

# cards

Design Patterns Team

November 11, 2015

## Contents

<b>1</b>	<b>Card shape</b>	<b>1</b>
<b>2</b>	<b>Decal</b>	<b>1</b>
2.1	Decal types . . . . .	2
2.2	Nesting . . . . .	2
2.3	Layout class . . . . .	2
2.3.1	Background . . . . .	2
2.3.2	General patterns . . . . .	2
<b>3</b>	<b>Theming</b>	<b>2</b>
<b>4</b>	<b>Cards</b>	<b>3</b>
<b>5</b>	<b>Example UML</b>	<b>3</b>

## 1 Card shape

- Rectangle, circle, etc
- Corners, rounded, sharp?

## 2 Decal

- Generic art, gets an art object
- Decals will need to be patterned, one decal might want to be on each corner, or might want to go in a row, would be very awkward to have to manually do all this patterning : Decal shouldn't do patterning, let the transformation class do it.

## **2.1 Decal types**

- Image decals
- Number decals
- Text decals
- Shape decal

## **2.2 Nesting**

Decals will want to be nested, a text box might want to be on top of a background.

## **2.3 Layout class**

Decorates an image with scale, position, etc, can be composited, and is clonable?

- Could then handle patterning
- Composite

### **2.3.1 Background**

- Decal, covering the whole card

### **2.3.2 General patterns**

- Textbox
- Border
- Etc?

## **3 Theming**

- Sub decks
- Some sets will have the same decal applied to the same spot
- Others will have the same decal, but used in a different spot per card

- EX) Cards have some number of \$family decals, but those decals are in different spots, they don't know what image they are until we tell it its family. Thus, would say something like \$family = spades, and all the \$family decals will use the spades image.

```
clone = prototype.clone();
clone.setDecal("family",spades);
```

Families will still need per card information. So perhaps...

```
clone.setCharacter(charizard.jpg)
clone.setHP(120);
```

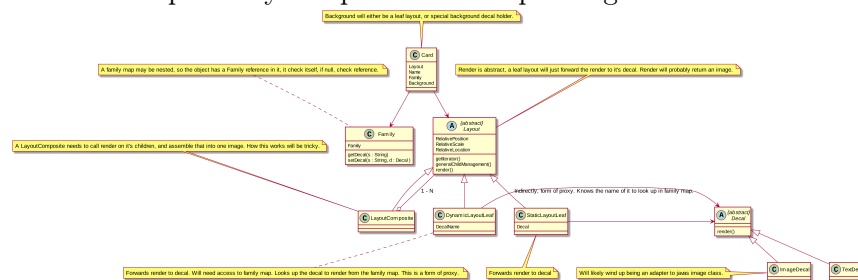
## 4 Cards

Card has a layout. Layouts can be cloned. Cards can be cloned.

- Card will have a name.
- Card will have a layout
- Card have a family? Family is just a map, string -> decal, Two types of layout leafs. FamilyLeaf, which just has a string, asks its family. DecalLeaf, which holds a decal.
  - A family can be nested, will query its map, then parent map.

## 5 Example UML

This should probably be split into multiple diagrams.



- Some patterns used here
  - Proxy : the dynamicleaf layout

- Composite : The layouts
- Adapter : Image Decal
- Iterator : For the composite