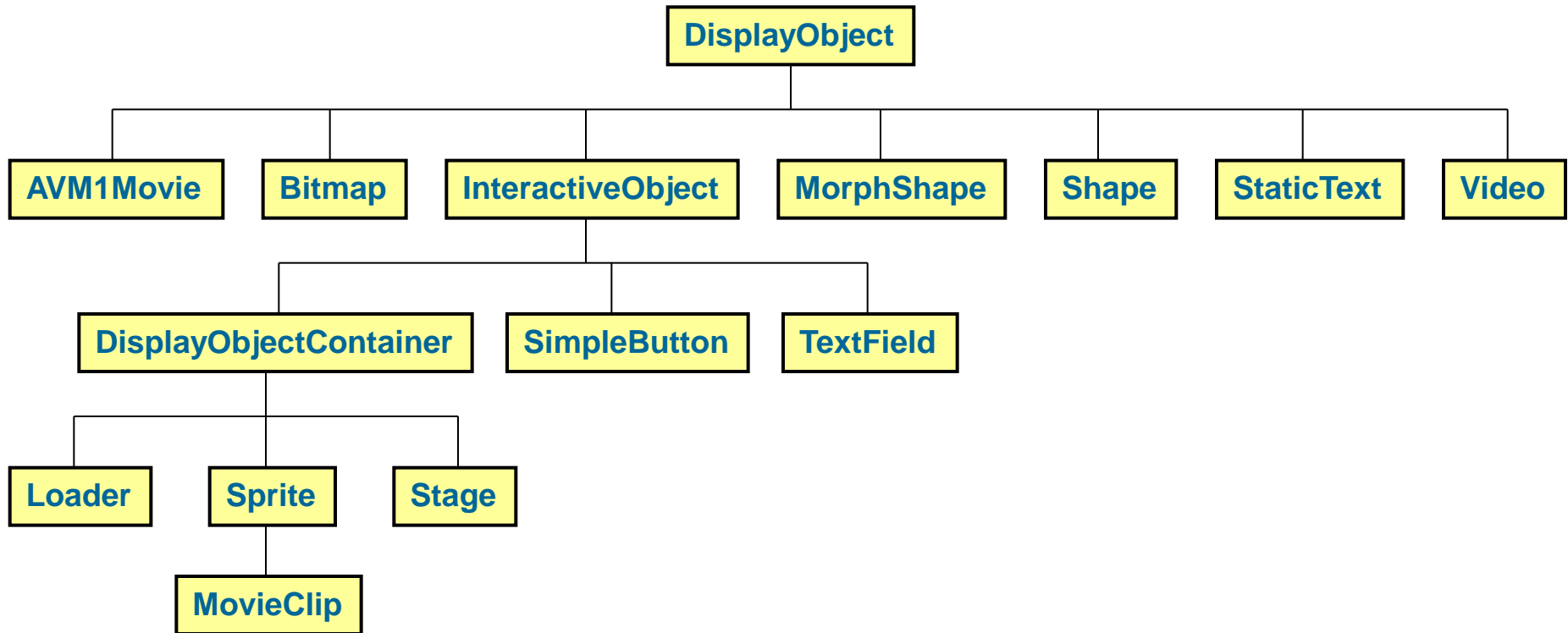


PROGRAMMAZIONE VISUALE

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 consentono 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 contengono contenuti caricati da disco
 - Discendenti di **Sprite** e **MovieClip** che a loro volta possono contenere altri oggetti.

DisplayObjectContainer

Sprite

MovieClip

Le calssi derivate da Sprite:

- Rispondono agli eventi del mouse e della tastiera
- Possono contenere altri oggetti grafici
- Hanno un solo frame

Le calssi derivate da MovieClip:

- Rispondono agli eventi del mouse e della tastiera
- Possono contenere altri oggetti grafici
- Hanno la timeline e quindi più frame

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 visualizzati nell'ordine in cui sono stati aggiunti (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 all'indice specificato.
- **removeChild**(child:DisplayObject) elimina l'oggetto specificato.
- **numChildren**: numero degli oggetti contenuti nella displaylist di un determinato oggetto.
- **setChildIndex**(child:DisplayObject, index:int) modifica la posizione di un oggetto nella displaylist

DOCUMENT CLASS

- La **Document Class** è la classe che associa 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 proprietà 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);  
    .....  
}  
.....
```

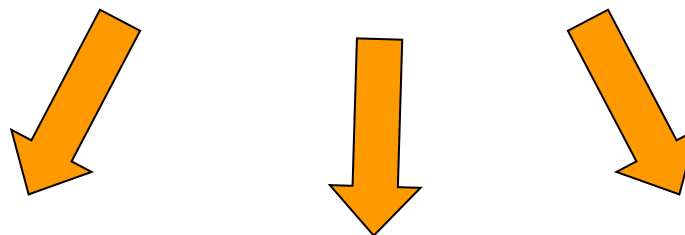
EREDITARIETÀ

EREDITARIETÀ

- Uno dei vantaggi assicurati dalla programmazione orientata agli oggetti è la possibilità di creare *sottoclassi* di una classe.
- La sottoclasse eredita tutte le proprietà e i metodi di una **superclasse**.
 - Posso creare una classe estendendo una classe predefinita.
 - Ma posso anche creare un set di classi che estenda una superclasse sempre creata da me.

EREDITARIETÀ

Vedura



EREDITARIETÀ

- La sottoclasse può aggiungere dei metodi alla superclasse e ridefinire i metodi ereditati creando così nuovi comportamenti per gli stessi metodi.
- L'uso di sottoclassi consente di riutilizzare il codice, in quanto estendendo una classe esistente si evita di riscrivere tutto il codice comune a entrambe le classi.

LE SOTTOCLASSI IN FLASH

- Nella programmazione orientata agli oggetti, una sottoclasse può ereditare le proprietà e i metodi di un'altra classe, detta **superclasse**. È possibile estendere le classi personalizzate e molte delle classi di ActionScript. Non possono essere estese la classe **TextField** e le classi statiche, quali **Math**, **Key** e **Mouse**.
- Per creare questo tipo di relazione tra due classi, è necessario utilizzare la clausola **extends** dell'istruzione *class*:

```
class SubClass extends SuperClass {  
    //corpo della classe  
}
```

OROLOGIO GENERICO

OROLOGIO GENERICO

- Creo una classe astratta da cui derivare orologi specifici
- Compito della classe sarà semplicemente tenere aggiornate (una volta al secondo) tre proprietà che conterranno rispettivamente ora, minuti e secondi.
- Useremo una nuova classe la classe Timer

LA CLASSE TIMER

- La classe timer è una classe fornita con ActionScript 3 che genera un evento a intervalli di tempo prestabilito
- E' utile quando ho bisogno di generare eventi ripetuti nel tempo indipendenti dal ritmo scandito dai frame
- In questo caso basta aggiornare l'orologio una volta la secondo.

OROLOGIO GENERICO

- Importo solo le classi che mi consentono di definire lo sprite e il timer.

```
package {  
    import flash.display.Sprite;  
    import flash.utils.Timer;  
    import flash.events.TimerEvent;  
  
    .....  
}
```

OROLOGIO GENERICO

- Definisco le proprietà che il mio timer dovrà aggiornare

```
package {  
    public class OrologioGenerico extends Sprite {  
        protected var ore:uint;  
        protected var minuti:uint;  
        protected var secondi:uint;  
  
        .....  
    }
```


OROLOGIO GENERICO

- Come **constructor** definisco una funzione che chiama i metodi necessari a disegnare l'orologio e a inserirvi una valore iniziale:

```
package {  
    public class OrologioGenerico extends Sprite {  
        public function OrologioGenerico () {  
            leggiOra();  
            inizializzaVisualizzazione();  
            visualizzaOra();  
            inizializzaTimer();  
        }  
        .....  
    }
```

OROLOGIO GENERICO

- Definisco i metodi che inizializzano il timer e lo fanno partire: **leggiOra** aggiorna le variabili sulla base dell'ora fornita dal computer:

```
package {  
    public class OrologioGenerico extends Sprite {  
        .....  
        protected function leggiOra() {  
            var adesso:Date = new Date();  
            ore = adesso.getHours();  
            minuti = adesso.getMinutes();  
            secondi = adesso.getSeconds();  
        }  
        .....  
    }
```

OROLOGIO GENERICO

- **inizializzaTimer** e **aggiorna** sono rispettivamente il metodo che crea e fa partire il timer e il metodo che viene chiamato ad ogni evento generato dal timer:

```
package {  
  public class OrologioGenerico extends Sprite {  
    .....  
    protected function inizializzaTimer() {  
      var myTimer:Timer = new Timer(1000);  
      myTimer.addEventListener(TimerEvent.TIMER, aggiorna);  
      myTimer.start();  
    }  
  
    private function aggiorna(e:TimerEvent) {  
      leggiOra();  
      visualizzaOra();  
    }  
  }  
}
```

OROLOGIO GENERICO

- Dichiaro i metodi **inizializzaVisualizzazione** e **visualizzaOra** che lascio vuoti in quanto saranno implementati nelle sub classi derivata da OrologioGenerico.

```
package {  
  public class OrologioGenerico extends Sprite {  
    .....  
    public function inizializzaVisualizzazione() {  
      //metodo da definire nelle classi derivate  
    }  
  
    public function visualizzaOra () {  
      //metodo da definire nelle classi derivate  
    }  
  }  
}
```