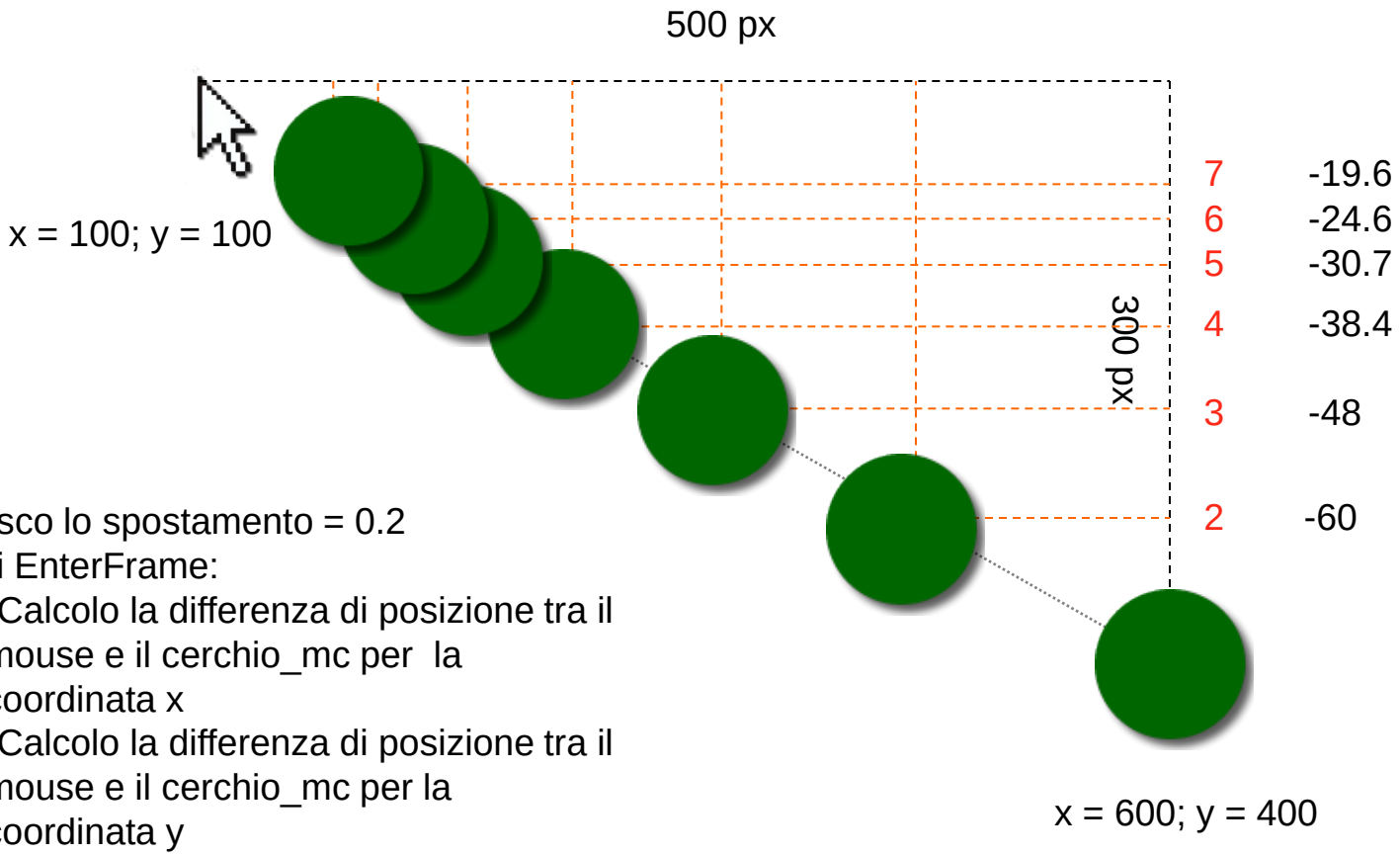
CREAZIONE DI CLASSI

Spostamento: **0.2** della distanza



- Definisco lo spostamento = 0.2
- A ogni EnterFrame:
  - Calcolo la differenza di posizione tra il mouse e il cerchio_mc per la coordinata x
  - Calcolo la differenza di posizione tra il mouse e il cerchio_mc per la coordinata y
  - Aggiorno la posizione di cerchio aggiungendo un quinto della distanza calcolato alle sue coordinate

SEGUIMI

```
import flash.display.*;
import flash.events.*;

function aggiornaPosizione (e:Event) {
   cerchio_mc.x = cerchio_mc.x +
                  (cerchio_mc.parent.mouseX -
                   cerchio_mc.x) * 0.2;
   cerchio_mc.y = cerchio_mc.y +
                  (cerchio_mc.parent.mouseY -
                   cerchio_mc.y) * 0.2;
}

cerchio_mc.addEventListener(Event.ENTER_FRAME,
                            aggiornaPosizione);
```

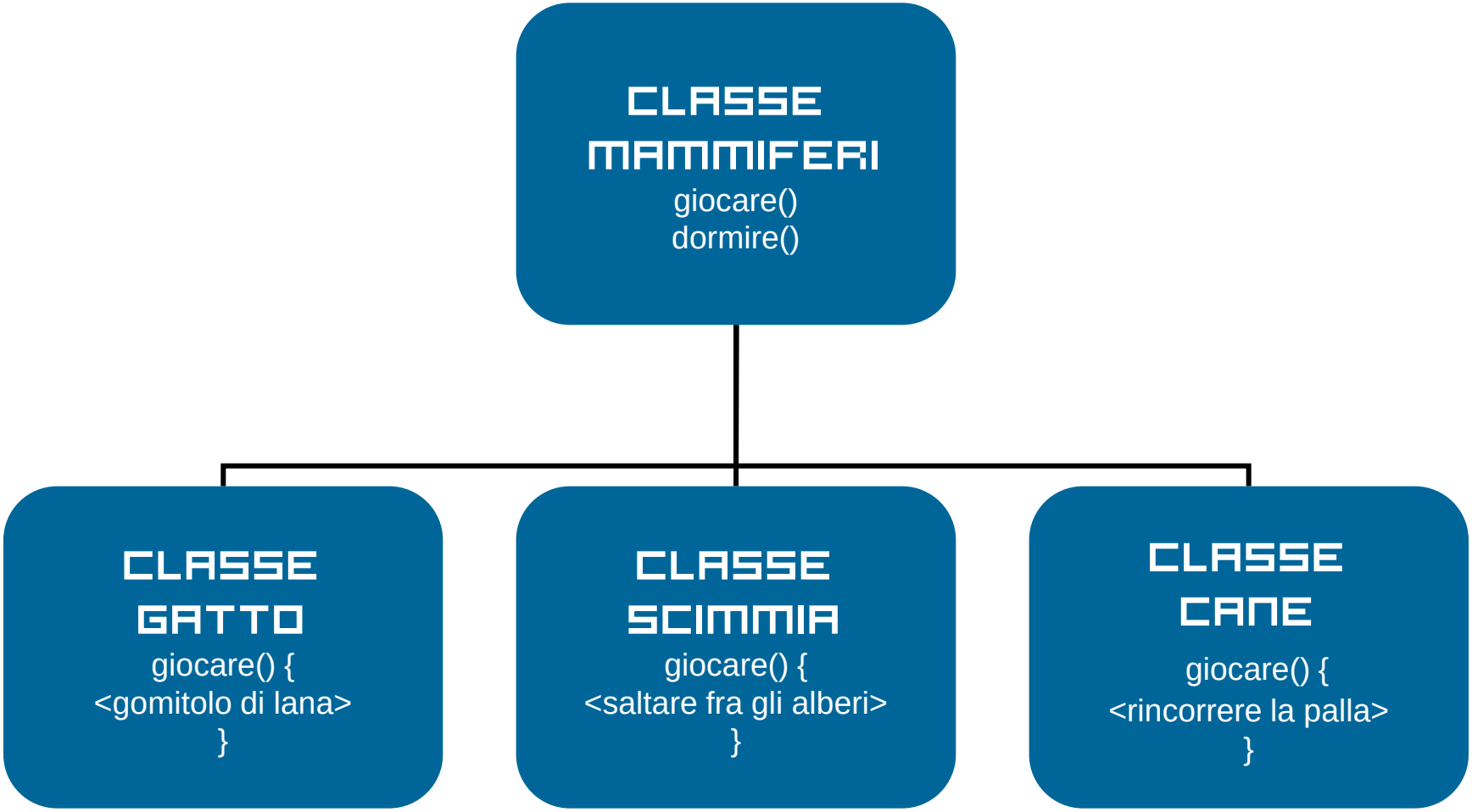
EREDITARIETÀ

MAMMIFERI

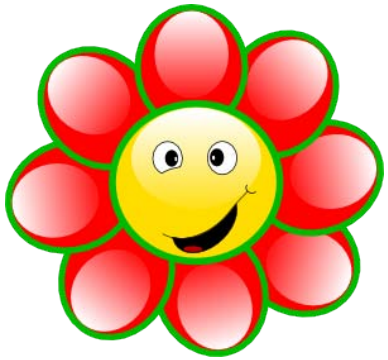
GATTO

SIAMESE

POLIMORFISMO



LA CLASSE FIORE



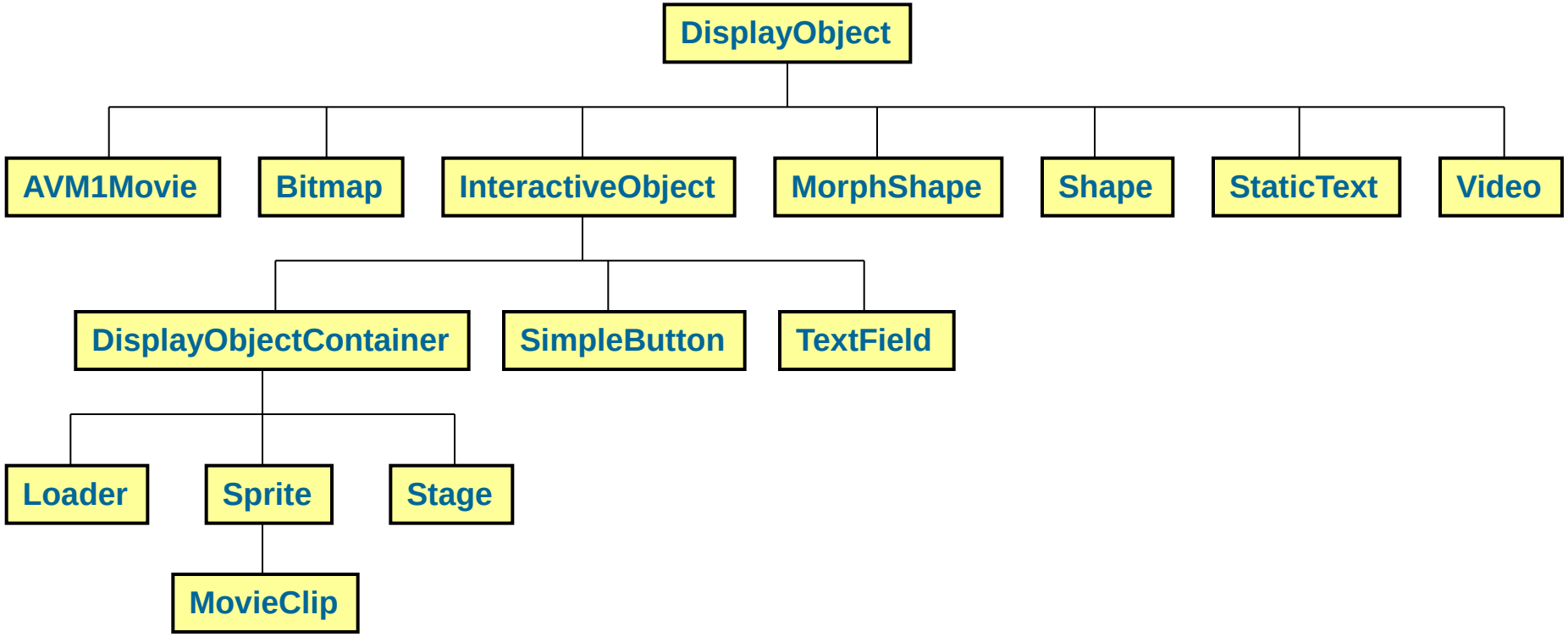
fiore

margherita

viola

rosa

LA CLASSE DISPLAYOBJECT



LA CLASSE DISPLAYOBJECT

- La classe **DisplayObject** che appartiene al package **flash.display** è l'oggetto da cui discendono le classi che mi consentono di gestire attraverso **ActionScript** tutti gli elementi visuali che possono essere mostrati in un filmato flash:
  - Testi.
  - Grafica vettoriale.
  - Immagini BitMapped.
  - Animazioni create runtime.
  - Caricamento e visualizzazione di filmati flash
  - Caricamento e visualizzazione di video.
  - Ecc.

LA CLASSE SEGUIMI

```
package {
import flash.display.*;
import flash.events.*;

public class Seguimi {
        private var dObject:DisplayObject;
        private var accelerazione:Number;
        private var contenitore:DisplayObjectContainer;

        public function Seguimi (obj:DisplayObject,e:Number) {
                ease = e;
                dObject = obj;
                contesto = dObject.parent;
        }

        private function objPos (e:Event) {
                dObject.x = dObject.x + (contesto.mouseX - dObject.x) * ease;
                dObject.y = dObject.y + (contesto.mouseY - dObject.y) * ease;
        }

        public function startFollow () {
                dObject.addEventListener(Event.ENTER_FRAME, objPos);
        }

        public function stopFollow () {
                dObject.removeEventListener(Event.ENTER_FRAME, objPos);
        }
}
}
```

LA CLASSE SEGUIMI

```
package {
  import flash.display.*;
  import flash.events.*;
  public class Seguimi {
     //definizione della classe
     ...
     ...
     ...
  }
}
```

LA CLASSE SEGUIMI

```
package {
   ......
   public class Seguimi {
        // Proprietà della classe
        private var dObject: DispalyObject;
        private var accelerazione:Number;
        private var contenitore:DisplayObjectContainer;
   }
}
```

LA CLASSE SEGUIMI

```
public class Seguimi {
..
/* Metodo "Constructor" : viene chiamato
automaticamente quando si crea una instanza della
classe */
    public function Seguimi (obj:DisplayObject,
                                        e:Number) {
            ease = e;
            dObject = obj;
            contesto = dObject.parent;
    }
..
}
```

LA CLASSE SEGUIMI

```
public class Seguimi {

..
    // Metodo che aggiorna la posizione
    // dell'oggetto
    private function objPos (e:Event) {
          dObject.x = dObject.x + (contesto.mouseX
                - dObject.x) * ease;
          dObject.y = dObject.y + (contesto.mouseY
                - dObject.y) * ease;
    }

}
```

LA CLASSE SEGUIMI

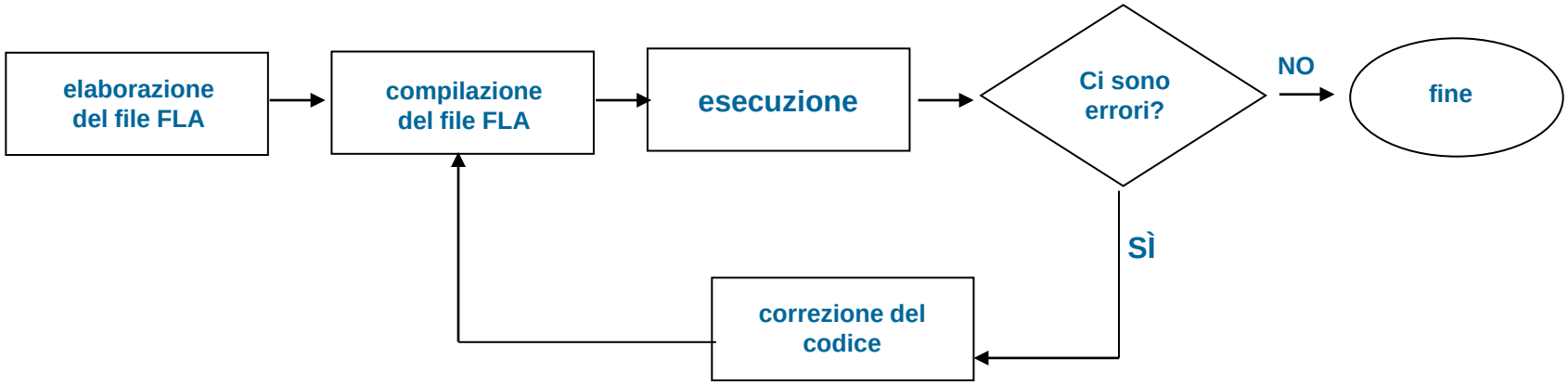
```
public class Seguimi {
..
    // Metodi che iniziano e terminano l'animazione
    // dell'oggetto
    public function startFollow () {
           dObject.addEventListener(Event.ENTER_FRAME,
                                    objPos);
    }

    public function stopFollow () {
          dObject.removeEventListener(Event.ENTER_FRAME,
                                      objPos);
    }

}
```

IL PROCESSO

- Come abbiamo gia detto il processo di realizzazione di un progetto flash è strutturata in un flusso simile a questo:



- Con la programmazione con le classi introduciamo un nuovo tipo di file: il file **Action Script (.as)** ma il processo rimane lo stesso.
- Se il progetto comprende anche dei file .as il compilatore li unirà ai dati contenuti nel .fla e creerà un unico file shockwave (filmato flash compilato e compresso).

I FILE DI CLASSE

- Il file .as è un normale file di testo (tipo blocco note) che contiene codice Action Script.
- Una classe in *ActionScript* viene sempre definita in un file esterno (un normale file di testo con estensione *.as*) che ha lo stesso nome della classe e che viene chiamato *file di classe*.
- Quando un filmato flash viene compilato (utilizzando Controllo > Prova filmato o File > Pubblica) per generare il file *.swf*, il codice contenuto nei file di classe necessari viene compilato e aggiunto al file *.swf*..

I FILE DI CLASSE

- Quando il compilatore trova che nel filmato da compilare viene utilizzata una classe **DEVE** trovare il file che contiene il codice relativo a quella classe:
- Il file di classe deve avere esattamente lo stesso nome della classe (case sensitive).
- Il compilatore deve sapere in che cartelle cercare.

I FILE DI CLASSE

1. Il compilatore in primo luogo inizierà la sua ricerca dalla cartella in cui è stato salvato il file .fla.
2. Esiste un elenco globale di cartelle che contengono classi che si può modificare andando in: Modifica>Preferenze>ActionScript e scegliendo il bottone “Impostazioni Action Script 3”

LE CLASSI AGGIUNTIVE

- Queste impostazione definiscono le cartelle di partenza dell'organizzazione delle classi
- Le classi sono organizzate in sottocartelle.
- In java e in Action Script le sottocartelle in cui sono organizzate le classi si chiamano **pakages** (pacchetti).

I PACKAGES STANDARD DI FLASH CS3

I PACKAGES STANDARD DI FLASH CS3

- Vedi **Help di Flash**:
  - Guida di riferimento al Linguaggio e alla Componenti di ActionScript 3.0 -> Tutti i packages
- Oppure **Programmare con ActionScript 3.0**
  - Pagina 56

IL COMANDO IMPORT

- Quanado utilizzo un file di classe lo devo comunicare al compilatore utilizzando il comando **import**.

```
import Seguimi
var s:Seguimi = new Seguimi(oggetto, 0.2);
```

IL FILE DI CLASSE

- Per definire una classe devo creare un file **actionScript** esterno.
- Prima di tutto dovrò definire il package a cui appartiene la classe
- Se il file di classe risiede nella stessa cartella il cui risiede il progetto userò un packege anonimo
- Altrimenti specificherò un package che ricalca il percorso in cui il file è memolrizzato

```
package {
  public class Seguimi {
    ...
  }
}
```

IL FILE DI CLASSE

- Dovrò poi inserire i comandi import necessari.
- I comandi import dovranno essere definiti **prima** della definizione di classe
- Dovro definire i comandi import sia pe le classi flash che per quelle da me definite

```
package {
  import flash.display.*;
  import flash.events.*;
  public class Seguimi {
    ...
  }
}
```

IL FILE DI CLASSE

- Inserirò quindi le dichiarazioni delle proprietà della classe
- Le proprietà di una classe sono variabili e vanno dichiarate nell stesso modo
- La dichiarazione è preceduta da un attributo che ne regola il **controllo di accesso** (scope)

```
package {
  ..
  public class Seguimi {
   private var dObject: DispalyObject;
   ...
  }
}
```

IL FILE DI CLASSE

- Inserirò quindi il costruttore
- Il costruttore è una funzione con attributo **public** che ha lo stesso nome della classe, viene eseguito quando viene creata un'istnaza della classe e quindi deve contenere le istruzioni di inizializzazione

```
package {
  ..
  public class Seguimi {
   ...
        public function Seguimi (obj:DisplayObject,
                                 e:Number) {
                ease = e;
                dObject = obj;
                contesto = dObject.parent;
        }
  }
}
```

FILE DI CLASSE

- Scriverò i metodi, raggruppandoli per funzionalità. Questo tipo di organizzazione dei metodi consente di migliorare la leggibilità e la chiarezza del codice.

```
package {
  ..
  public class Seguimi {
    ...
    public function startFollow () {
        dObject.addEventListener(Event.ENTER_FRAME,
                          objPos);
        }

    public function stopFollow () {
        dObject.removeEventListener(Event.ENTER_FRAME,
                          objPos);
        }
  }
}
```

PROVARE UNA CLASSE

- Per creare e usare una classe è necessario:
  - Definizione di una classe in un file di classe *ActionScript* esterno.
  - Salvataggio del file di classe nella directory specificata per il percorso della classe (o nel percorso in cui Flash cerca le classi) oppure nella stessa directory del file FLA dell'applicazione.
  - Creazione di un'istanza della classe in un altro script, ossia un documento FLA o un file di script esterno, oppure tramite creazione di una sottoclasse basata sulla classe originale.

USARE UNA CLASSE

- Per creare un'istanza di una classe *ActionScript*, si utilizza l'operatore ***new*** per richiamare la funzione di costruzione della classe. Tale funzione ha sempre lo stesso nome della classe e restituisce un'istanza della classe che generalmente viene assegnata a una variabile.

```
var s:Seguimi = new Seguimi(oggetto, 0.2);
```

- Usando l'operatore punto (.) si accede al valore di una proprietà o a un metodo di un'istanza.

```
s.startFollow();
```

GLI ATTRIBUTI DI CONTROLLO DI ACCESSO

- Gli attributi di controllo di accesso deteminano la visibiltà o scopo di classi, proprietà e metodi.
- La sintassi è la seguente
- <attributo> **class**….
- <attributo> **var** ….
- <attributo> **function** …..

GLI ATTRIBUTI DI CONTROLLO DI ACCESSO

public	Visibilità completa
internal	Visibilità limitata alle classi che si trovano nello stesso package
private	Visibilità limitata alla sola classe di appartenenza
protected	Visibilità limitata alla classe di appartenenza e alle sottoclassi

METODI E PROPRIETÀ STATICI

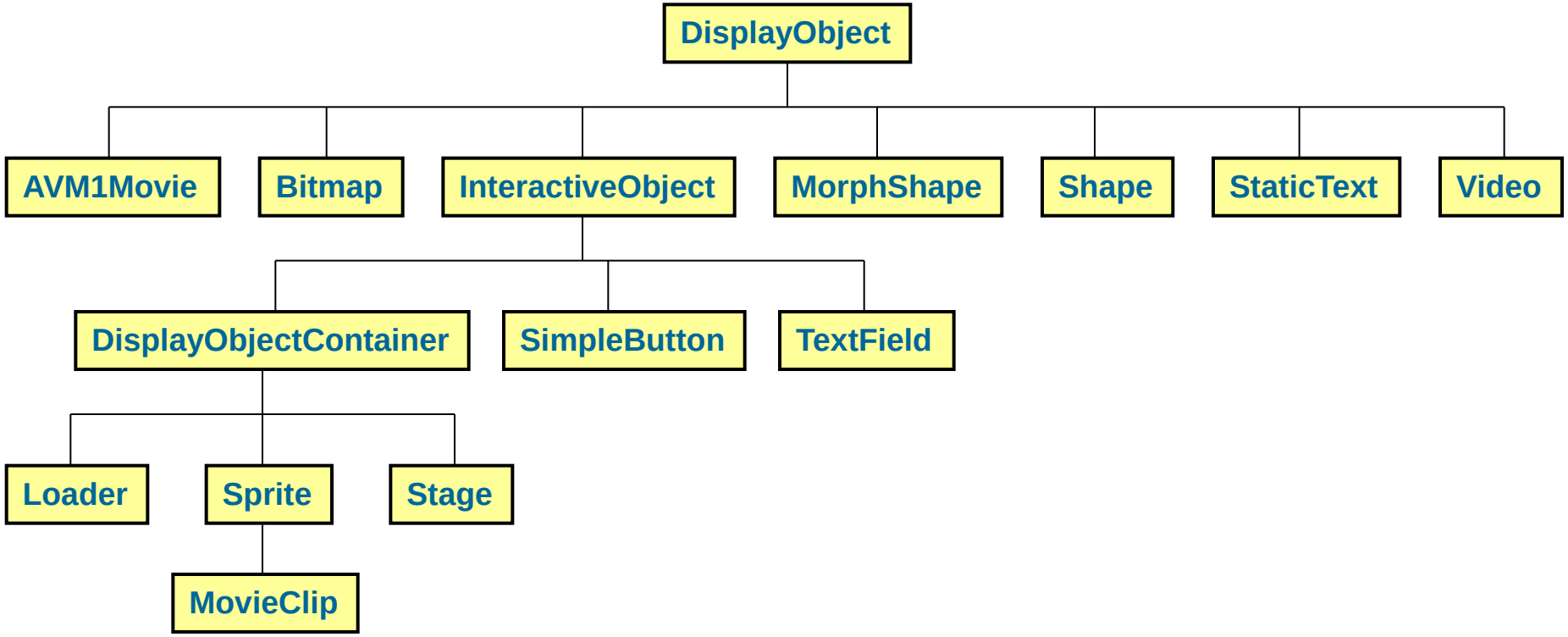
- La parola chiave *static* specifica che una variabile o una funzione viene creata solo una volta per ogni classe anziché in ogni oggetto basato sulla classe. È possibile accedere a un membro di classe statico senza creare un'istanza della classe. I metodi e le proprietà statici possono essere sia pubblici che privati.
- In ActionScript ci sono classi predefinite che hanno solo metodi e proprietà statiche.

LA PROGRAMMAZIONE VISUALE

LA CLASSE DISPLAYOBJECT

- La classe **DisplayObject** che appartiene al package **flash.display** è l'oggetto da cui discendono le classi che mi consentono di gestire attraverso **ActionScript** tutti gli elementi visuali che possono essere mostrati in un filmato flash:
  - Testi.
  - Grafica vettoriale.
  - Immagini BitMapped.
  - Animazioni create runtime.
  - Caricamento e visualizzazione di filmati flash
  - Caricamento e visualizzazione di video.
  - Ecc.

LA CLASSE DISPLAYOBJECT



DISPLAY LIST

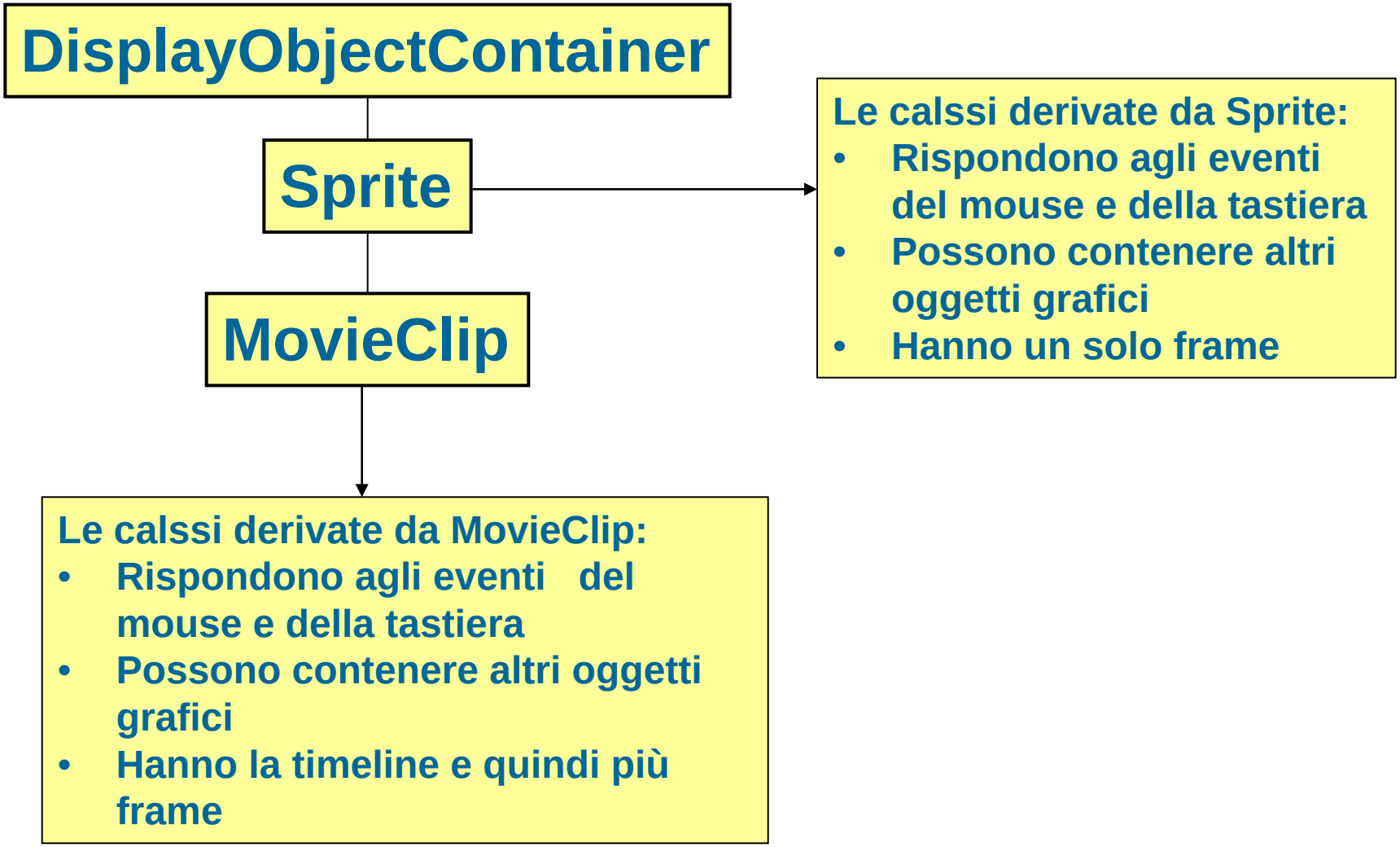
- La Display list è la struttura ad albero che contiene tutti gli elementi visuali di un filmato Flash.
- La Display List determina quali oggetti vengono visualizzati e in che ordine
- Action script può intervenire sulla Display List e quindi intervenire su cosa viene visualizzato in un filmato Flash attraverso le classi che discendono da DisplayObjectContainer.

DISPLAY LIST

- La classe **Loader** consente di gestire il caricamento in un filmato Flash di risorse esterne presenti su disco (file swf o immagini)
- Le classi Sprite e MovieClip consento di aggiungere, togliere cambiare l'ordine di visualizzazione di oggetti grafici creati run time, caricati utilizzando la classe loader, o presenti in libreria

DisplayObjectContainer

- Le classi derivate **Sprite** e **MovieClip** possono contenere e gestire la visualizzazione di qualsiasi oggetto grafico discendente da DisplayObject:
  - Oggetti semplici come TextField o Shape
  - Oggetti Loader che contegono contenuti caricati da disco
  - Discendenti di Sprite e MovieClip che a loro volta possono contenere altri oggetti.



CHILD LIST

- Le classi **Sprite** e **MovieClip** hanno metodi specifici per gestire la propria child list. Cioè l'elenco degli oggetti grafici che contengono.
- **addChild**(child:DisplayObject) aggiunge un elemento alla child listt
  - Ad ogni elemento viene assegnato un indice. Gli elementi vengono visaulizzati nell'ordine in cui sono stati aggiunto (l'ultimo risulta in primo piano)

CHILD LIST

- **addChildAt**(child:DisplayObject, index:int) aggiunge un elemento in un punto determinato della child list
- **getChildAt**(index:int):DisplayObject restituisce l'oggetto grafico che si trova al'indice specificato.
- **removeChild**(child:DisplayObject) elimina l'oggetto specificato.

DOCUMENT CLASS

- La **Document Class** è la classe che associo al filmato flash principale
- In l'istanza della classe questo caso è il filmato stesso e viene creata automaticamente in fase di compilazione.
- **Se la Document Class non è una sottoclasse di Sprite o di MovieClip la compilazione verrà interrotta da un errore.**

ESEMPIO

1. Dichiarazione di una classe facendola discendere da Sprite o da MovieClip:

```
package {
  import flash.display.Sprite;
  ......
  public class Orologio extends Sprite {
       ......
  }
}
```

ESEMPIO

2. Definizione di una o più propietà che contengano gli oggetti grafici da aggiungere alla child list:

```
.....
import flash.text.TextField
public class Orologio extends Sprite {
    private var orologio_txt:TextField;
    .....
}
```

ESEMPIO

3. Creazione degli oggetti grafici da aggiungere alla child list:

```
public class Orologio extends Sprite {
    private var orologio_txt:TextField;
    .....
    public function Orologio () {
        orologio_txt = new TextField();
        .....
    }
    .....
}
```

ESEMPIO

4. Impostazione delle proprietà degli oggetti creati:

```
.....
public function Orologio () {
 orologio_txt = new TextField();
 orologio_txt.text = "00:00:00“ ;
 orologio_txt.autoSize=
                TextFieldAutoSize.LEFT;
}
.....
```

ESEMPIO

5. Aggiunta degli oggetti creati alla child list nell'ordine desiderato

```
.....
public function Orologio () {
  .....
  addChild(orologio_txt);
  .....
}
.....
```