# CREAZIONE DI CLASSI

# LE CLASSI

- Nella programmazione orientata agli oggetti una classe definisce una *categoria di oggetti*.
- Le classi in pratica sono nuovi tipi di dati che il programmatore può creare per definire un nuovo tipo di oggetto. Una classe descrive le *proprietà* (*dati*) e i *metodi* (*comportamenti*) di un oggetto in modo molto simile a come un progetto di design descrive le caratteristiche di un oggetto, o a come una classificazione botanica descrive le caratteristiche di una pianta.
- Le proprietà (variabili definite all'interno di una classe) e i metodi (funzioni definite all'interno di una classe) di una classe sono detti *membri* della classe.
- In generale per utilizzare le proprietà e i metodi definiti da una classe, è necessario creare prima un'istanza di tale classe. La relazione tra un'istanza e la relativa classe è simile a quella che intercorre tra un oggetto reale e il relativo progetto del designer o tra una pianta reale e la sua classificazione.

# LA CLASSE BIKE

Rapporto usato

+

Diametro ruota

+

Cadenza

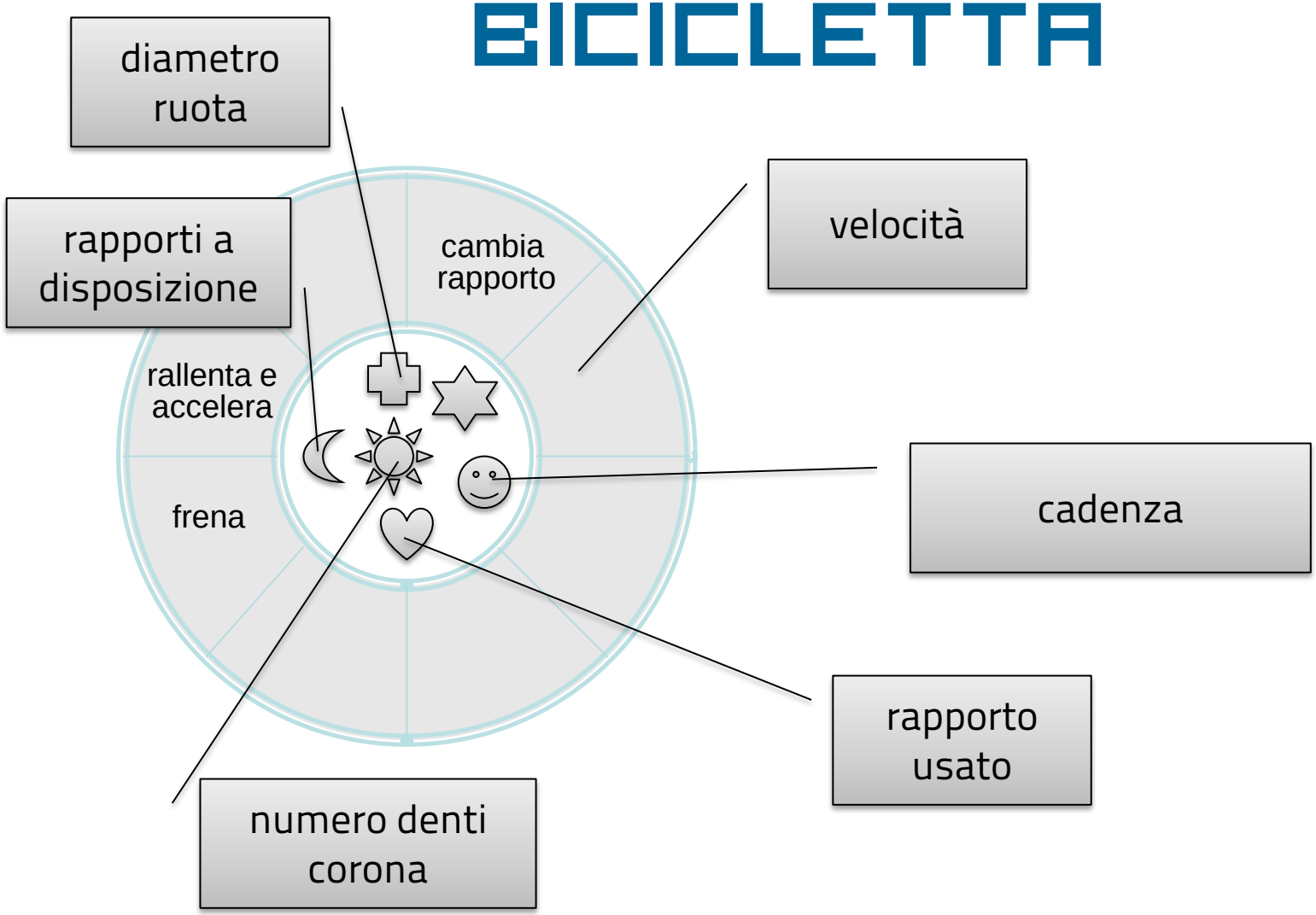
→ Velocità

diametro ruota

numero denti

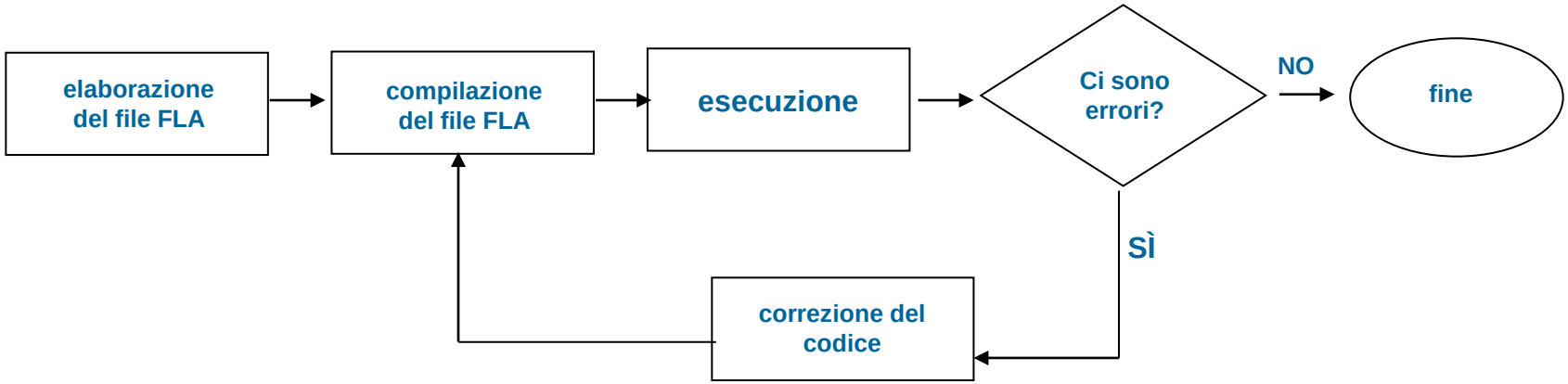
numero denti

# BICICLETTA



# IL PROCESSO

- 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"

# I PACCHETTI

- In ActionScript 3 è fondamentale capire il concetto di package.
- Con l'aggiunta del comando package (obbligatorio) per ogni classe viene definto un package di appartenenza che corrisponde a come i file di classe sono organizzati su disco
- Supponiamo che il mio progetto Bicicletta.fla si collocato in D:\ProgettiFlash\Bicicletta\

# I PACCHETTI

```
//package aninimo
package {
  public class Bike{
    ...
  }
}
```

- Il file di classe si chiamerà Bike.as e risiederà nella cartella del progetto: D:\ProgettiFlash\Bicicletta\Bike.as

# I PACCHETTI

```
//package con nome
package classiProgetto{
  public class Rosa {
    ...
  }
}
```

Il file di classe si chiamerà **Bike.as** e risiederà nella cartella classiProgetto:

**D:\ProgettiFlash\Bicicletta\classiProgetto\Bike.as**

# I PACCHETTI

```
//package con nome
package classiProgetto.uno{
  public class Rosa {
    ...
  }
}
```

Il file di classe si chiamerà **Bike.as** e risiederà nella cartella classiProgetto\uno:

**D:\ProgettiFlash\Bicicletta\classiProgetto\uno\Bike.as**

## LA CLASSE BIKE

```
pakage
{
  public class Bike
  {
      //definizione della classe
      ...
      ...
      ...
  }
}
```

## LA CLASSE ROSA

```
public class Bike{
    // Proprietà della classe
    private var rapporti:Array = [24,
                22, 20, 18, 16, 14];
    private var corona:uint = 48;
    private var diametroRuota:Number;

    public var cadenza:Number;
    public var statoCambio:uint;

}
```

## LA CLASSE ROSA

```
public class Bike{
..
/* Metodo "Constructor" : viene chiamato
  quando si crea una instanza della classe
  */
    public function Bike(misura:uint):void
    {
      diametroRuota = misura * 2.54001;
      cadenza = 0;
      statoCambio = 2;
    }
..
}
```

## LA CLASSE ROSA

```
public class Bike{
..
// Metodo per cambiare l Cadenza
 public function cambiaCadenza (diff:Number) :void
    {
          cadenza = cadenza + diff;
          if (cadenza < 0)
          {
                cadenza = 0;
          }
    }

}
```

## LA CLASSE ROSA

```
public class Bike{
..
// Metodo per ricavare la velocità
    public function speed ():Number
    {
      var passo = diametroRuota * Math.PI *
          corona / rapporti[statoCambio];
      var km_ora =  passo * cadenza * 60 /100000;
      return km_ora;
    }

}
```

# 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 bici:Bike = new Byke();
```

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

```
Bici.cambiaCadenza(-2);
```

# 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

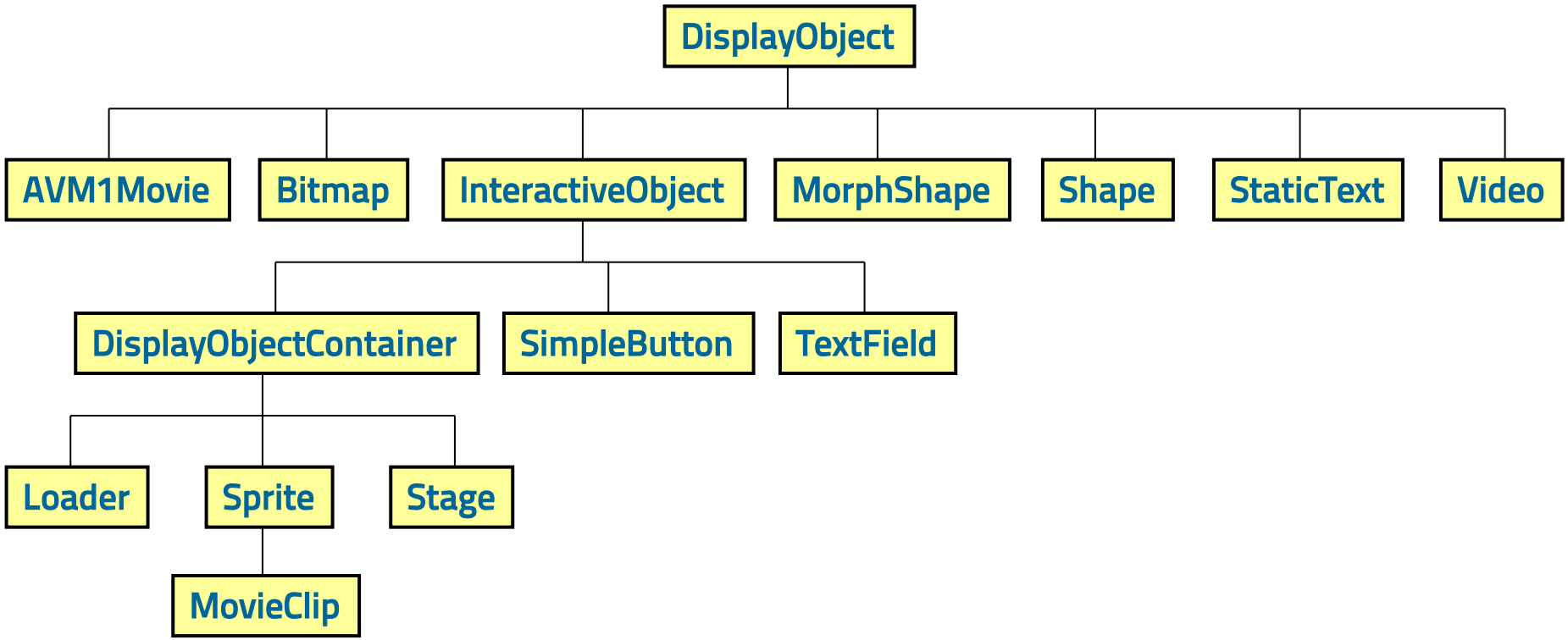
- La parola chiave *static* specifica che una variabile o una funzione viene creata solo una volta per ogni classe anziché in ogni oggetto basato sulla classe. È possibile accedere a un membro di classe statico senza creare un'istanza della classe. I metodi e le proprietà statici possono essere sia pubblici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

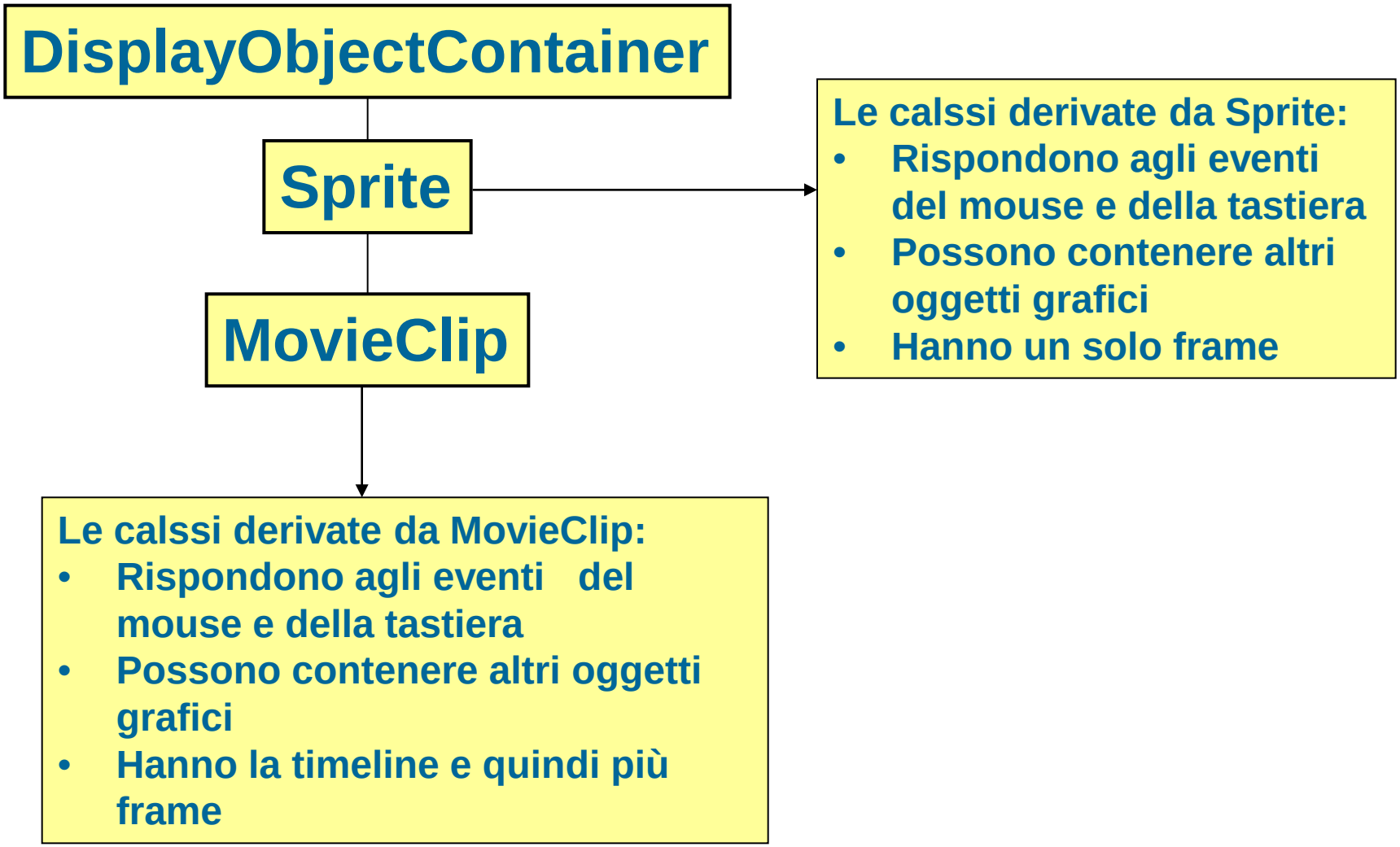
- La Display list è la struttura ad albero che contiene tutti gli elementi visuali di un filmato Flash.
- La Display List determina quali oggetti vengono visualizzati e in che ordine
- Action script può intervenire sulla Display List e quindi intervenire su cosa viene visualizzato in un filmato Flash 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);
  .....
}
.....
```

# UNA SUPERFICIE SU CUI DISEGNARE

# I METODI PER DISEGNARE

- Ogni Shape, Sprite e MovieClip ha una proprietà **graphics** che è una istanza della classe **Graphics** creata automaticamente con la creazione dell'oggetto.
- Usando i metodi offerti da **Graphics** è possibile disegnare linee e riempimenti e figure.
- I disegni vengono fatti su una superfice trasparente come su un lucido.
- Si possono usare indifferentemente uno delle tre classi. La classe Shape è di gran lunga la più efficiente.
- Per visualizzare una shape è necessario aggiungerla a una child list di uno Sprite o di una MovieClip

# DISEGNARE UN LINEA

- Questa classe disegna una linea

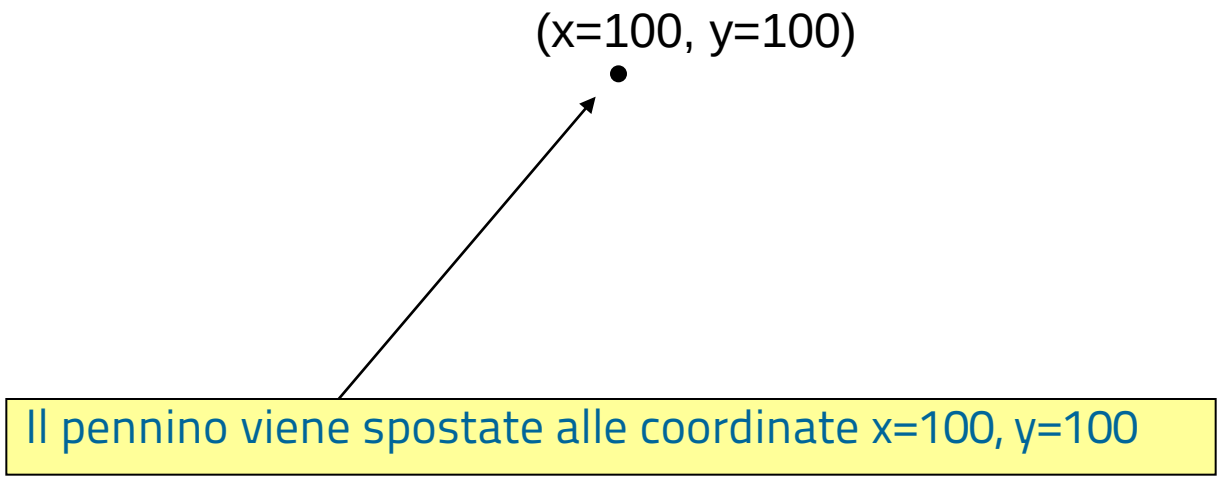
```
package {
	import flash.display.Shape;
	import flash.display.Graphics;
	import flash.display.Sprite;

	public class Disegno extends Sprite {
		public function Disegno() {
			var linea:Shape=new Shape()  ;

	linea.graphics.lineStyle(2,0xFF0000);
				linea.graphics.moveTo(100,100);
				linea.graphics.lineTo(200,200);
				addChild(linea)
		}
	}
}
```

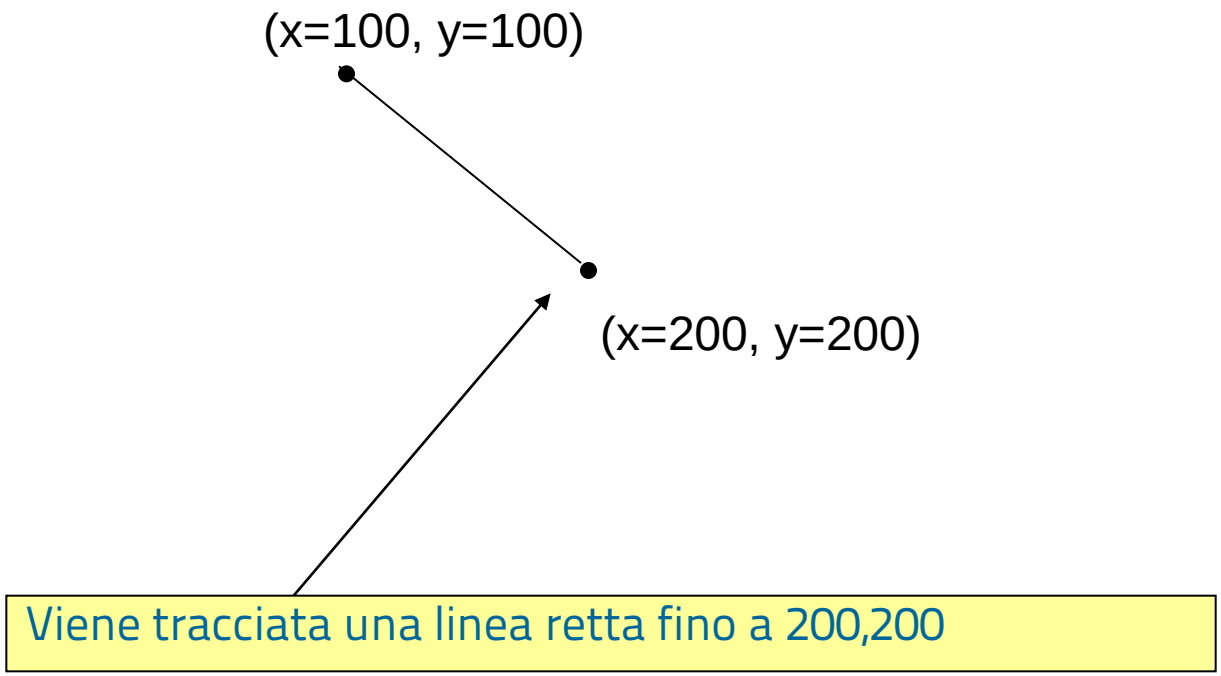
# DISEGNARE UN LINEA

```
linea.graphics.moveTo(100,100);
```



# DISEGNARE UN LINEA

```
linea.graphics.lineTo(200,200);
```



# DISEGNARE UN LINEA

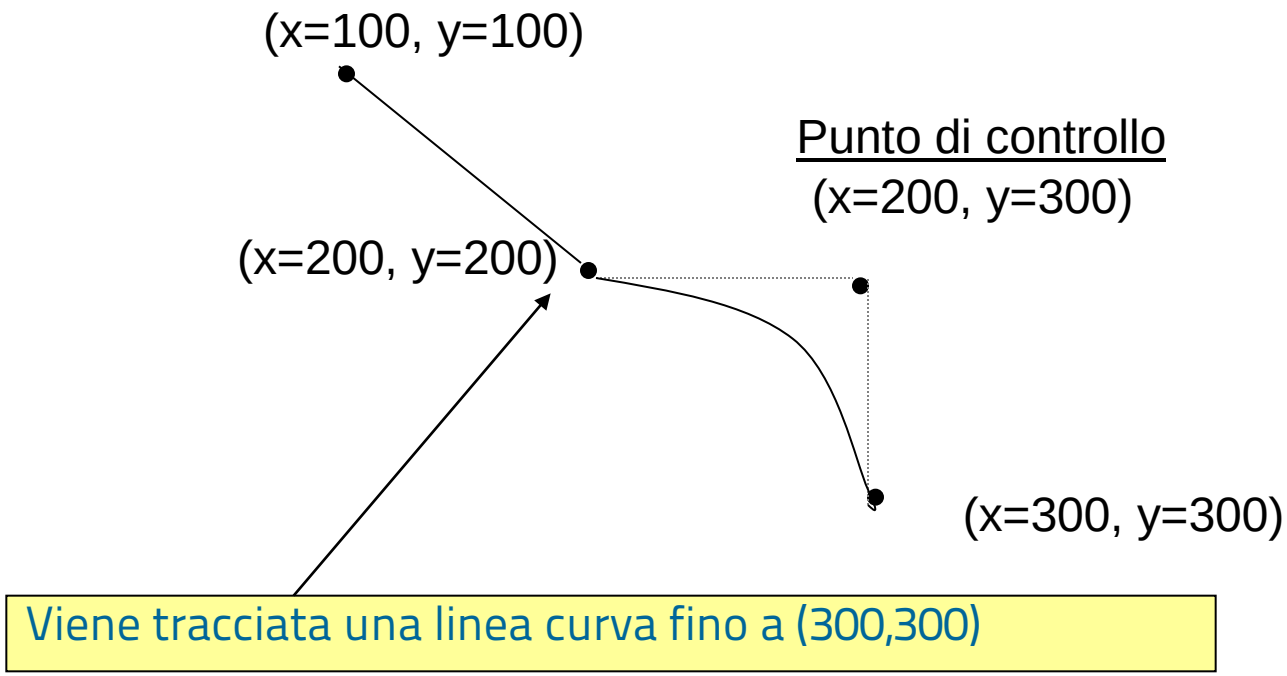
- Questa classe disegna aggiunge un tratto curvo

```
package {
        import flash.display.Shape;
        import flash.display.Graphics;
        import flash.display.Sprite;
        public class Disegno2 extends Sprite {
                public function Disegno2() {
                        var linea:Shape=new Shape()  ;
                        //var curva:Shape = new Shape();

                        linea.graphics.lineStyle(2,0xFF0000);
                        linea.graphics.moveTo(100,100);
                        linea.graphics.lineTo(200,200);
                        linea.graphics.curveTo(200,300,300,300);
                        addChild(linea)
                }
        }
}
```

# DISEGNARE UN LINEA

```
linea.graphics.curveTo(200,300,300,300);
```



# DISEGNARE FIGURE

- Disegna un cerchio e un elissi

```
package {
        import flash.display.Shape;
        import flash.display.Graphics;
        import flash.display.Sprite;
        public class Disegno3 extends Sprite {
           public function Disegno3() {
                var cerchio:Shape=new Shape()  ;
                var ellissi:Shape = new Shape();
                cerchio.graphics.lineStyle (1,0x00FF00);
                cerchio.graphics.drawCircle(100,100,45);
                addChild(cerchio)
                ellissi.graphics.lineStyle (1,0x0000FF);
                ellissi.graphics.drawEllipse(200,150,45, 100);
                addChild(ellissi)
          }
        }
}
```

# DISEGNARE FIGURE

- Disegna un rettangolo

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

        public class Disegno4 extends Sprite {
           public function Disegno4() {
                var rettangolo:Shape = new Shape();

                rettangolo.graphics.lineStyle (1,0x0000FF);
                rettangolo.graphics.drawRect(30. 40,45, 100);
                addChild(rettangolo)
           }
        }
}
```

# DISEGNARE FIGURE

- Disegna un rettangolo

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

        public class Disegno4 extends Sprite {
           public function Disegno4() {
                var rettangolo:Shape = new Shape();

                rettangolo.graphics.lineStyle (1,0x0000FF);
                rettangolo.graphics.drawRect(30. 40,45, 100);
                addChild(rettangolo)
           }
        }
}
```

# DISEGNARE FIGURE

- Disegna un rettangolo con riempimento

```
package {
        import flash.display.Shape;
        import flash.display.Graphics;
        import flash.display.Sprite;
        import flash.display.GradientType;
        public class Disegno5 extends Sprite {
                public function Disegno5() {
                var rettangolo:Shape = new Shape();

                rettangolo.graphics.lineStyle (1,0x0000FF);
                rettangolo.graphics.beginFill(0x0000FF);
                rettangolo.graphics.drawRect(30, 40,45, 100);
                addChild(rettangolo)

                }
        }
}
```