public class interfaceExample extends Sprite  
    {  
        public function interfaceExample()  
        {  
            var dreptunghi:Dreptunghi = new Dreptunghi(5,7);  
            var patrat:IGeom = new Patrat(5);  
            arataAria(dreptunghi);  
            arataAria(patrat);  
        }  
  
        private function arataAria(figura:IGeom):void  
        {  
            trace("Avem un " + figura + " cu aria " + figura.aria());  
        }  
    }  
}
```

# Sa rescriem aplicatia principala:

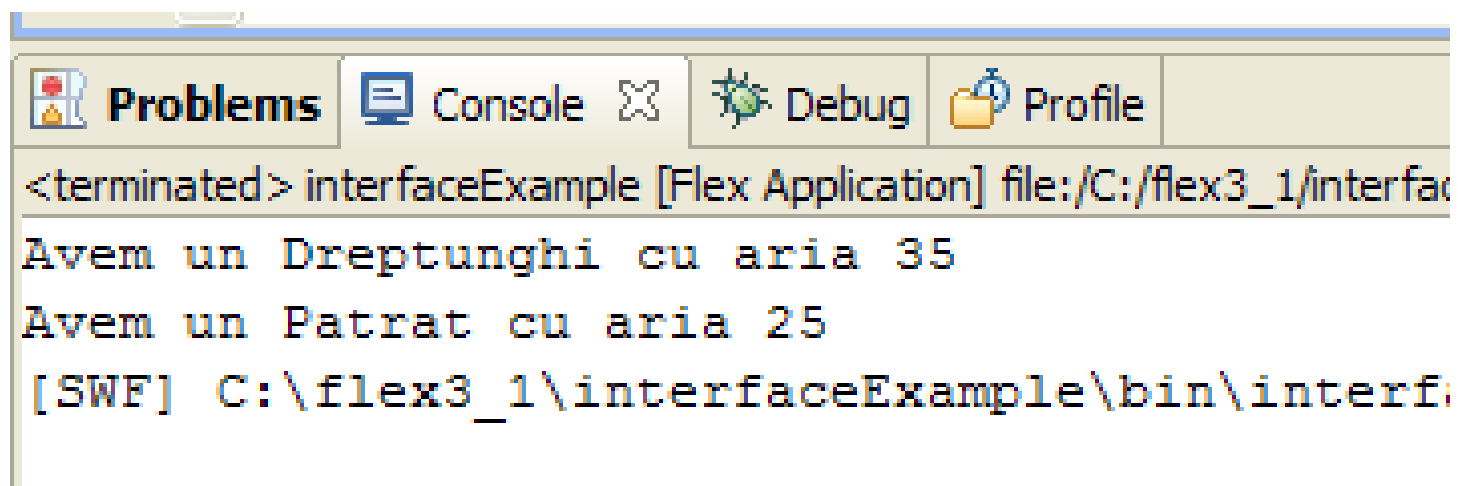
```
package {
    import flash.display.Sprite;
    import geometrie.*;

    public class interfaceExample extends Sprite
    {
        public function interfaceExample()
        {
            var dreptunghi:Dreptunghi = new Dreptunghi(10,10);
            var patrat:IGeom = new Patrat(10);
            arataAria(dreptunghi);
            arataAria(patrat);
        }

        private function arataAria(figura:IGeom):void
        {
            trace("Avem un " + figura + " cu aria " + figura.aria());
        }
    }
}
```

Stiati ca metoda **toString** este apelata automat daca obiectul este tratat ca un sir de caractere ? Din acest motiv, metoda **toString** trebuie sa returneze **String** !

# Evident, rezultatul:



The screenshot shows a console window from an IDE. The window title is "interfaceExample [Flex Application] file:/C:/flex3\_1/interfac". The console output is as follows:

```
<terminated> interfaceExample [Flex Application] file:/C:/flex3_1/interfac  
Avem un Dreptunghi cu aria 35  
Avem un Patrat cu aria 25  
[SWF] C:\flex3_1\interfaceExample\bin\interf
```

# Exemplu cu imagini...

## Clasa ce incarca o imagine....

O gramada de importuri.... Oricum Flex le face singur 😊

```
package classes
{
    import flash.display.Bitmap;
    import flash.display.Loader;
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.IOErrorEvent;
    import flash.events.ProgressEvent;
    import flash.net.URLRequest;
    import flash.system.LoaderContext;
```

# Scheletul clasei GraphicImage

```
public dynamic class GraphicImage extends Sprite
{
    public var imageURL : String = new String();
    public var bitmap    : Bitmap;
    public var loaded    : Boolean = false;
    public var succes    : Boolean = false;
    public function GraphicImage( url:String )
    {
    }
    public function loadImage():void
    {
    }
    private function loadCompleted(e:Event):void
    {
    }
    private function progress(e:ProgressEvent):void
    {
    }
    private function errorLoading(e:IOErrorEvent):void
    {
    }
}
```

# Scheletul clasei GraphicImage

Va tine minte unde se afla imaginea (calea din Inernet sau locala)

```
public dynamic class GraphicImage {
    public var imageURL : String = new String();
    public var bitmap : Bitmap;
    public var loaded : Boolean = false;
    public var succes : Boolean = false;
    public function GraphicImage( url:String )
    {
    }
    public function loadImage():void
    {
    }
    private function loadCompleted(e:Event):void
    {
    }
    private function progress(e:ProgressEvent):void
    {
    }
    private function errorLoading(e:IOErrorEvent):void
    {
    }
}
```

# Scheletul clasei GraphicImage

Aici vom gasi imaginea  
dupa incarcare...

```
public dynamic class GraphicImage {
    public var imageURL : String = new String();
    public var bitmap : Bitmap;
    public var loaded : Boolean = false;
    public var succes : Boolean = false;
    public function GraphicImage( url:String )
    {
    }
    public function loadImage():void
    {
    }
    private function loadCompleted(e:Event):void
    {
    }
    private function progress(e:ProgressEvent):void
    {
    }
    private function errorLoading(e:IOErrorEvent):void
    {
    }
}
```

# Scheletul clasei GraphicImage

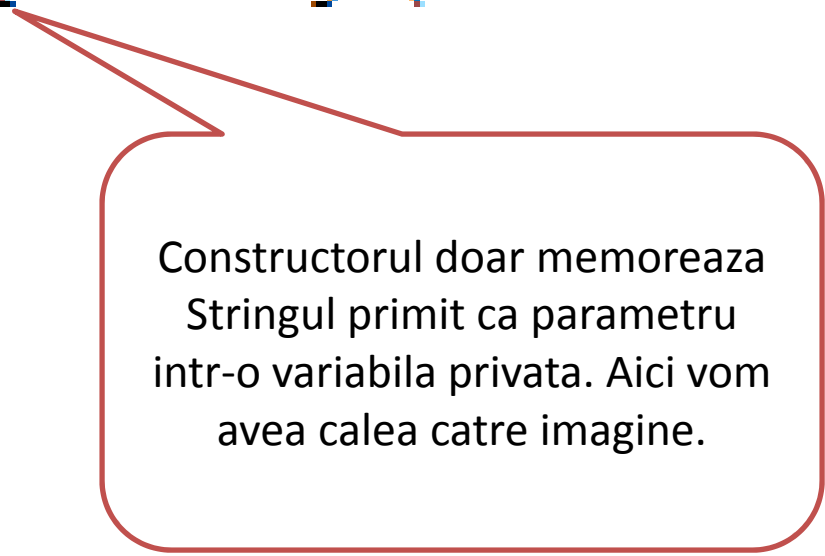
```
public dynamic class GraphicImage extends Sprite {  
    public var imageURL : String;  
    public var bitmap : Bitmap;  
    public var loaded : Boolean = false;  
    public var succes : Boolean = false;  
    public function GraphicImage( url:String )  
    {  
    }  
    public function loadImage():void  
    {  
    }  
    private function loadCompleted(e:Event):void  
    {  
    }  
    private function progress(e:ProgressEvent):void  
    {  
    }  
    private function errorLoading(e:IOErrorEvent):void  
    {  
    }  
}
```

Proprietatea load va putea fi interogata pentru a ne asigura ca imaginea s-a incarcat.(nu vrem sa afisam poze vide)

In continuare functiile....

# Clasa ce incarca o imagine: metodele:

```
public function GraphicImage( url:String )  
{  
    ImageURL = url;  
}
```



Constructorul doar memoreaza Stringul primit ca parametru intr-o variabila privata. Aici vom avea calea catre imagine.

# Clasa ce incarca o imagine: metodele :

Aici se va porni incacarea...

```
public function loadImage():void
{
    var context:LoaderContext = new LoaderContext();
    context.checkPolicyFile = true;
    var pictLdr:Loader = new Loader();
    var pictURL:String = ImageURL;
    var pictURLReq:URLRequest = new URLRequest(pictURL);
    pictLdr.load(pictURLReq, context);
    pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, loadCompleted);
    pictLdr.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS, progress);
    pictLdr.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, errorLoading);
}
```

# Clasa ce incarca o imagine: metodele :

Avem nevoie de un container pentru orice am incarca (nu numai imagini, ci si XML sau alte kestii)

```
public function loadImage():void
{
    var context:LoaderContext = new LoaderContext();
    context.checkPolicyFile = true;
    var pictLdr:Loader = new Loader();
    var pictURL:String = ImageURL;
    var pictURLReq:URLRequest = new URLRequest(pictURL);
    pictLdr.load(pictURLReq, context);
    pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, loadCompleted);
    pictLdr.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS, progress);
    pictLdr.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, errorLoading);
}
```

# Clasa ce incarca o imagine: metodele :

Ignoram asta

```
public function loadImage():void
{
    var context:LoaderContext = new LoaderContext();
    context.checkPolicyFile = true;
    var pictLdr:Loader = new Loader();
    var pictURL:String = ImageURL;
    var pictURLReq:URLRequest = new URLRequest(pictURL);
    pictLdr.load(pictURLReq, context);
    pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, loadCompleted);
    pictLdr.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS, progress);
    pictLdr.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, errorLoading);
}
```

# Clasa ce incarca o imagine: metodele :

Obiectul Loader va incarca efectiv imaginea

```
public function loadImage():void
{
    var context:LoaderContext = new LoaderContext();
    context.checkPolicyFile = true;
    var pictLdr:Loader = new Loader();
    var pictURL:String = ImageURL;
    var pictURLReq:URLRequest = new URLRequest(pictURL);
    pictLdr.load(pictURLReq, context);
    pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, loadCompleted);
    pictLdr.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS, progress);
    pictLdr.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, errorLoading);
}
```

# Clasa ce incarca o imagine: metodele :

Incarcarea incepe aici

```
public function loadImage():void
{
    var context:LoaderContext = new LoaderContext();
    context.checkPolicyFile = true;
    var pictLdr:Loader = new Loader();
    var pictURL:String = ImageURL;
    var pictURLReq:URLRequest = new URLRequest(pictURL);
    pictLdr.load(pictURLReq, context);
    pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, loadCompleted);
    pictLdr.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS, progress);
    pictLdr.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, errorLoading);
}
```

# Clasa ce incarca o imagine: metodele :

```
public function loadImage():void
{
    var context:LoaderContext = new LoaderContext();
    context.checkPolicyFile = true;
    var pictLdr:Loader = new Loader();
    var pictURL:String = ImageURL;
    var pictURLReq:URLRequest = new URLRequest(pictURL);
    pictLdr.load(pictURLReq, context);
    pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, loadCompleted);
    pictLdr.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS, progress);
    pictLdr.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, errorLoading);
}
```

Deoarece incarcarea este asincrona, nu stim cand se va termina. Cand se va intampla acest lucru, ar fi bine sa setam variabila loaded cu true si sa copiem imaginea in proprietatea bitmap

# Clasa ce incarca o imagine: metodele :

```
public function loadImage():void
{
    var context:LoaderContext = new LoaderContext();
    context.checkPolicyFile = true;
    var pictLdr:Loader = new Loader();
    var pictURL:String = ImageURL;
    var pictURLReq:URLRequest = new URLRequest(pictURL);
    pictLdr.load(pictURLReq, context);
    pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, loadCompleted);
    pictLdr.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS, progress);
    pictLdr.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, errorLoading);
}
```

Prin intermediul celorlalte doua listenere putem afla procentul din imagine incarcata respectiv momentul producerii unei erori (speram sa nu fie cazul)

# Clasa ce incarca o imagine: metodele :

```
private function loadCompleted(e:Event):void
{
    var localLoader:Loader = Loader(e.target.loader);
    bitmap = Bitmap(localLoader.content);
    this.addChild(bitmap);
    height = bitmap.height;
    width  = bitmap.width;
    loaded = true;
}
```

Cand este executata aceasta metoda (de catre evenimentul COMPLETE), stim ca imaginea s-a incarcat.

# Clasa ce incarca o imagine: metodele :

```
private function loadCompleted(e:Event):void
{
    var localLoader:Loader = Loader(e.target.loader);
    bitmap = Bitmap(localLoader.content);
    this.addChild(bitmap);
    height = bitmap.height;
    width  = bitmap.width;
    loaded = true;
}
```

Duplicam obiectul loader pentru a putea extrage informatiile din el. Am fi putut face si altfel (cu un loader pus privat care sa fie vizibil in toate metodele)

# Clasa ce incarca o imagine: metodele :

```
private function loadCompleted(e:Event):void
{
    var localLoader:Loader = Loader(e.target.loader);
    bitmap = Bitmap(localLoader.content);
    this.addChild(bitmap);
    height = bitmap.height;
    width  = bitmap.width;
    loaded = true;
}
```

Setam obiectul bitmap...  
si eventual il adaugam si  
in acest obiect (desi nu  
e neaparata nevoie)

# Clasa ce incarca o imagine: functiile:

```
private function loadCompleted(e:Event):void
{
    var localLoader:Loader = Loader(e.target.loader);
    bitmap = Bitmap(localLoader.content);
    this.addChild(bitmap);
    height = bitmap.height;
    width  = bitmap.width;
    loaded = true;
}
```

Facem variabila loaded true.

# Clasa ce incarca o imagine: functiile:

Putem afla  
progresul...

```
private function progress(e:ProgressEvent):void
{
    trace( e.bytesLoaded, e.bytesTotal );
}
private function errorLoading(e:IOErrorEvent):void
{
}
```

Sau daca s-a produs o  
eroare

Acasa veti putea studia tot codul (am pus totul la un loc desi nu este vizibil din sala)...

Nu va mai chinuiti... 😊

```
package classes
{
import flash.display.*;
import flash.events.*;
import flash.net.URLRequest;
import flash.system.LoaderContext;
public dynamic class GraphicImage extends Sprite
{
public var ImageURL : String = new String();
public var bitmap : Bitmap;
public var loaded : Boolean = false;
public var succes : Boolean = false;
public function GraphicImage( url:String )
{
ImageURL = url;
}
public function loadImage():void
{
var context:LoaderContext = new LoaderContext();
context.checkPolicyFile = true;
var pictLdr:Loader = new Loader();
var pictURL:String = ImageURL;
var pictURLReq:URLRequest = new URLRequest(pictURL);
pictLdr.load(pictURLReq, context);
pictLdr.contentLoaderInfo.addEventListener(Event.COMPLETE, loadCompleted);
pictLdr.contentLoaderInfo.addEventListener(ProgressEvent.PROGRESS, progress);
pictLdr.contentLoaderInfo.addEventListener(IOErrorEvent.IO_ERROR, errorLoading);
}
private function loadCompleted(e:Event):void
{
var localLoader:Loader = Loader(e.target.loader);
bitmap = Bitmap(localLoader.content);
this.addChild(bitmap);
height = bitmap.height;
width = bitmap.width;
loaded = true;
}
private function progress(e:ProgressEvent):void
{
trace( e.bytesLoaded, e.bytesTotal );
}
private function errorLoading(e:IOErrorEvent):void
{
}
}
}
```

# Clasa principala (ce se foloseste de GraphicImage):

```
package {
    import classes.GraphicImage;
    import flash.display.*;
    import flash.events.MouseEvent;
    import flash.utils.setInterval;
    public class PhotoAlbum extends Sprite
    {
        private var poze : Array = new Array();
        private var legaturiPoze : Array = new Array();
        private var imagineAfisata:Bitmap = new Bitmap( new BitmapData(320, 240));
        private var curentImage:uint = 0;

        public function PhotoAlbum()
        {
        }
        private function adaugaLegaturiLaPoze():void
        {
        }
        private function schimbaPoza():void
        {
        }
    }
}
```

Deoarece e principala  
trebuie sa extind  
Sprite

```
package {
    import classes.GraphicImage;
    import flash.display.*;
    import flash.events.MouseEvent;
    import flash.utils.setInterval;
    public class PhotoAlbum extends Sprite
    {
        private var poze : Array = new Array();
        private var legaturiPoze : Array = new Array();
        private var imagineAfisata:Bitmap = new Bitmap( new BitmapData(320, 240));
        private var curentImage:uint = 0;

        public function PhotoAlbum()
        {
        }
        private function adaugaLegaturiLaPoze():void
        {
        }
        private function schimbaPoza():void
        {
        }
    }
}
```

```
package {
    import classes.GraphicImage;
    import flash.display.*;
    import flash.events.MouseEvent;
    import flash.utils.setInterval;
    public class PhotoAlbum extends Sprite
    {
        private var poze : Array = new Array();
        private var legaturiPoze : Array = new Array();
        private var imagineAfisata:Bitmap = new Bitmap( new BitmapData(320, 240));
        private var curentImage:uint = 0;

        public function PhotoAlbum()
        {
        }
        private function adaugaLegaturiLaPoze():void
        {
        }
        private function schimbaPoza():void
        {
        }
    }
}
```

Aici vor fi chiar  
obiectele de tip  
GraphicImage

In timp ce aici vor fi  
retinute doar  
legaturile

```
package {
    import classes.GraphicImage;
    import flash.display.*;
    import flash.events.MouseEvent;
    import flash.utils.setInterval;
    public class PhotoAlbum extends Sprite
    {
        private var poze : Array = new Array();
        private var legaturiPoze : Array = new Array();
        private var imagineAfisata:Bitmap = new Bitmap( new BitmapData(320, 240));
        private var curentImage:uint = 0;

        public function PhotoAlbum()
        {
        }
        private function adaugaLegaturiLaPoze():void
        {
        }
        private function schimbaPoza():void
        {
        }
    }
}
```

Un container ce va fi afisat pe ecran si in care vom copia de fiecare data bitii din imaginile incarcate...

```
package {
    import classes.GraphicImage;
    import flash.display.*;
    import flash.events.MouseEvent;
    import flash.utils.setInterval;
    public class PhotoAlbum extends Sprite
    {
        private var poze : Array = new Array();
        private var legaturiPoze : Array = new Array();
        private var imagineAfisata:Bitmap = new Bitmap( new BitmapData(320, 240));
        private var curentImage:uint = 0;

        public function PhotoAlbum()
        {
        }
        private function adaugaLegaturiLaPoze():void
        {
        }
        private function schimbaPoza():void
        {
        }
    }
}
```

Pentru a sti la ce imagine suntem si ce urmeaza, utilizam acesta proprietate

```
package {
    import classes.GraphicImage;
    import flash.display.*;
    import flash.events.MouseEvent;
    import flash.utils.setInterval;
    public class PhotoAlbum extends Sprite
    {
        private var poze : Array = new Array();
        private var legaturiPoze : Array = new Array();
        private var imagineAfisata:Bitmap = new Bitmap( new BitmapData(320, 240));
        private var curentImage:uint = 0;

        public function PhotoAlbum()
        {
        }
        private function adaugaLegaturiLaPoze():void
        {
        }
        private function schimbaPoza():void
        {
        }
    }
}
```

```
package {  
    import classes.GraphicImage;  
    import flash.display.*;  
    import flash.events.MouseEvent;  
    import flash.utils.setInterval;  
    public class PhotoAlbum extends Sprite  
    {  
        private var poze : Array = new Array();  
        private var legaturiPoze : Array = new Array();  
        private var imagineAfisata:Bitmap = new Bitmap( new BitmapData(320, 240));  
        private var curentImage:uint = 0;  
  
        public function PhotoAlbum()  
        {  
        }  
        private function adaugaLegaturiLaPoze():void  
        {  
        }  
        private function schimbaPoza():void  
        {  
        }  
    }  
}
```

Avem doar trei metode...

Creem sirul de  
legaturi...

```
public function PhotoAlbum()  
{  
    adaugaLegaturiLaPoze();  
    for(var i:uint = 0; i<legaturiPoze.length; i++)  
    {  
        poze.push( new GraphicImage(legaturiPoze[i] ) );  
        poze[poze.length-1].loadImage();  
    }  
    addChild( imagineAfisata );  
    setInterval(schimbaPoza, 3000);  
}
```

Rusinos dar adevarat...



```
private function adaugaLegaturiLaPoze():void
{
    legaturiPoze.push( "Poze/acvariul_nostru.jpg" );
    legaturiPoze.push( "Poze/alti_nori.jpg" );
    legaturiPoze.push( "Poze/asfintit.jpg" );
    legaturiPoze.push( "Poze/buburuza.jpg" );
    legaturiPoze.push( "Poze/baturuga.jpg" );
    legaturiPoze.push( "Poze/chitzuuuu.jpg" );
}
```

Pentru fiecare legatura,  
construim obiectul de tip  
GraphicImage cu legatura  
curenta si ii ordonam sa incarce  
imaginea

```
public function PhotoAlbum()  
{  
    adaugaLegaturiLaPoze();  
    for(var i:uint = 0; i<legaturiPoze.length; i++)  
    {  
        poze.push( new GraphicImage(legaturiPoze[i] ) );  
        poze[poze.length-1].loadImage();  
    }  
    addChild( imagineAfisata );  
    setInterval(schimbaPoza, 3000);  
}
```

Nu uitam sa punem in scena imaginea ce va juca rol de container pentru pozele incarcate...

```
public function PhotoAlbum()  
{  
    adaugaLegaturiLaPoze();  
    for(var i:uint = 0; i < legaturiPoze.length; i++)  
    {  
        poze.push( new GraphicImage(legaturiPoze[i] ) );  
        poze[poze.length-1].loadImage();  
    }  
    addChild( imagineAfisata );  
    setInterval(schimbaPoza, 3000);  
}
```

Si tot in constructor vom ordona  
executarea metodei  
schimbaPoza (care va dati  
seama ce face) la fiecare  
3000ms = 3s

```
public function PhotoAlbum()  
{  
    adaugaLegaturiLaPoze();  
    for(var i:uint = 0; i<legaturiPoze.length; i++)  
    {  
        poze.push( new GraphicImage(legaturiPoze[i] ) );  
        poze[poze.length-1].loadImage();  
    }  
    addChild( imagineAfisata );  
    setInterval( schimbaPoza, 3000 );  
}
```

Metoda ce schimba poza va fi executata la fiecare 3000ms

```
private function schimbaPoza():void
{
    if( poze[curentImage].loaded )
    {
        imagineAfisata.bitmapData = poze[curentImage].bitmap.bitmapData.clone();
        curentImage++;
        curentImage%=legaturiPoze.length;
    }
}
```

Vom schimba poza numai daca suntem siguri ca s-a incarcat

```
private function schimbaPoza():void
{
    if( poze[curentImage].loaded )
    {
        imagineAfisata.bitmapData = poze[curentImage].bitmap.bitmapData.clone();
        curentImage++;
        curentImage%=legaturiPoze.length;
    }
}
```

Pentru a duplica poza copiem bytes-ii aflati in proprietatea bitmapData de tip BitmapData din obiectul grafic in container

```
private function schimbaPoza():void
{
    if( poze[curentImage].loaded )
    {
        imagineAfisata.bitmapData = poze[curentImage].bitmap.bitmapData.clone();
        curentImage++;
        curentImage%=legaturiPoze.length;
    }
}
```

Daca nu am fi folosit clone(), nu am copia efectiv bytes-ii din sursa in destinatie ci am fi facut obiectul imagineAfisata.bitmapData sa pointeze catre bitmapData-ul obiectului grafic.

**Pro:** ar fi fost mai rapid fara clone.

**Contra:** fara clone, daca am modifica imagineAfisata s-ar fi modificat si obiectul graphicImage din poze[...]

```
private function schimbaPoza():void
{
    if( poze[curentImage].loadComplete )
    {
        imagineAfisata.bitmapData = poze[curentImage].bitmap.bitmapData.clone();
        curentImage++;
        curentImage%=legaturiPoze.length;
    }
}
```

In final trecem la urmatoarea imagine ce poate fi afisata

```
private function schimbaPoza() :  
{  
    if( poze[curentImage].length > 0 )  
    {  
        imagineAfisata.bitmapData = poze[curentImage].bitmap.bitmapData.clone();  
        curentImage++;  
        curentImage%=legaturiPoze.length;  
    }  
}
```

Sau la prima data deja am ajuns la sfarsit

```
private function schimbaPoza():void
{
    if( poze[curentImage].loaded )
    {
        imagineAfisata.bitmapData = poze[curentImage].bitmap.bitmapData.clone();
        curentImage++;
        curentImage%=legaturiPoze.length;
    }
}
```

larasi codul  
gramada...

```
package {
import classes.GraphicImage;
import flash.display.Bitmap;
import flash.display.BitmapData;
import flash.display.Sprite;
import flash.events.MouseEvent;
import flash.utils.setInterval;
public class PhotoAlbum extends Sprite
{
    private var poze : Array = new Array();
    private var legaturiPoze : Array = new Array();
    private var imagineAfisata:Bitmap = new Bitmap( new BitmapData(320, 240));
    private var curentImage:uint = 0;
    public function PhotoAlbum()
    {
        adaugaLegaturiLaPoze();
        for(var i:uint = 0; i<legaturiPoze.length; i++)
        {
            poze.push( new GraphicImage(legaturiPoze[i] ) );
            poze[poze.length-1].loadImage();
        }
        addChild( imagineAfisata );
        setInterval(schimbaPoza, 3000);
    }
    private function adaugaLegaturiLaPoze():void
    {
        legaturiPoze.push( "Poze/acvariul_nostru.jpg" );
        legaturiPoze.push( "Poze/alti_nori.jpg" );
        legaturiPoze.push( "Poze/asfintit.jpg" );
        legaturiPoze.push( "Poze/buburuza.jpg" );
        legaturiPoze.push( "Poze/buturuga.jpg" );
        legaturiPoze.push( "Poze/chitzuuuu.jpg" );
    }
    private function schimbaPoza():void
    {
        if( poze[curentImage].loaded )
        {
            imagineAfisata.bitmapData = poze[curentImage].bitmap.bitmapData;
            curentImage++;
            curentImage%=legaturiPoze.length;
        }
    }
}
}
```

# Concluzii...

- O imagine poate fi atasata cu addChild unui obiect de tip Sprite. Atentie, faceti acest lucru numai dupa ce sunteti SIGURI ca imaginea s-a incarcat (de preferinta in metoda ce trateaza producerea evenimentului Event.COMPLETE)
- Imaginile sunt pastrate in obiecte de tip Bitmap iar pentru a avea acces la octetii ce formeaza imaginea, puteti apela la bitmapData ce este un obiect de tip BitmapData din cadrul Bitmap. (de fapt un Array de octeti)

# Concluzii...

- Se utilizeaza clone() pentru a copia o imagine in alta (prin atribuire nu veti modifica decat referinta primului obiect la adresa celui deal doilea)
- Imaginile se incarca asincron de pe server. Nu incercati sa utilizati imaginea imediat dupa ce ati executat comanda load. Va genera o eroare.

# Tema1:

- Creati un joc in care diverse obiecte sa cada din partea superioara spre partea inferioara a ecranului. Obiectele vor fi imagini (preferabil in format PNG pentru ca au transparenta) atasate Sprite-urilor, modul de deplasare a sprite-urilor fiind cunoscut din prezentarea anterioara.
- Impuscand obiectele acestea vor disparea...

## Tema2:

- Desenati intr-un obiect de tip Sprite diverse elemente grafice (utilizati proprietatea `graphic` al sprite-ului) dupa care incercati sa modificati proprietatile obiectului. Cautati toate proprietatile Spriteului la adresa:

<http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/>

**Ca sa nu uitati, trebuie sa lucrati...**

# Intrebari ?!?!