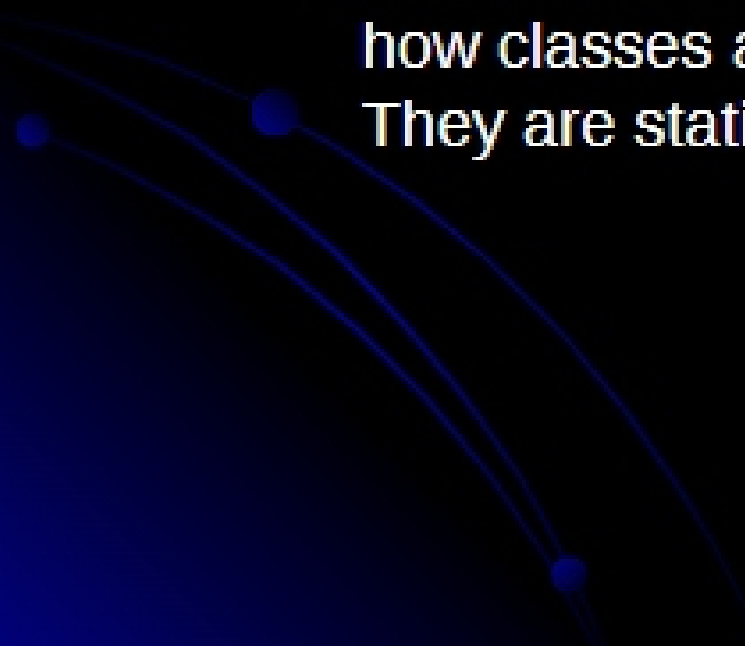


DECORATOR

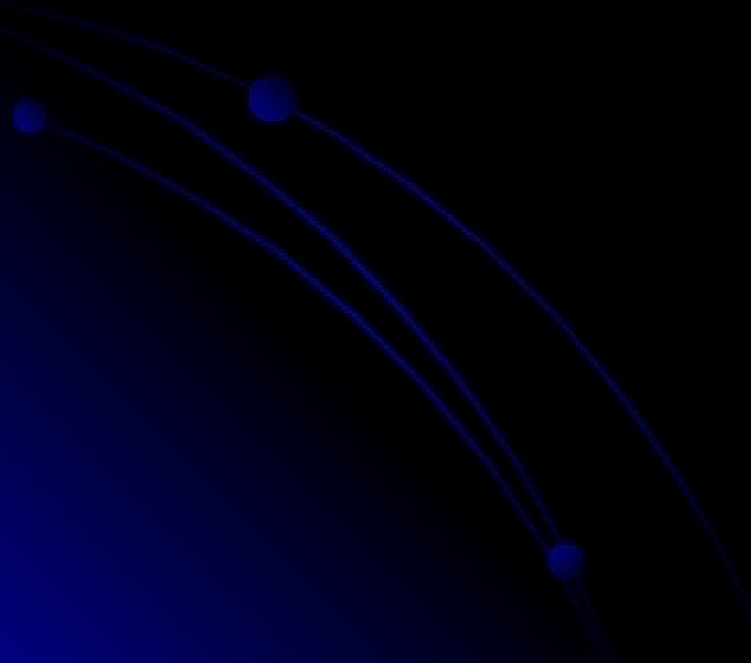
by Ramani Natarajan

- Also known as 'Wrapper.'
 - According to 'gang of four'(sounds like an Akira Kurosawa movie): A Decorator is used "to attach additional responsibilities to an object dynamically. A Decorator provides a flexible alternative to subclassing for extending functionality."
 - It is a 'Structural Object Pattern.'
 - Structural Class patterns like 'Adapter' are concerned with how classes and objects are composed using inheritance. They are static
- 

Decorator (contd.)

compared to structural object patterns like Decorator that describe ways to compose objects to realize new functionality.

- Such a structural object pattern gives the ability to change compositions at run-time.
- It allows open-ended number of additional responsibilities, like adding borders, shadow, instill scrolling, zooming.....



What am I talking about?

Consider the class interface for the universal set
{ x | x is a frog }:

```
DEFINE CLASS Frog AS CUSTOM  
FUNCTION Jump(nValue)  
FUNCTION Eat()  
ENDDEFINE
```

Suppose we want to make Kermit dance; we subclass as:

```
DEFINE CLASS DancingFrog AS Frog  
FUNCTION Dance()  
ENDDEFINE
```