

Super and Inner — Together at Last!

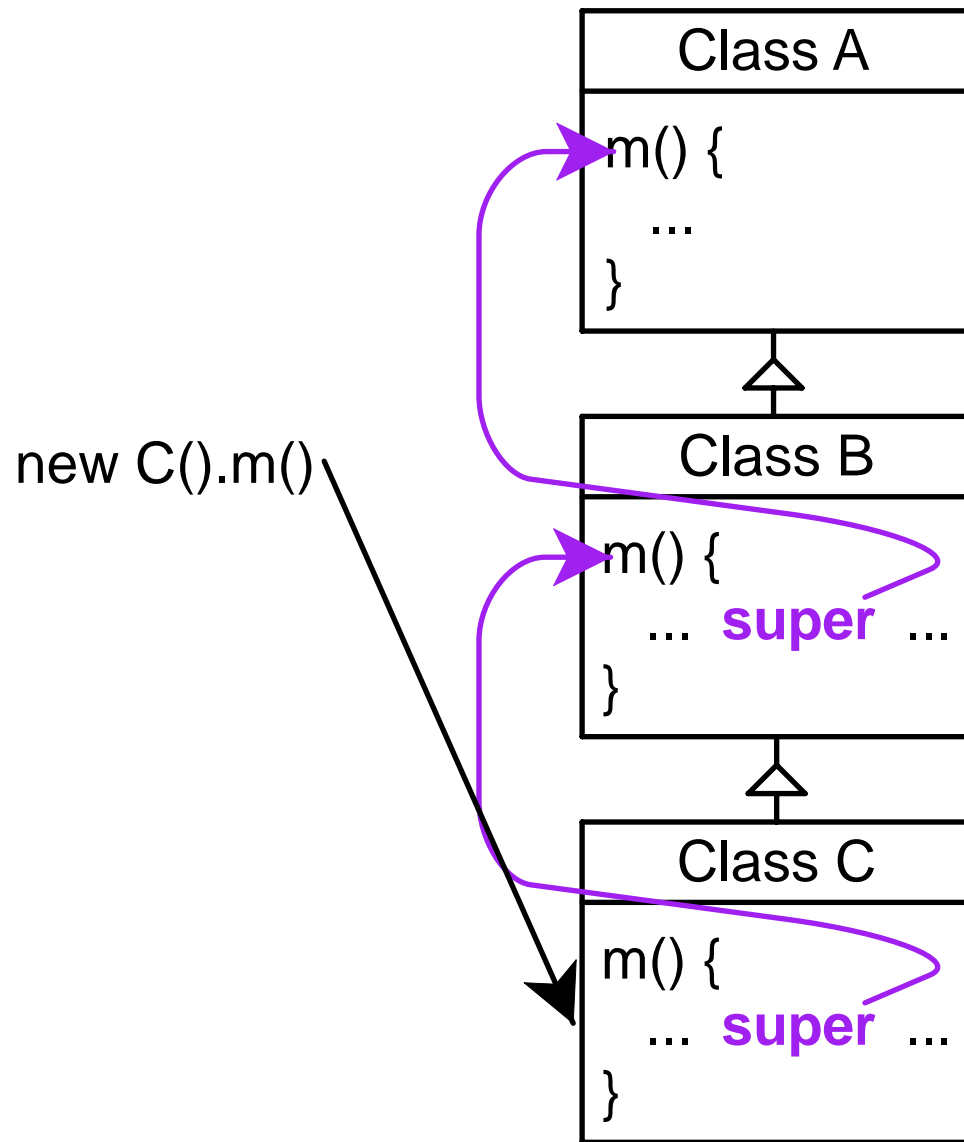


David S. Goldberg
University of Utah

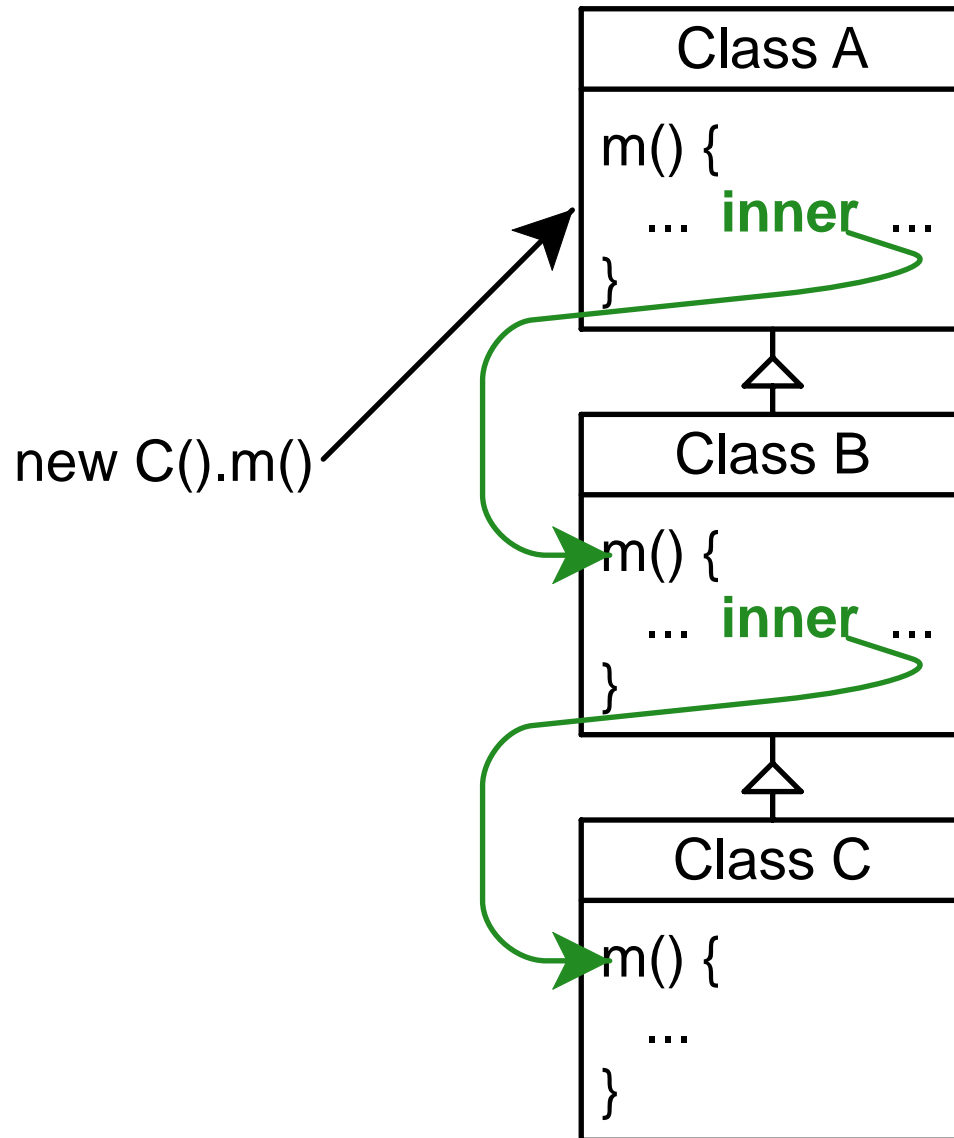
Robert Bruce Findler
University of Chicago

Matthew Flatt
University of Utah

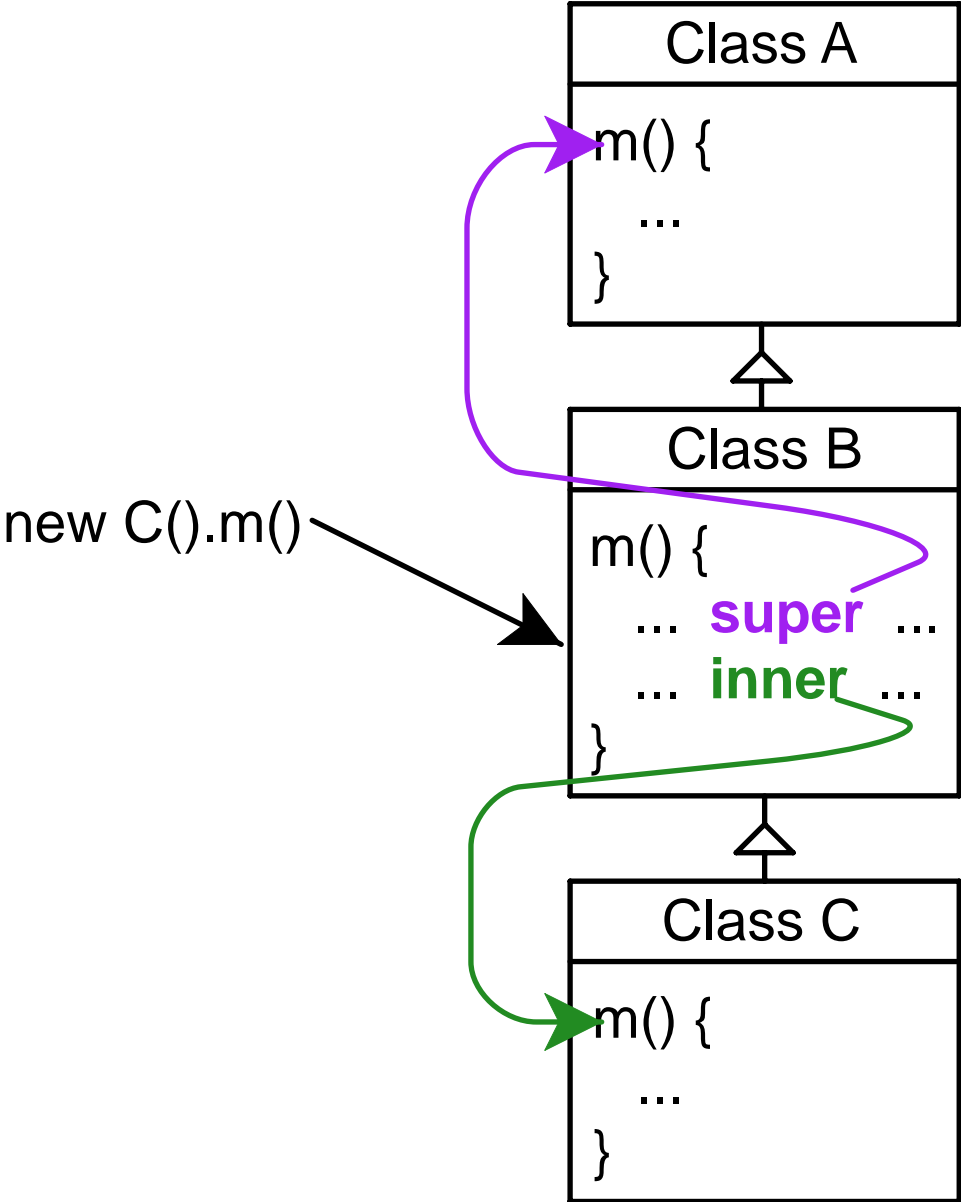
Java-Style Method Overriding



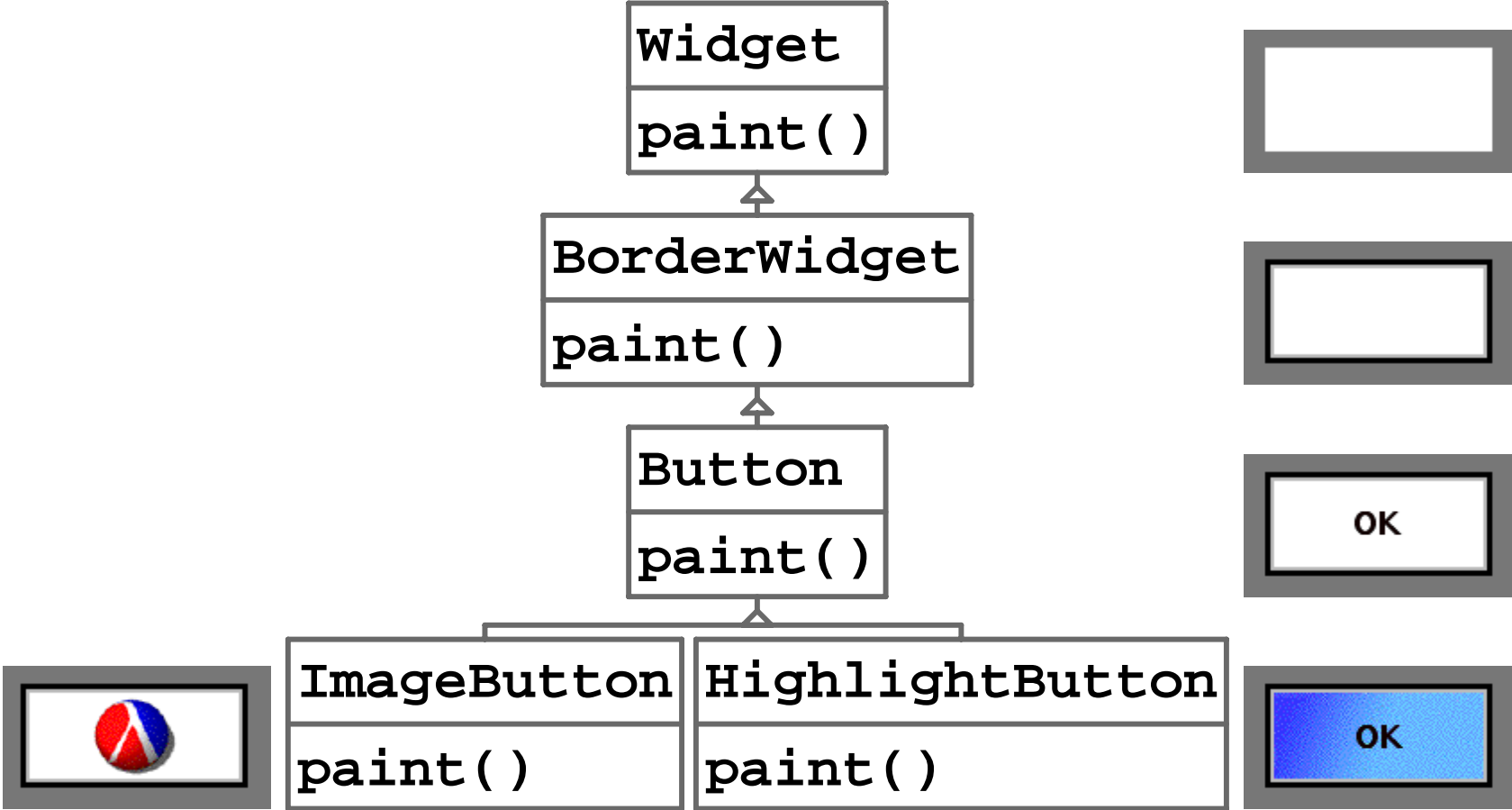
Beta-Style Method Augmentation



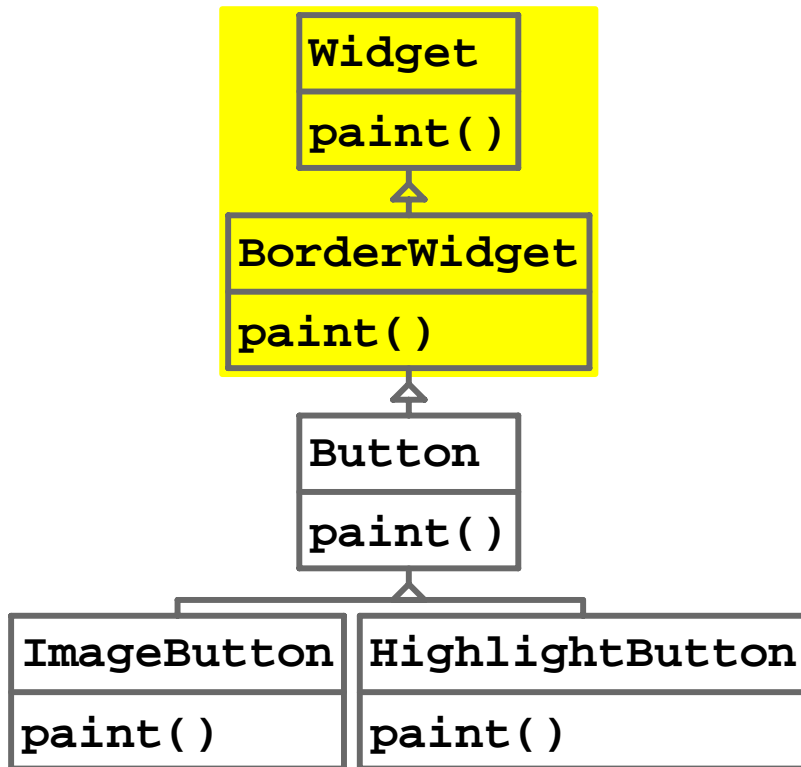
A Combination



Creating a GUI Widget Hierarchy



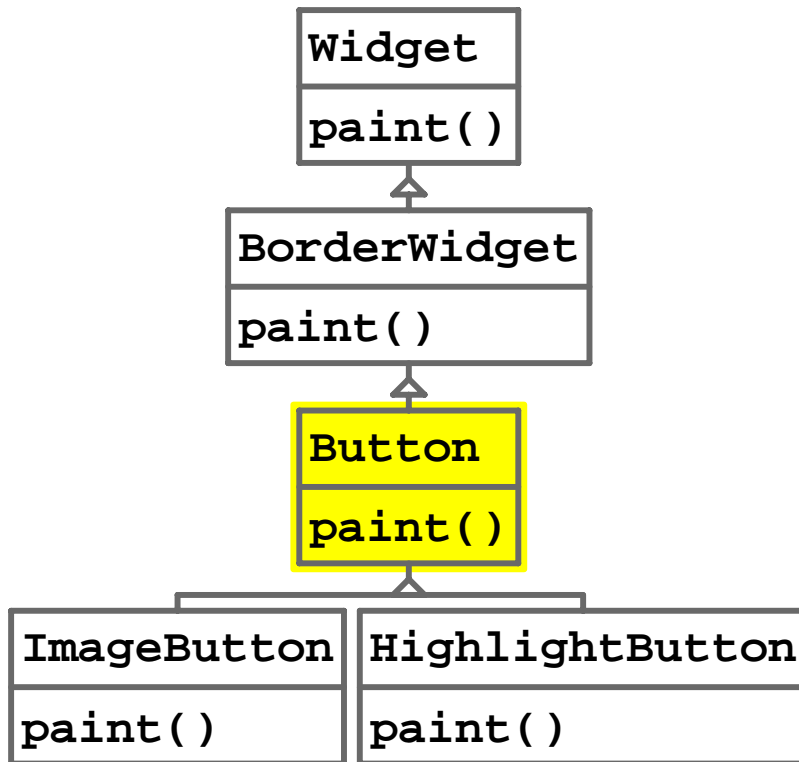
Widget and BorderLayout in Java



```
class Widget ...
    void paint() {
        // paint the background:
        ...;
    }
```

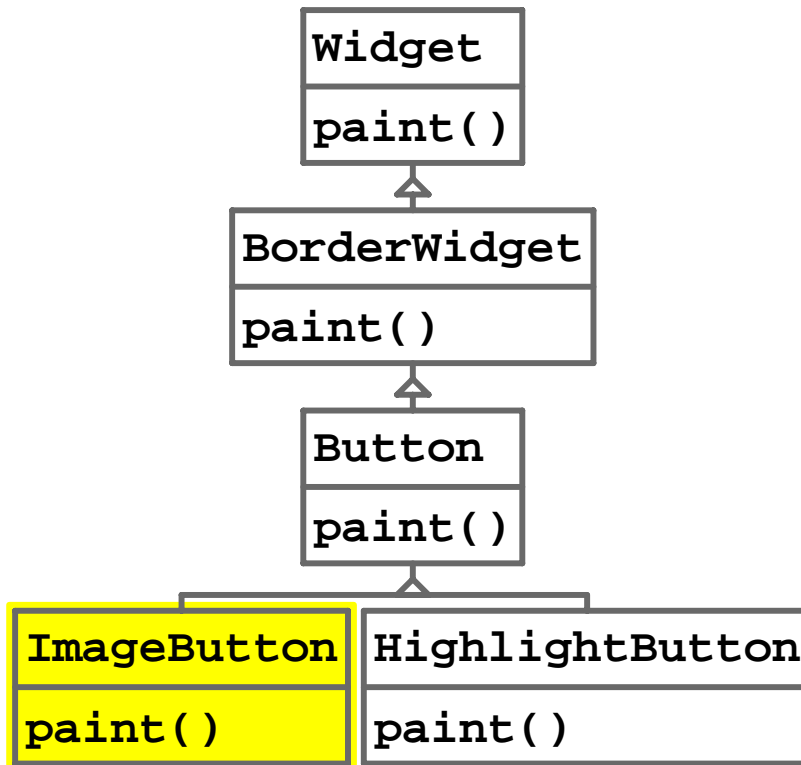
```
class BorderLayout ...
    void paint() {
        // paint background:
        super.paint();
        // draw a border:
        ...;
    }
```

Button in Java



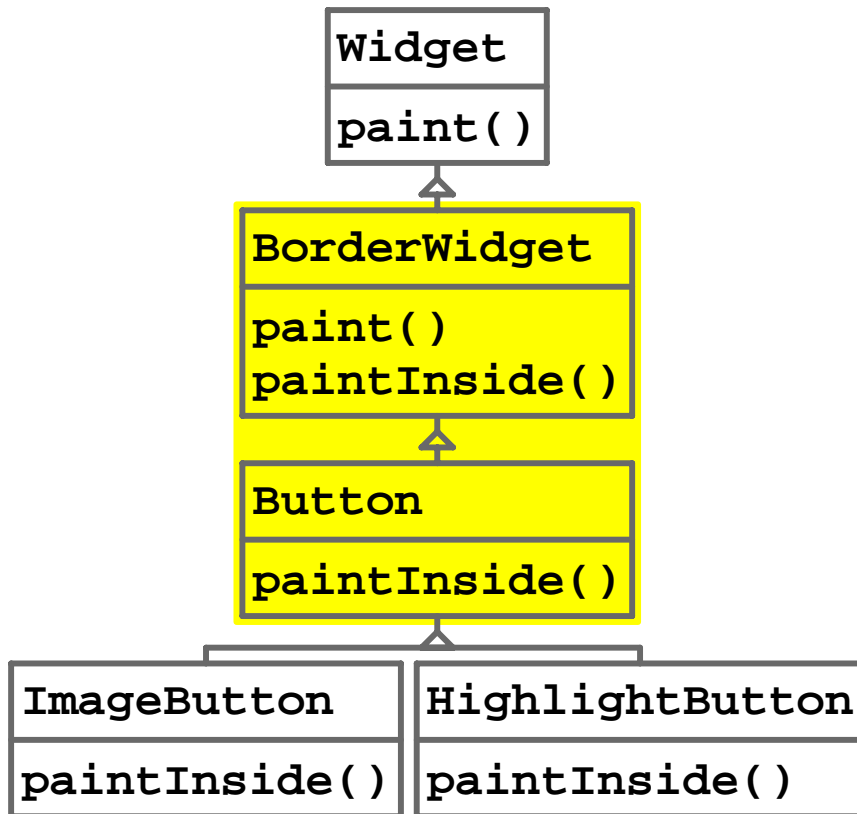
```
class Button ...
    void paint() {
        // paint background, border:
        super.paint();
        // draw a button label:
        ...;
    }
```

ImageButton in Java



```
class ImageButton ...
    void paint() {
        // paints background, border
        // - and label!
        super.paint();
        // draw image
        // must cover label:
        ...;
    }
```

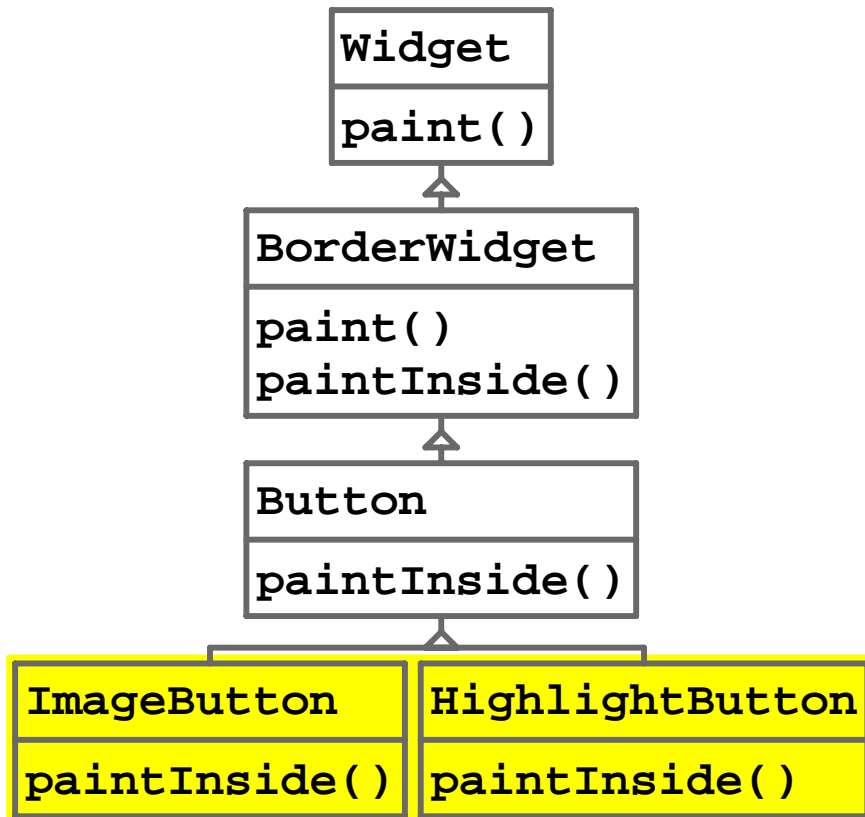

Fixed BorderLayout and Button in Java



```
class BorderLayout ...
    final void paint() {
        // paint background:
        super.paint();
        // paint a border:
        ...;
        paintInside();
    }
    void paintInside() {}
```

```
class Button ...
    void paintInside() {
        // draw label:
        ...;
    }
```

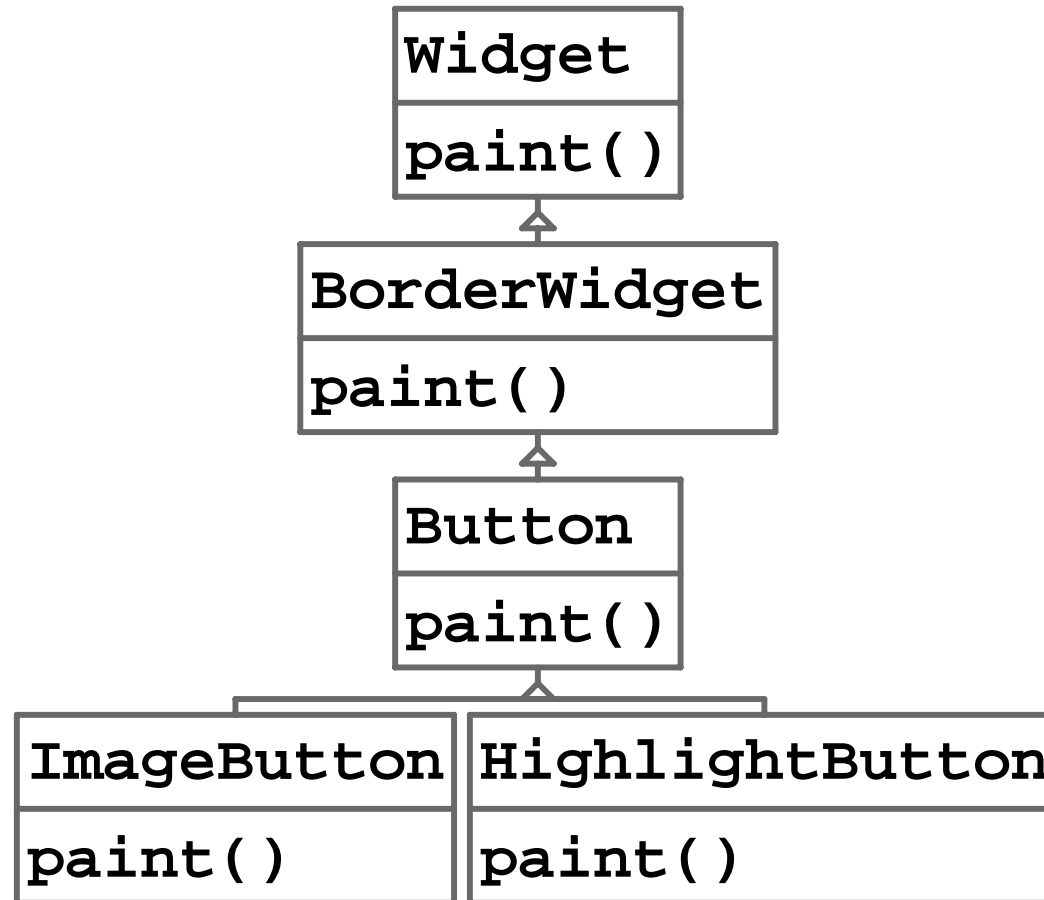
ImageButton and HighlightButton in Java



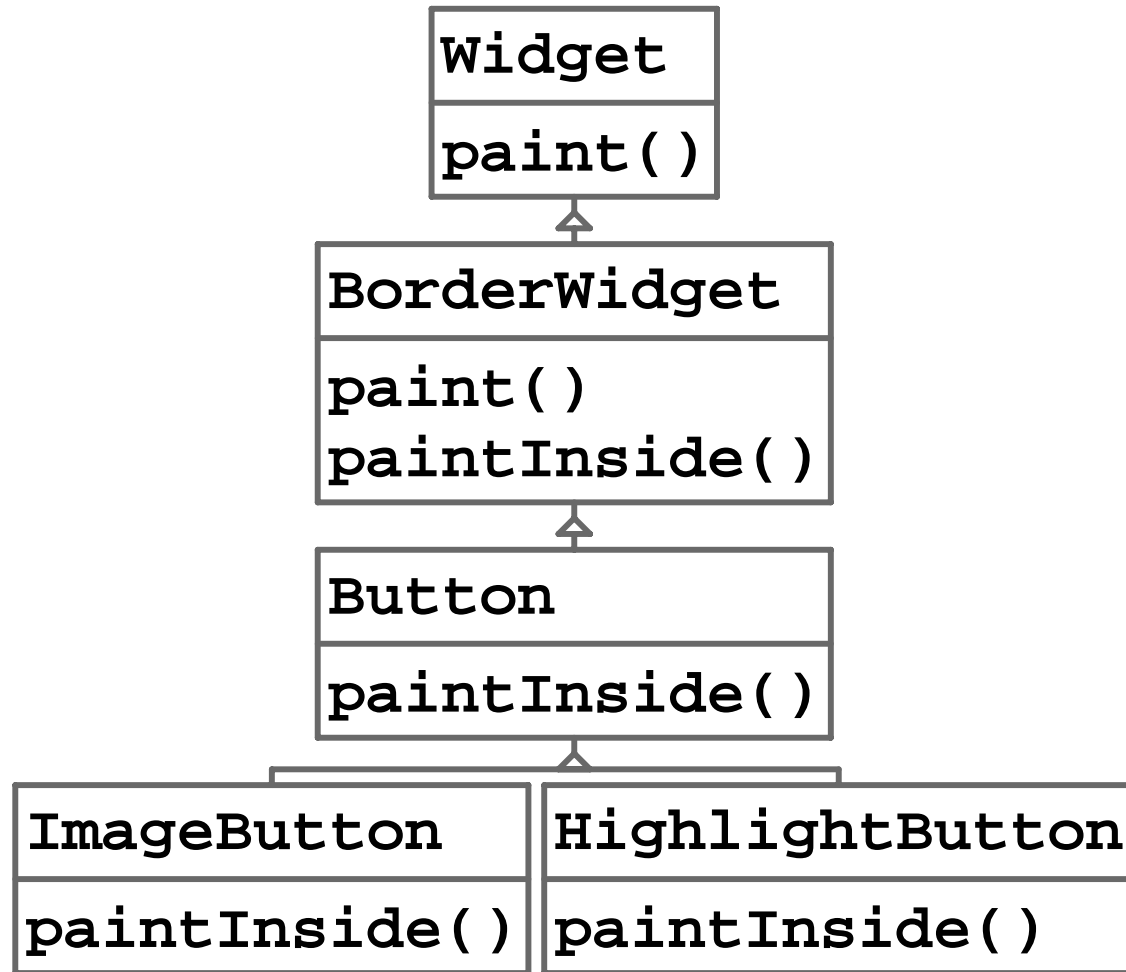
```
class ImageButton ...
    void paintInside() {
        // draws image
        // doesn't call super
        // for label:
        ...;
    }
```

```
class HighlightButton ...
    void paintInside() {
        // replace background with
        // blue gradient:
        ...;
        // draws the label:
        super.paintInside();
    }
```

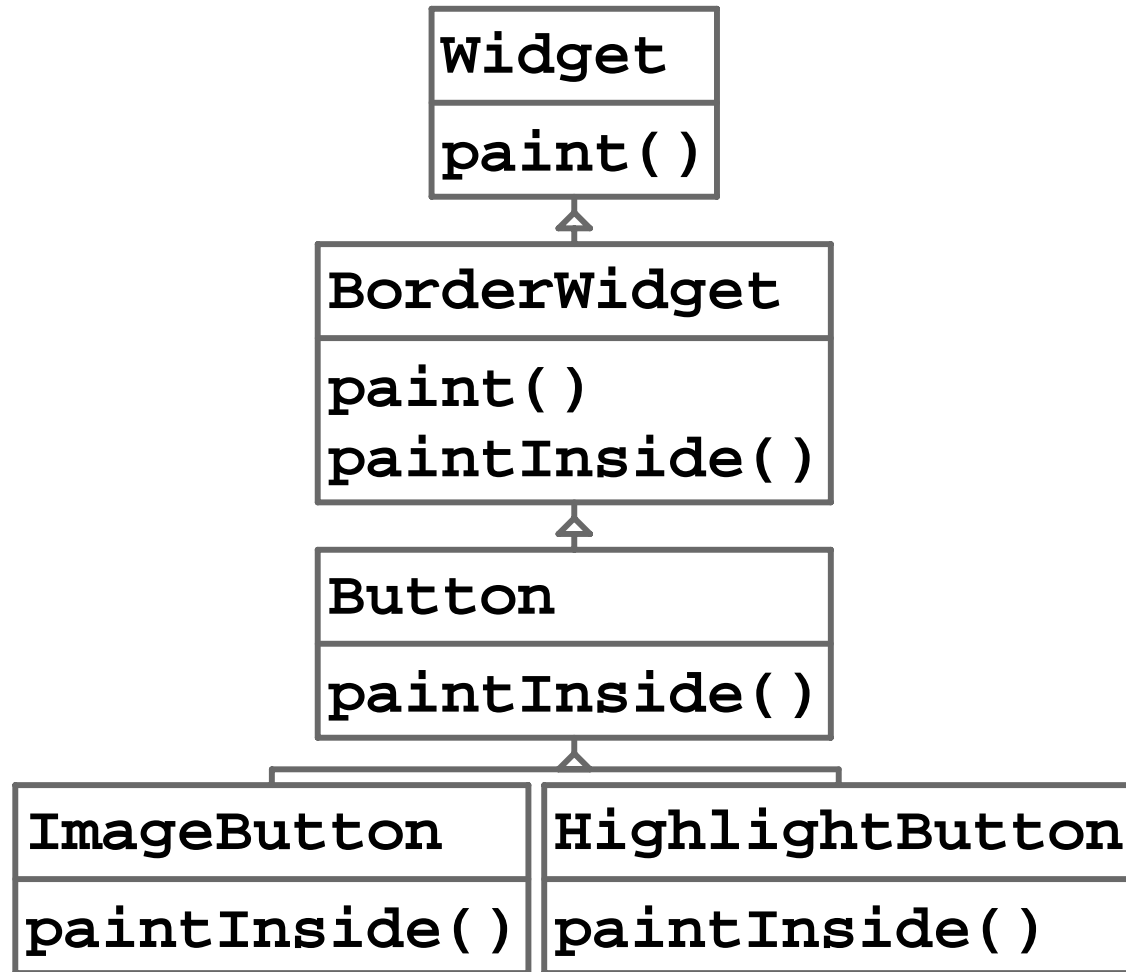
The Problem with Inventing Names



The Problem with Inventing Names

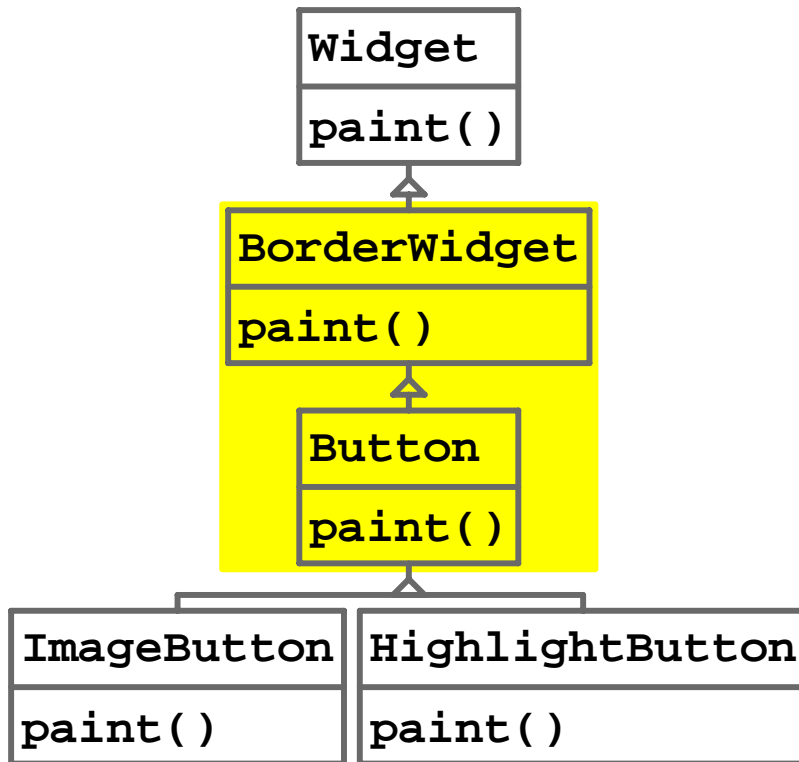


The Problem with Inventing Names



- Other workarounds: Decorators/Delegation, Aspects

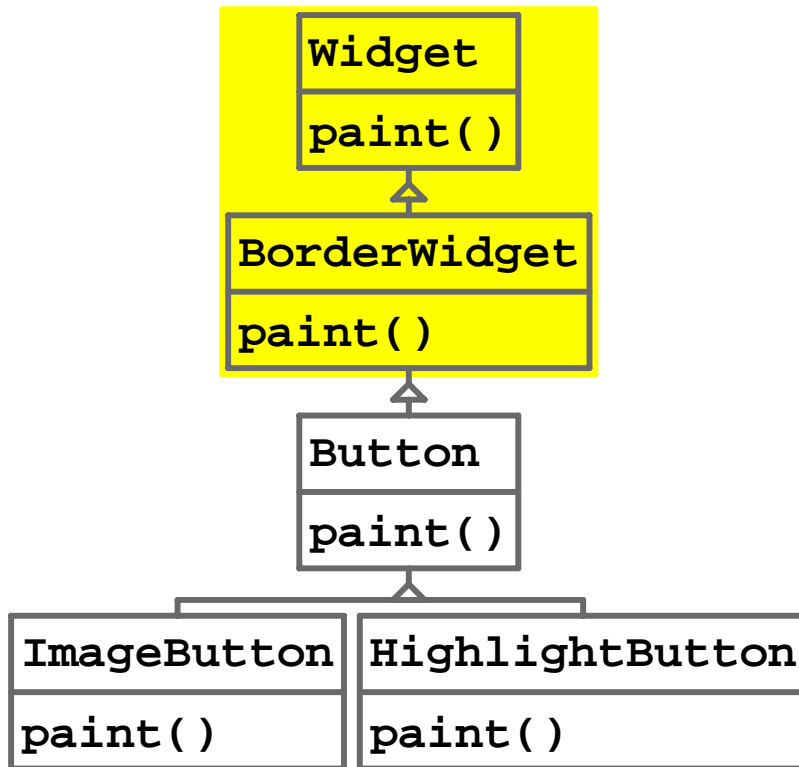
BorderWidget and Button in Beta



```
class BorderWidget ...
    void paint() {
        // draw a border:
        ...;
        inner.paint();
    }
```

```
class Button ...
    void paint() {
        // draw a label:
        ...;
    }
```

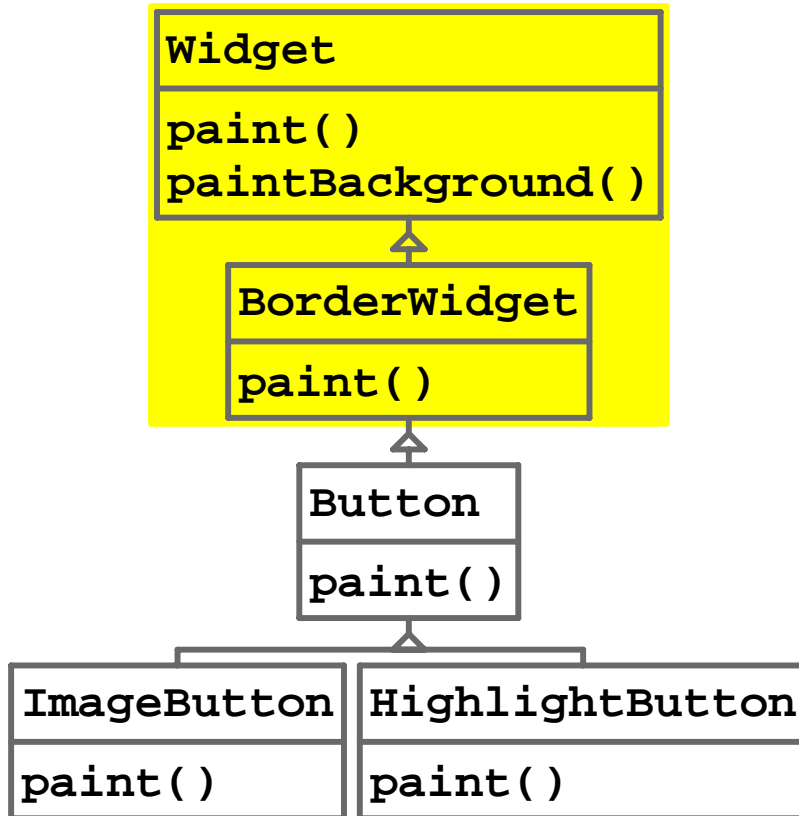
Widget and BorderLayout in Beta



```
class Widget ...
    void paint() {
        // paint the background:
        ...;
        inner.paint();
    }
```

```
class BorderLayout ...
    void paint() {
        // draw a border:
        ...;
        inner.paint();
    }
```

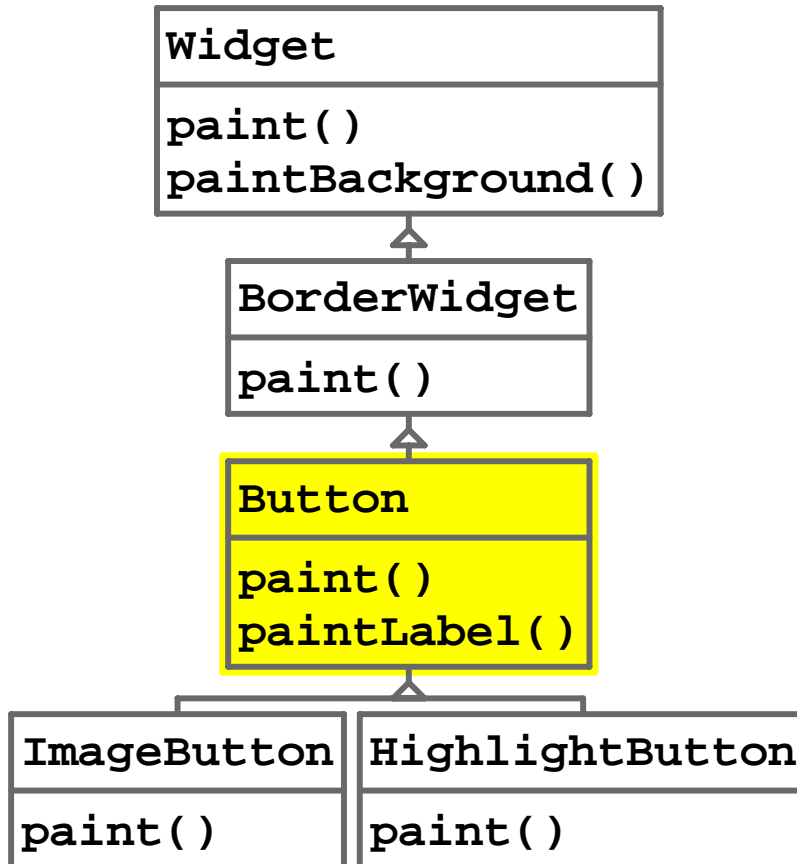
Fixed Widget and BorderLayout in Beta



```
class Widget ...
    void paintBackground() {
        // paint the background:
        ...;
    }
    void paint() {
        inner.paint()
        else paintBackground();
    }
}
```

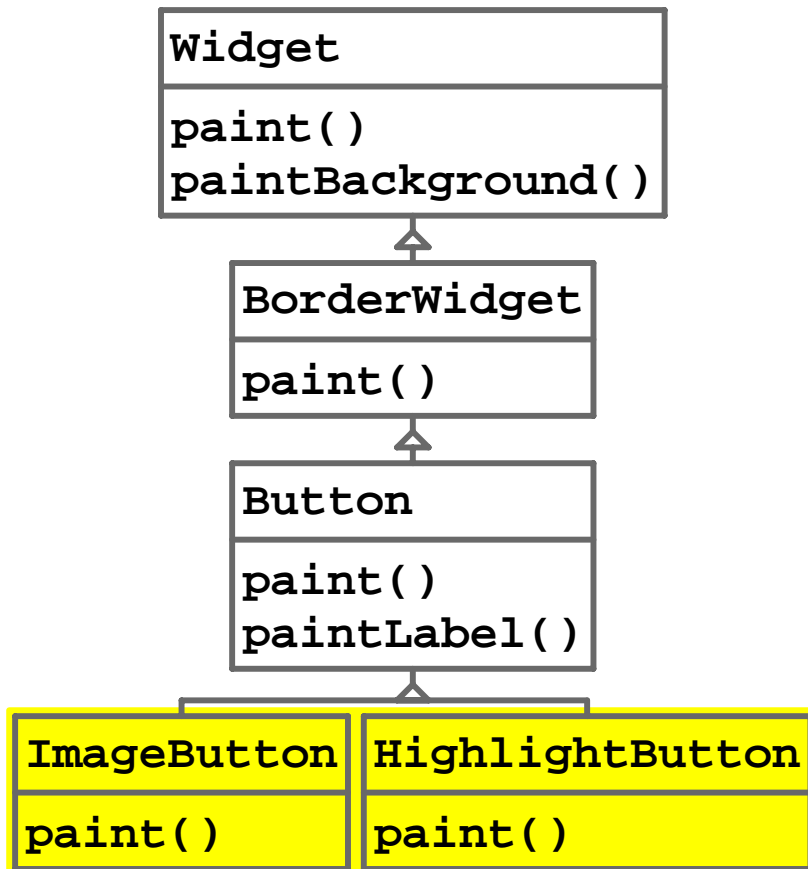
```
class BorderLayout ...
    void paint() {
        paintBackground();
        // draw a border:
        ...;
        inner.paint();
    }
}
```


Button in Beta



```
class Button ...
    void paintLabel() {
        // draw a button label:
        ...;
    }
    void paint() {
        inner.paint()
        else paintLabel();
    }
```

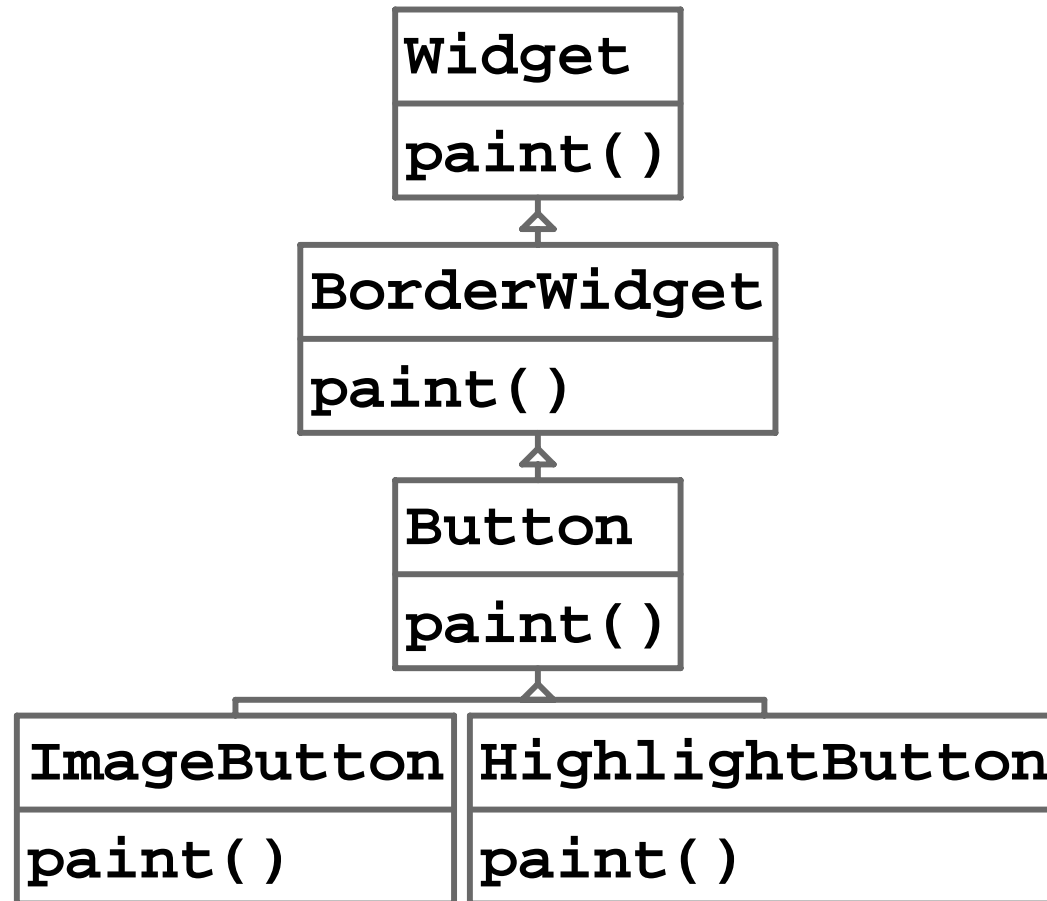
ImageButton and HighlightButton in Beta



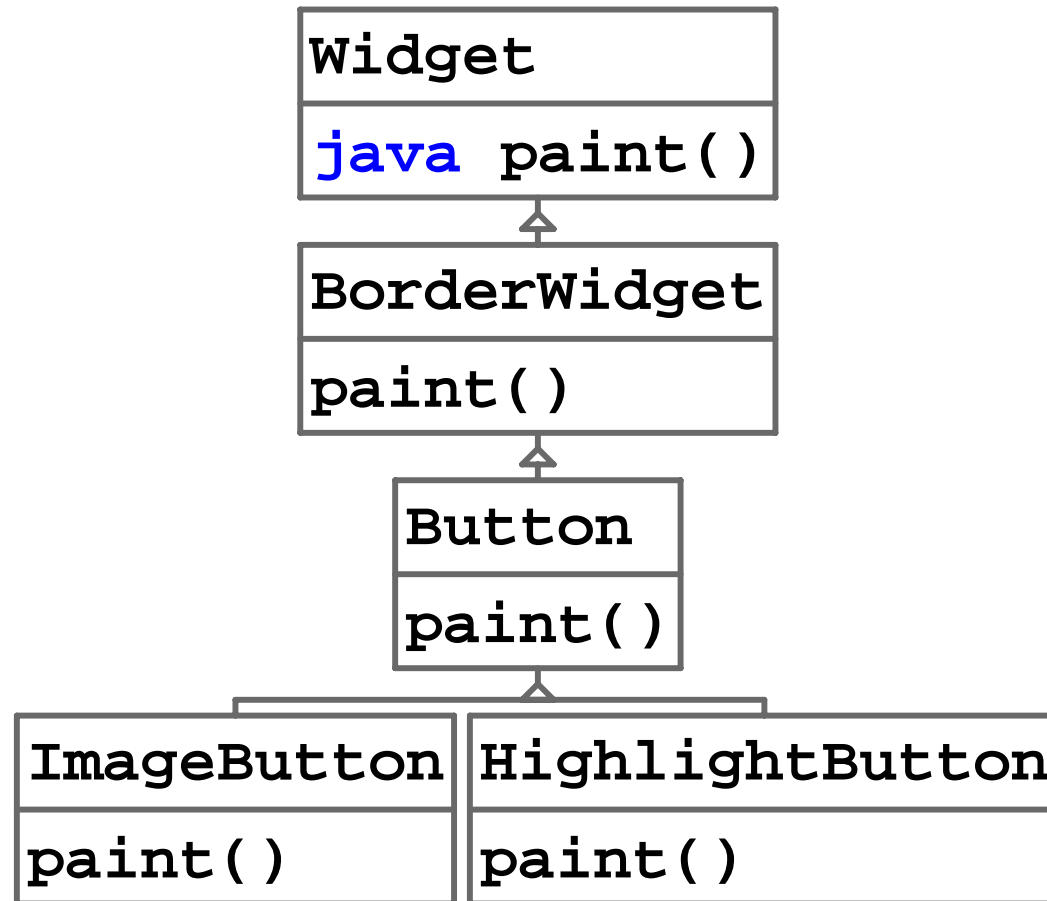
```
class ImageButton ...
    void paint() {
        // draw image
        // (don't call paintLabel):
        ...;
    }
```

```
class HighlightButton ...
    void paint() {
        // replace background with
        // blue gradient:
        ...;
        paintLabel(); // draws label
    }
}
```

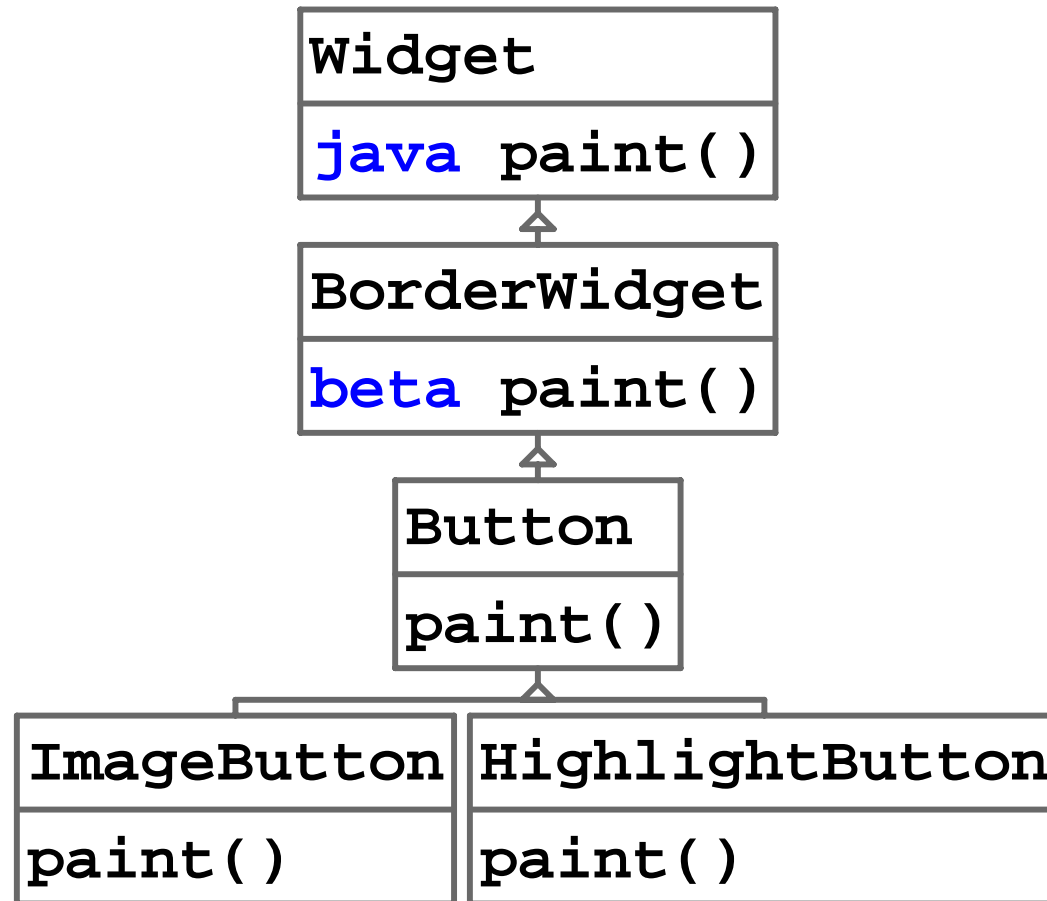
Combining Beta and Java Styles



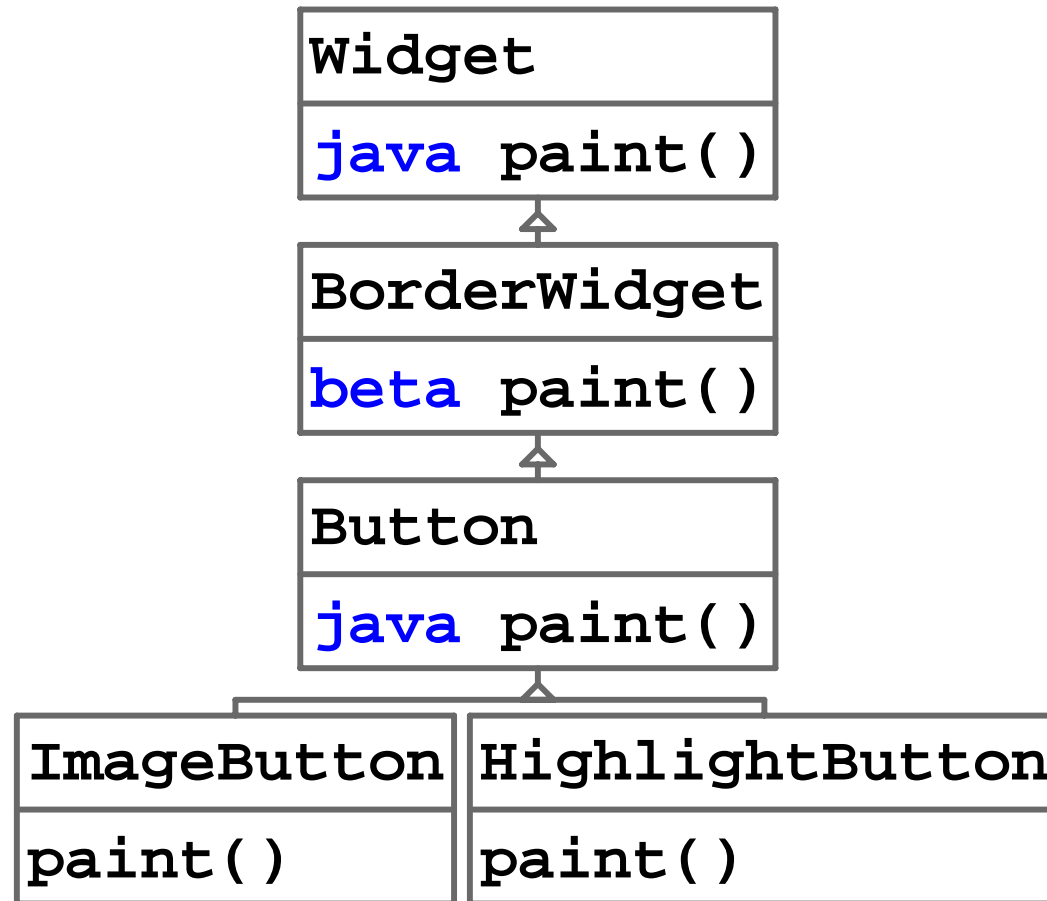
Combining Beta and Java Styles



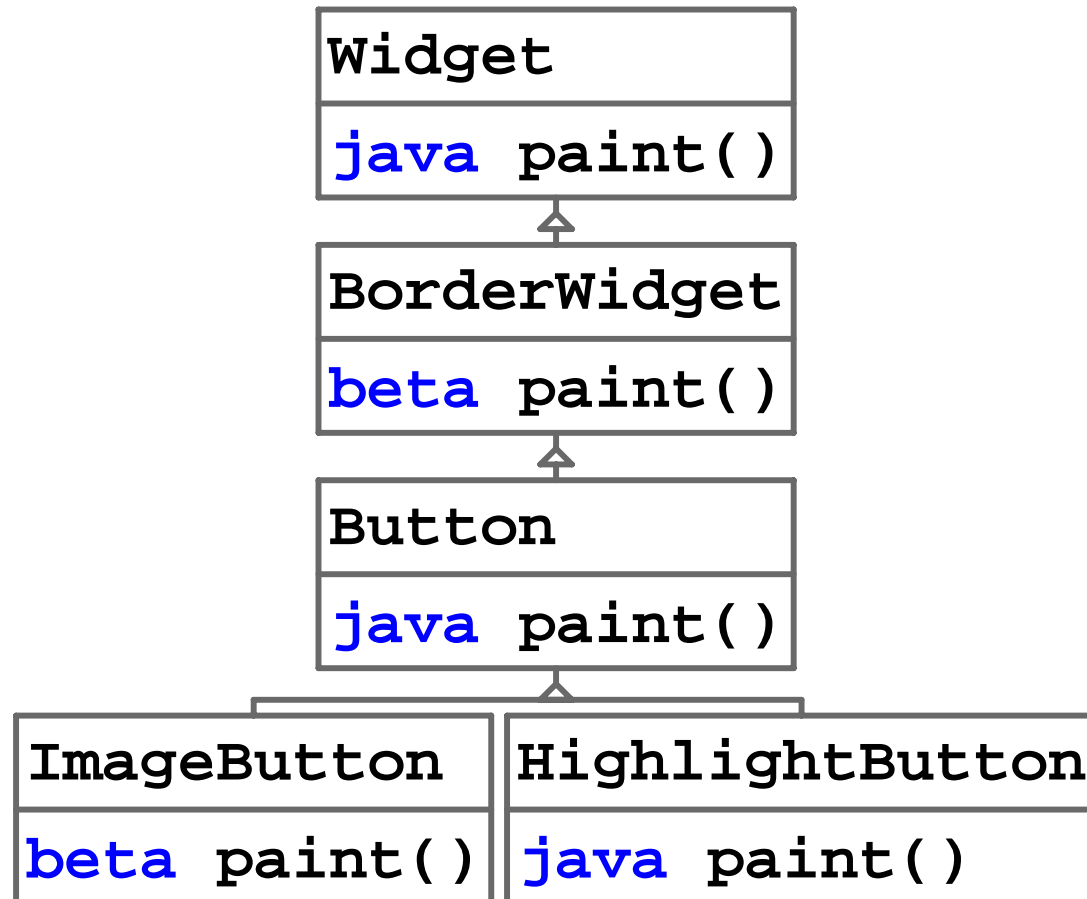
Combining Beta and Java Styles



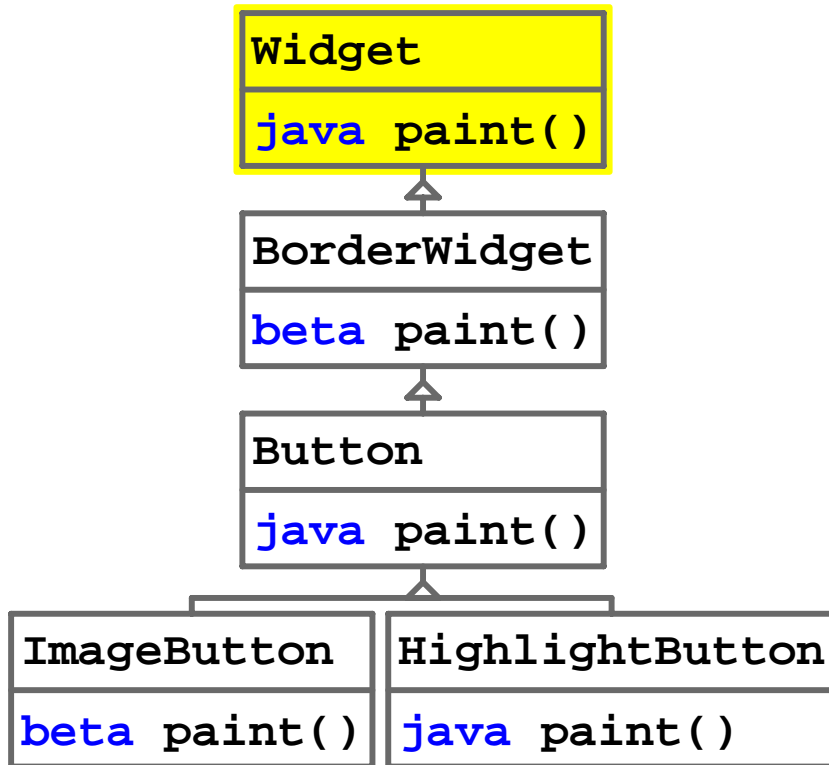
Combining Beta and Java Styles



Combining Beta and Java Styles

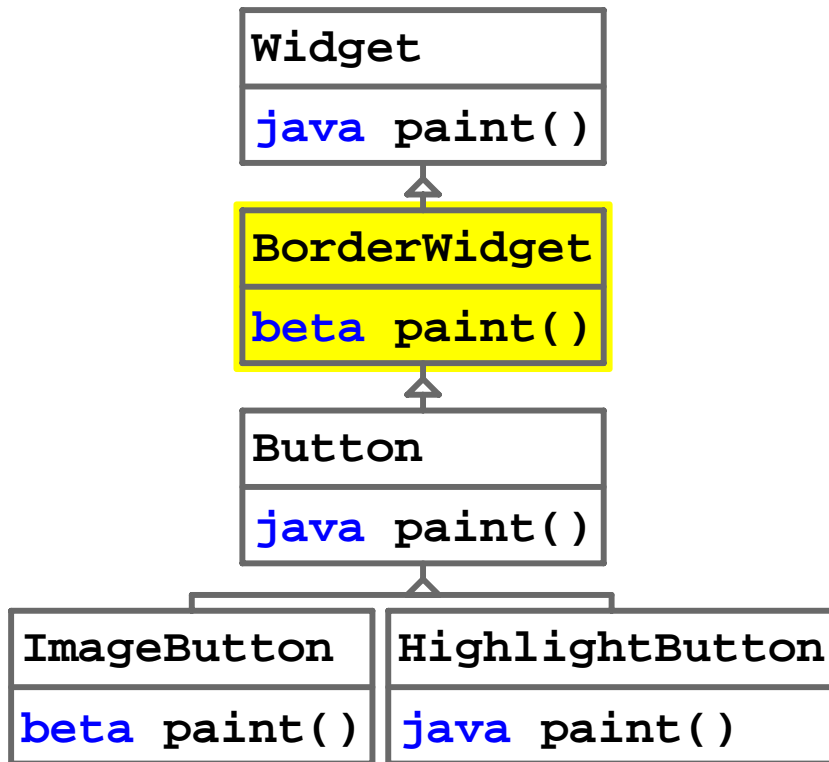


Code for Beta/Java Combination



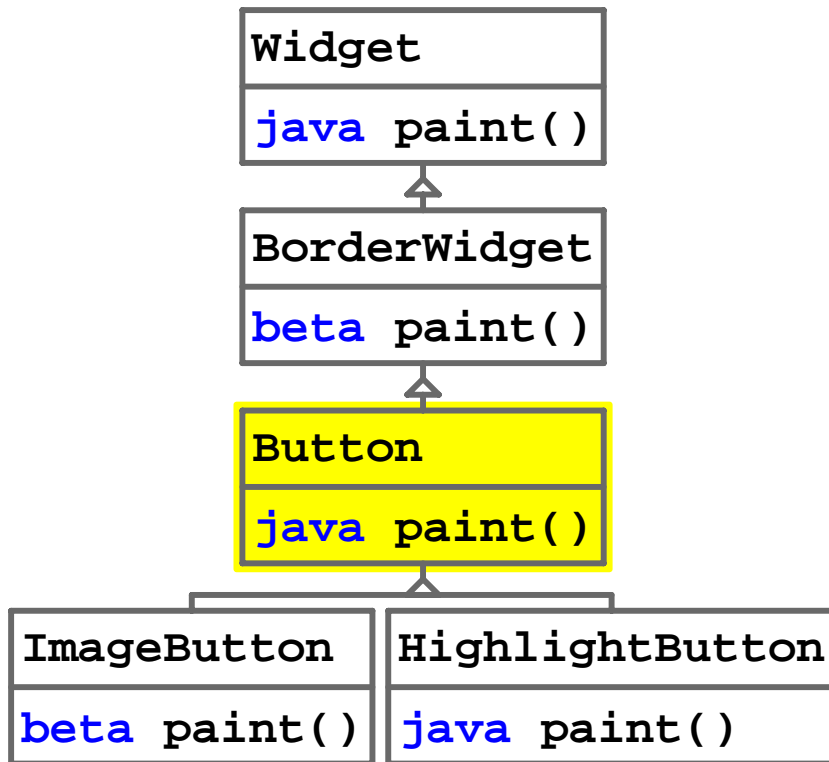
```
class Widget ...
    java void paint() {
        // paint the background:
        ...;
    }
```


Code for Beta/Java Combination



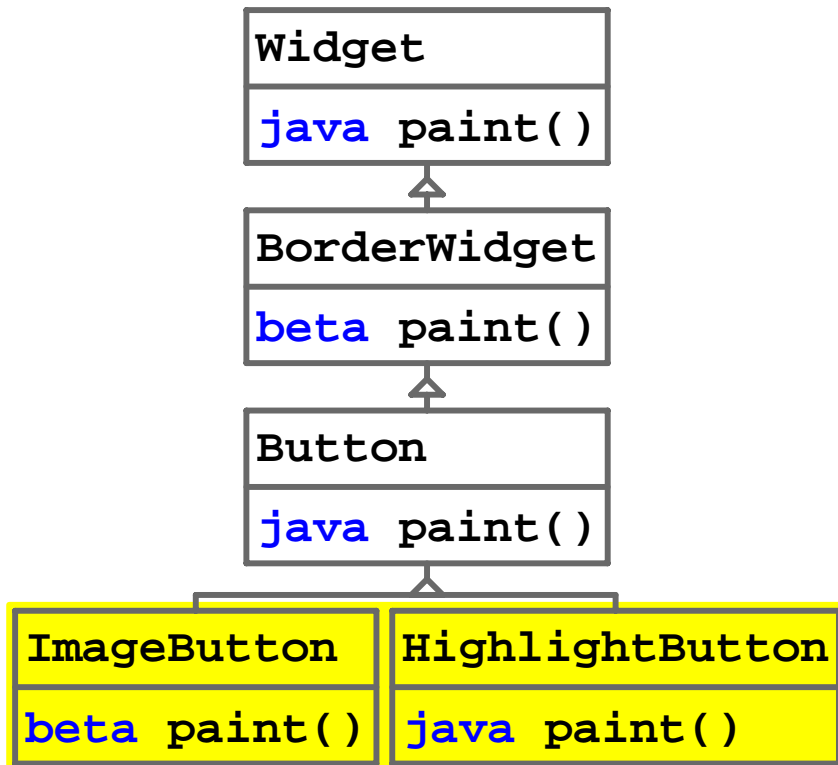
```
class BorderWidget ...
    beta void paint() {
        // paint background:
        super.paint();
        // draw a border:
        ...;
        inner.paint();
    }
```

Code for Beta/Java Combination



```
class Button ...
    java void paint() {
        // draw a button label:
        ...;
    }
```

Code for Beta/Java Combination



```
class ImageButton ...
    beta void paint() {
        // draw image,
        // don't call super
        // for label:
        ...;
        inner.paint();
    }
```

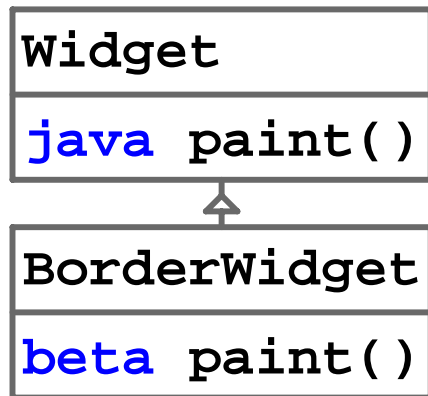
```
class HighlightButton ...
    java void paint() {
        // replace background with
        // blue gradient:
        ...;
        // draw label
        super.paint();
    }
```

Summary: Java to Beta/Java Combination

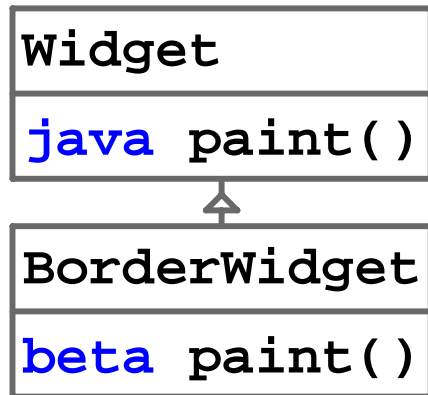
- Add **java** and **beta** keywords for methods
- Add **inner** statement:

inner . *Identifier* (*Expression*, ... *Expression*) **else** *Statement*

Operational Semantics: Calling paint on BorderWidget



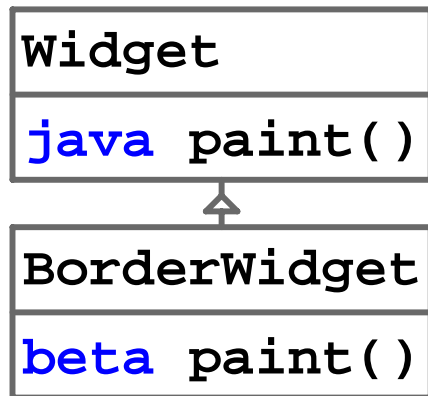
Calling paint on BorderLayout: Method Dispatch



```
new BorderLayout().paint();
```

Calling paint on BorderLayout: Method Dispatch

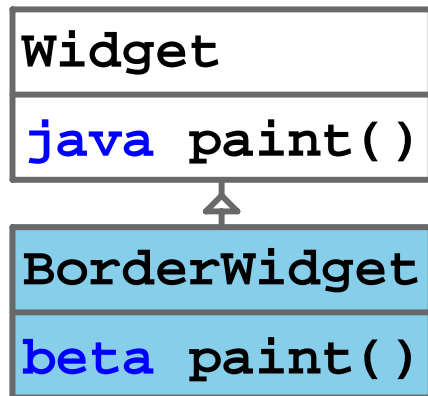
- Highest method declared **beta** or lowest method, if none



```
new BorderLayout().paint();
```

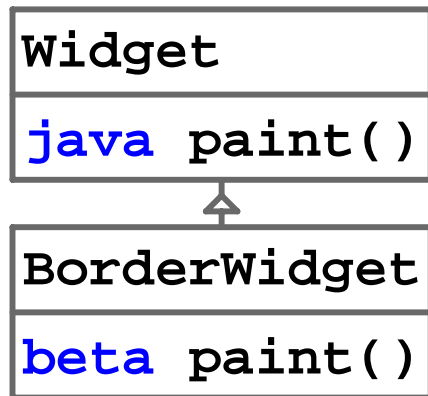
Calling paint on BorderLayout: Method Dispatch

- Highest method declared **beta** or lowest method, if none



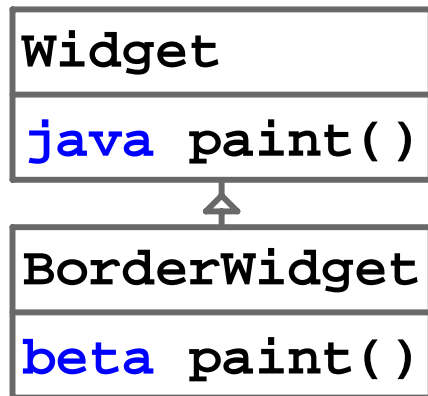
```
new BorderLayout().paint();
```


Calling paint on BorderLayout: Method Dispatch



```
class BorderLayout ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

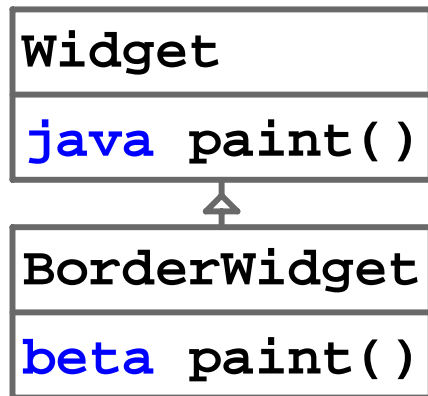
Calling paint on BorderLayout: super Dispatch



```
class BorderLayout ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

Calling paint on BorderLayout: super Dispatch

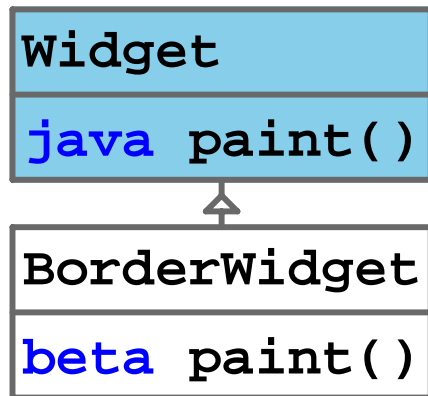
- Next method above in hierarchy



```
class BorderLayout ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

Calling paint on BorderLayout: super Dispatch

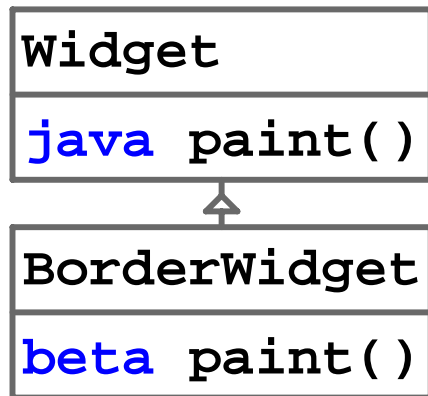
- Next method above in hierarchy



```
class BorderLayout ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

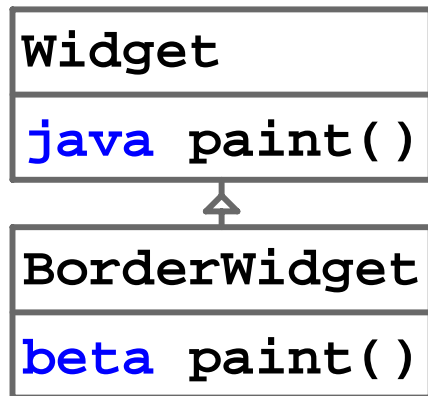
Calling paint on BorderLayout: super Dispatch

- Next method above in hierarchy



```
class Widget ...
    java void paint() {
        // paint the background:
        ...;
    }
```

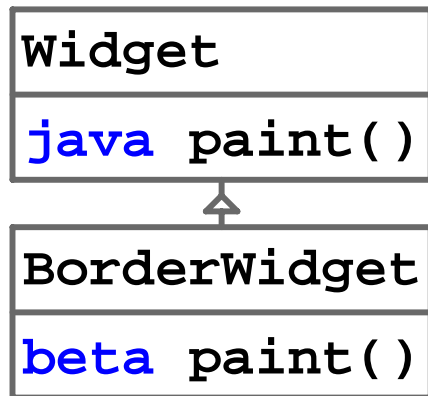
Calling paint on BorderLayout: inner Dispatch



```
class BorderLayout ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

Calling paint on BorderLayout: inner Dispatch

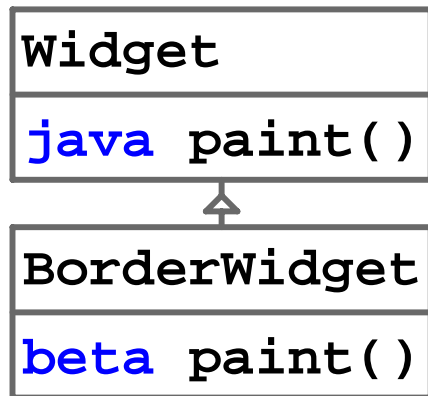
- Next method below declared **beta** or lowest method, if none



```
class BorderLayout ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

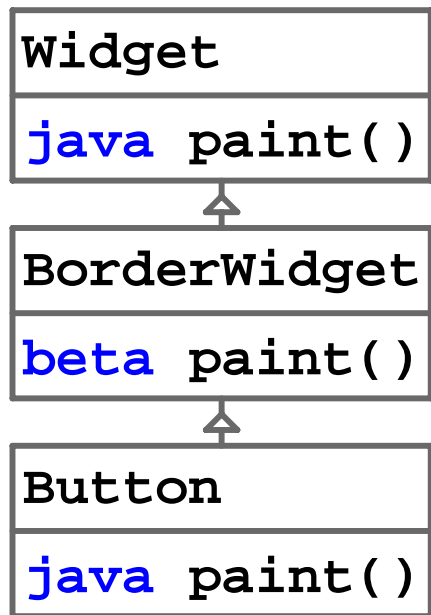
Calling paint on BorderLayout: inner Dispatch

- Next method below declared **beta** or lowest method, if none

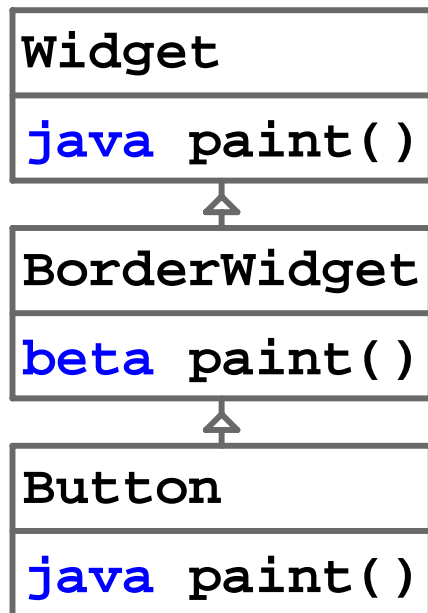


```
class BorderLayout ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```


Operational Semantics: Calling paint on Button



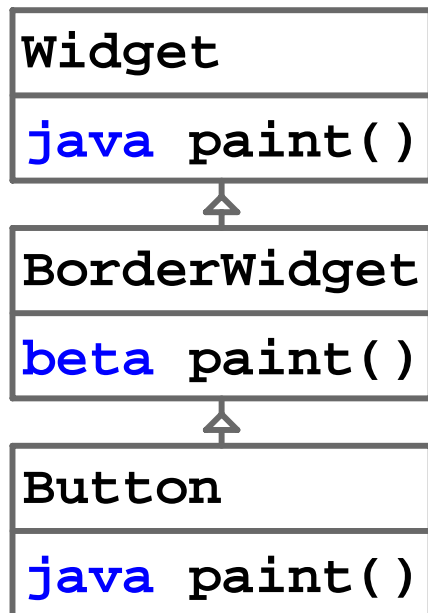
Calling paint on Button: Method Dispatch



```
new Button().paint();
```

Calling paint on Button: Method Dispatch

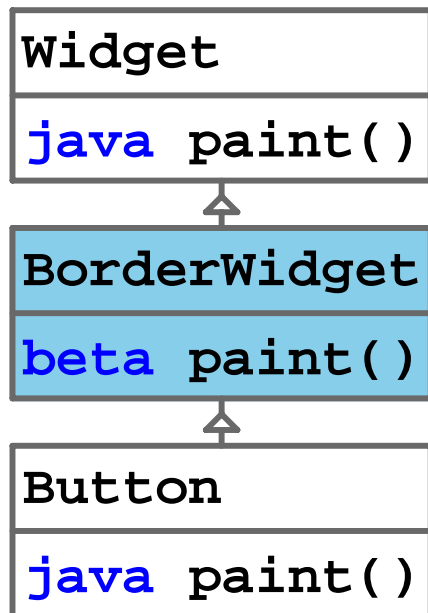
- Highest method declared **beta** or lowest method, if none



```
new Button().paint();
```

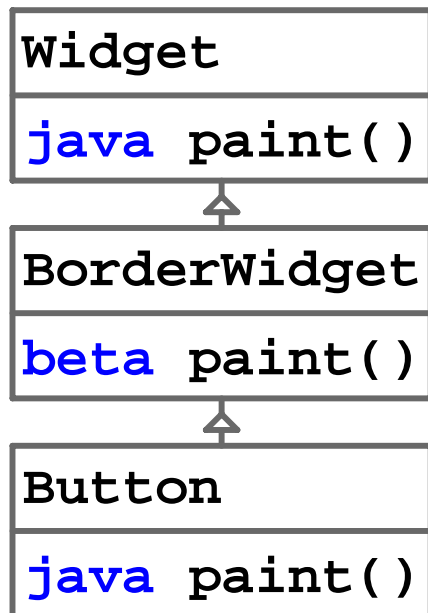
Calling paint on Button: Method Dispatch

- Highest method declared **beta** or lowest method, if none



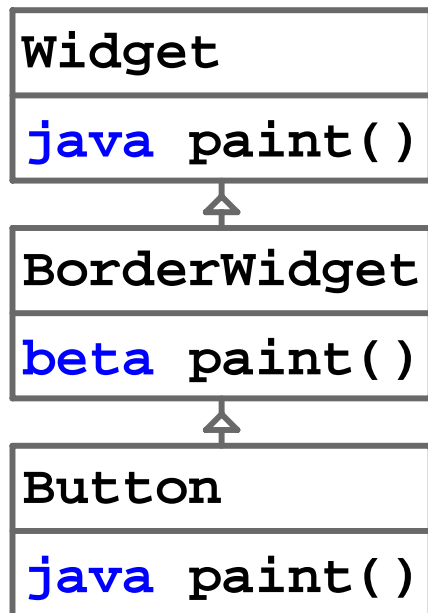
```
new Button().paint();
```

Calling paint on Button: Method Dispatch



```
class BorderWidget ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

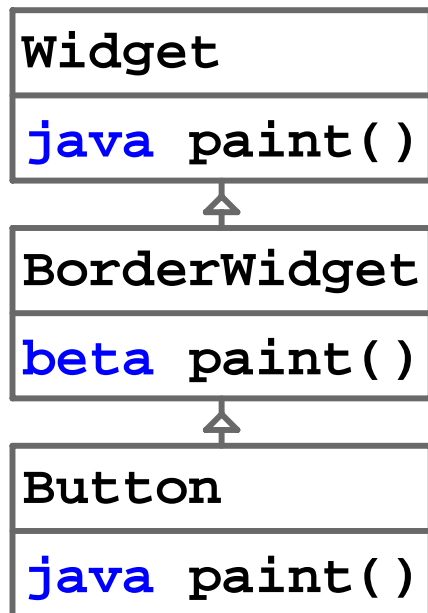
Calling paint on Button: inner Dispatch



```
class BorderWidget ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

Calling paint on Button: inner Dispatch

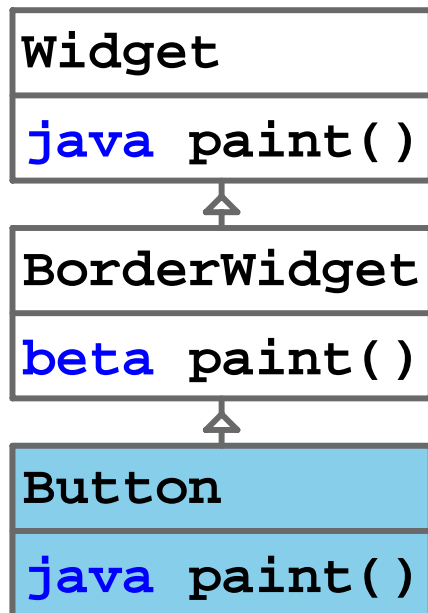
- Next method below declared **beta** or lowest method, if none



```
class BorderWidget ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

Calling paint on Button: inner Dispatch

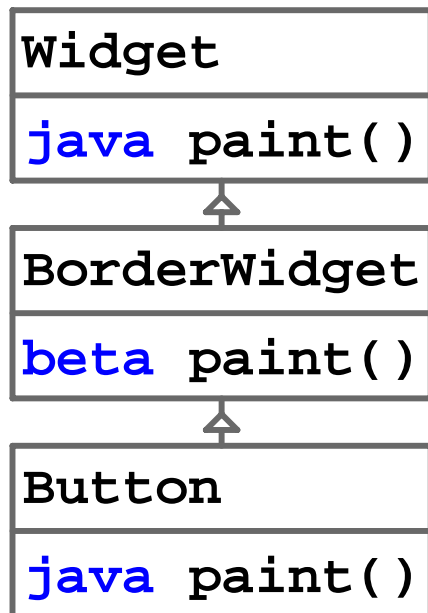
- Next method below declared **beta** or lowest method, if none



```
class BorderWidget ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

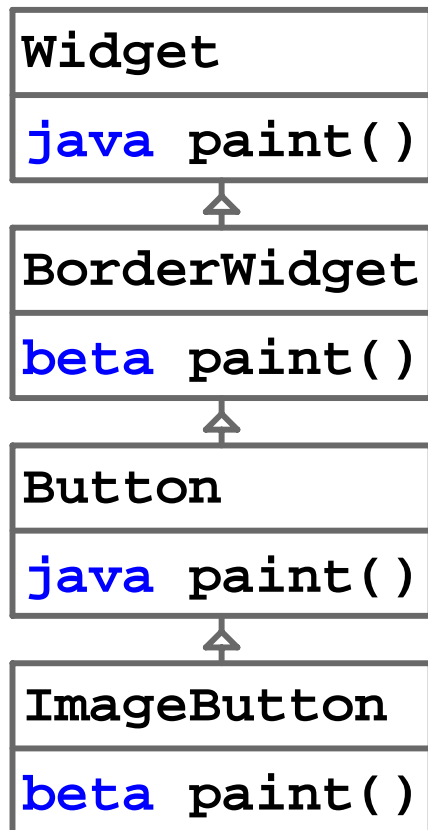

Calling paint on Button: inner Dispatch

- Next method below declared **beta** or lowest method, if none

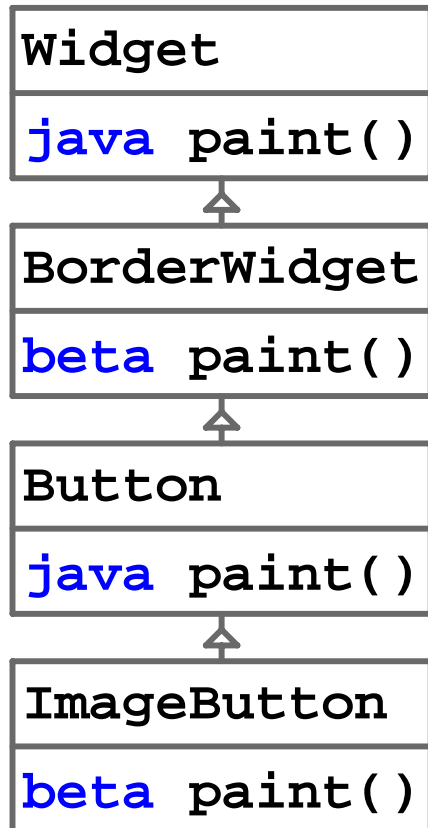


```
class Button ...
    java void paint() {
        // draw a button label:
        ...;
    }
```

Operational Semantics: Calling paint on ImageButton



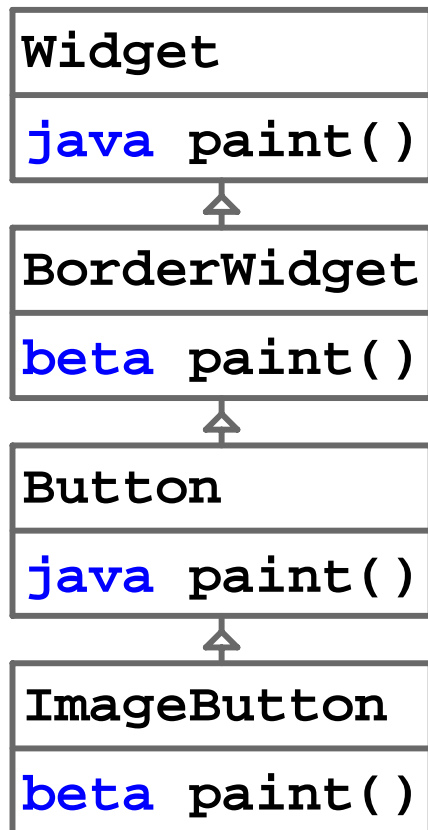
Calling paint on ImageButton: Method Dispatch



```
new ImageButton().paint();
```

Calling paint on ImageButton: Method Dispatch

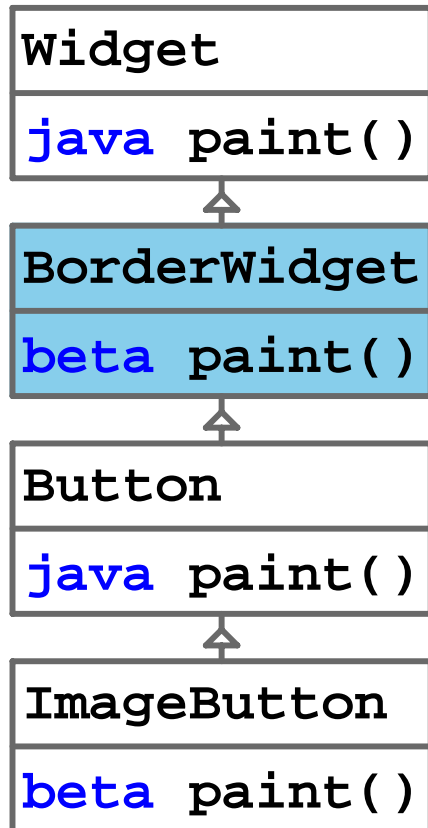
- Highest method declared **beta** or lowest method, if none



```
new ImageButton().paint();
```

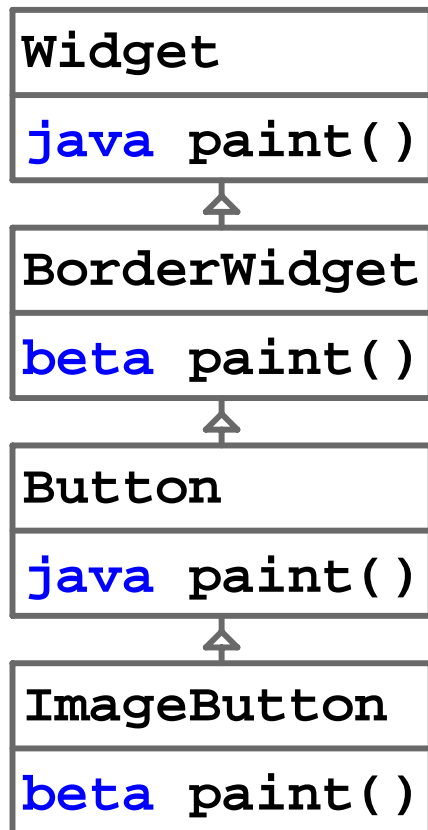
Calling paint on ImageButton: Method Dispatch

- Highest method declared **beta** or lowest method, if none



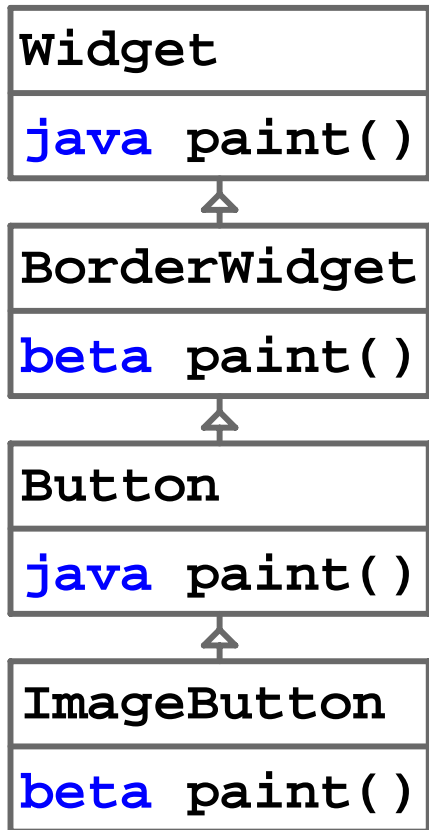
```
new ImageButton().paint();
```

Calling paint on ImageButton: Method Dispatch



```
class BorderWidget ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

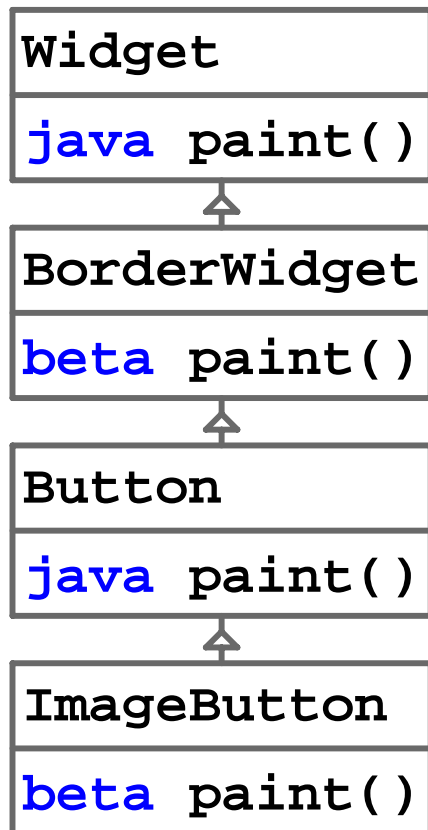
Calling paint on ImageButton: inner Dispatch



```
class BorderWidget ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

Calling paint on ImageButton: inner Dispatch

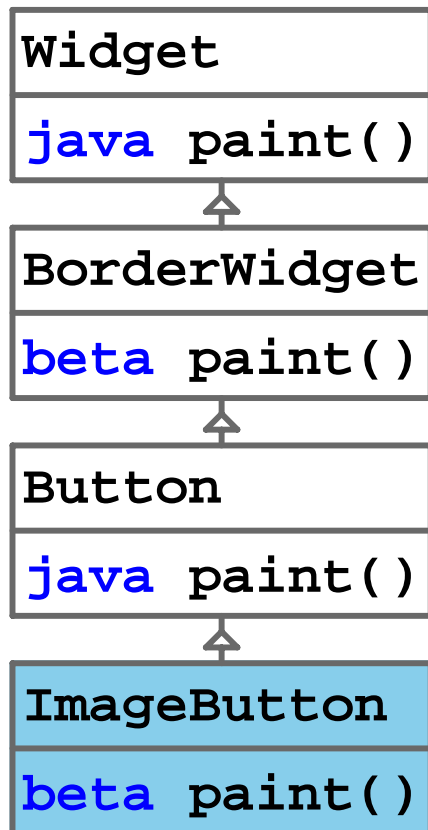
- Next method below declared **beta** or lowest method, if none



```
class BorderWidget ...  
    beta void paint() {  
        super.paint();  
        ...  
        inner.paint() else ...;  
    }
```


Calling paint on ImageButton: inner Dispatch

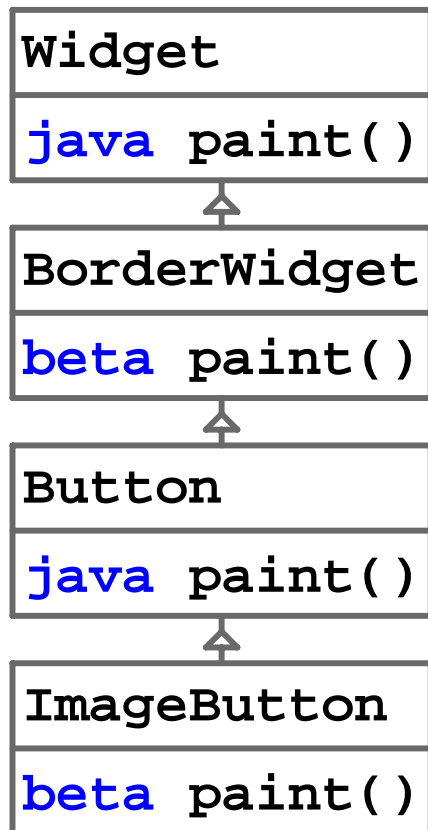
- Next method below declared **beta** or lowest method, if none



```
class BorderLayout ...  
    beta void paint() {  
        super.paint();  
        ...  
        inner.paint() else ...;  
    }
```

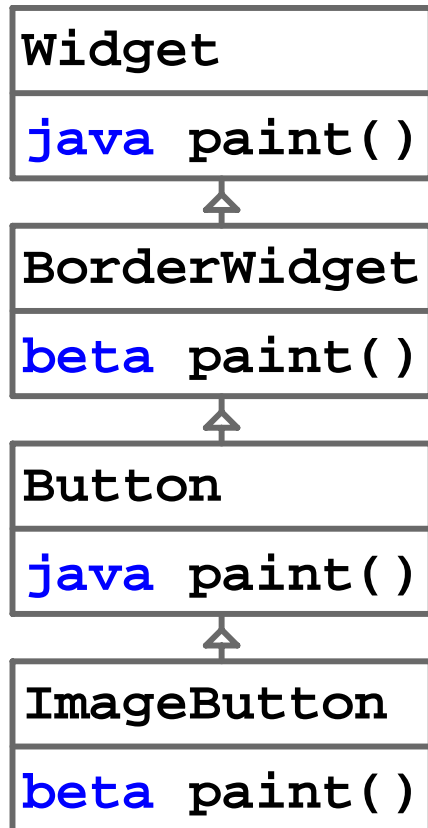
Calling paint on ImageButton: inner Dispatch

- Next method below declared **beta** or lowest method, if none

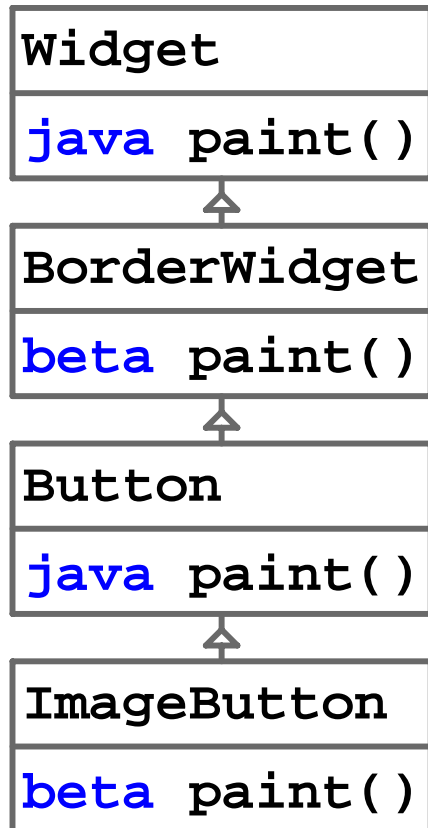


```
class ImageButton ...
    beta void paint() {
        // draw image,
        // don't call super
        // for label:
        ...;
        inner.paint();
    }
```

Compiling Method Dispatch



Compiling Method Dispatch



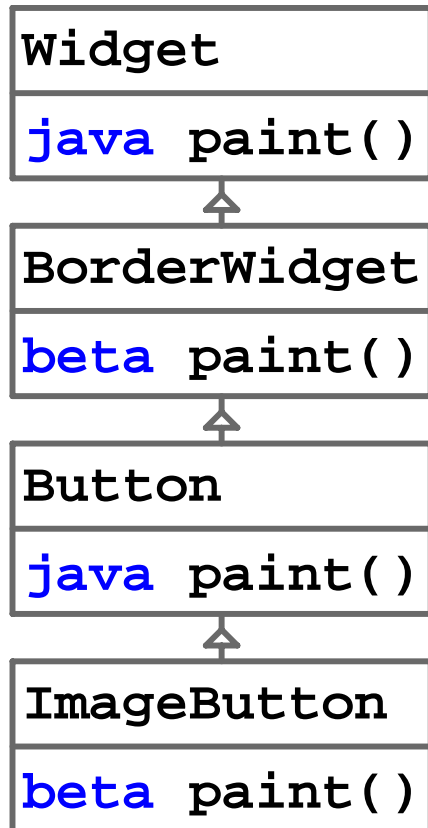
paint *Widget*

paint *BorderWidget*

paint *BorderWidget*

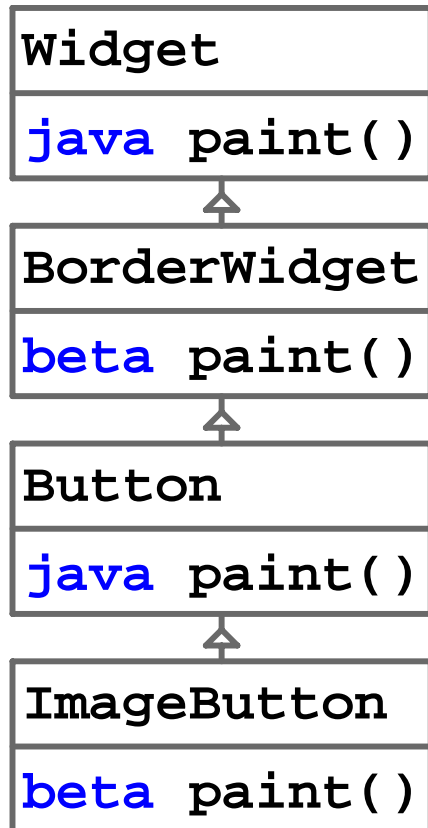
paint *BorderWidget*

Compiling super Calls



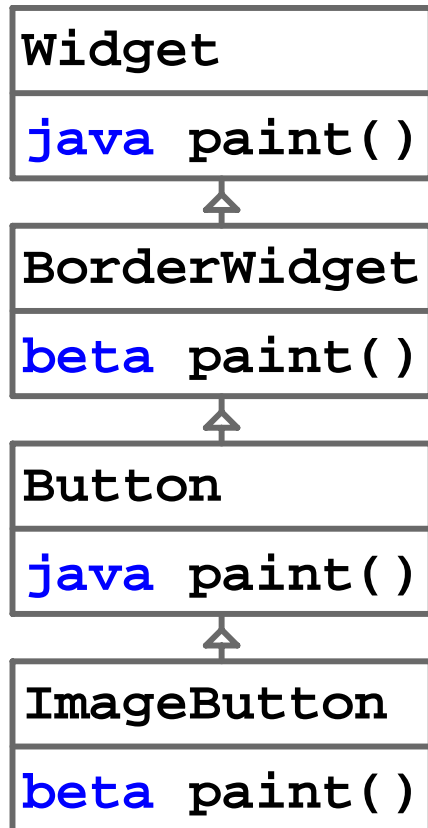
```
class BorderWidget ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

Compiling super Calls

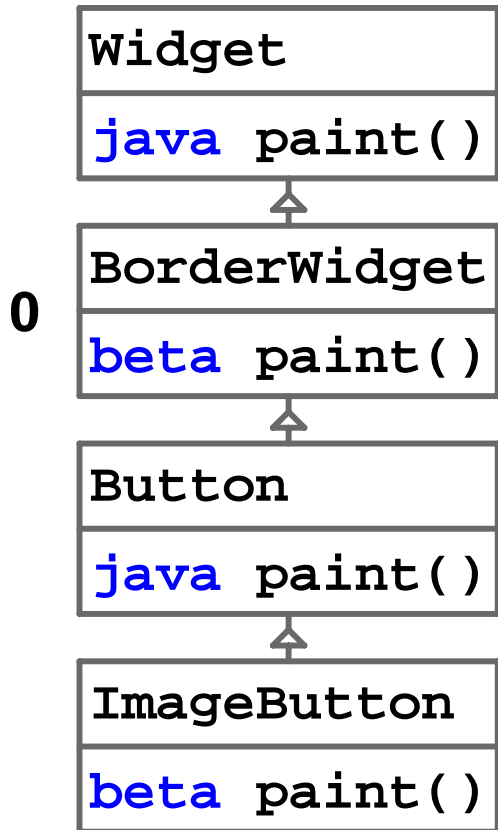


```
class BorderWidget ...
    beta void paint() {
        paint() in class Widget
        ...
        inner.paint() else ...;
    }
```

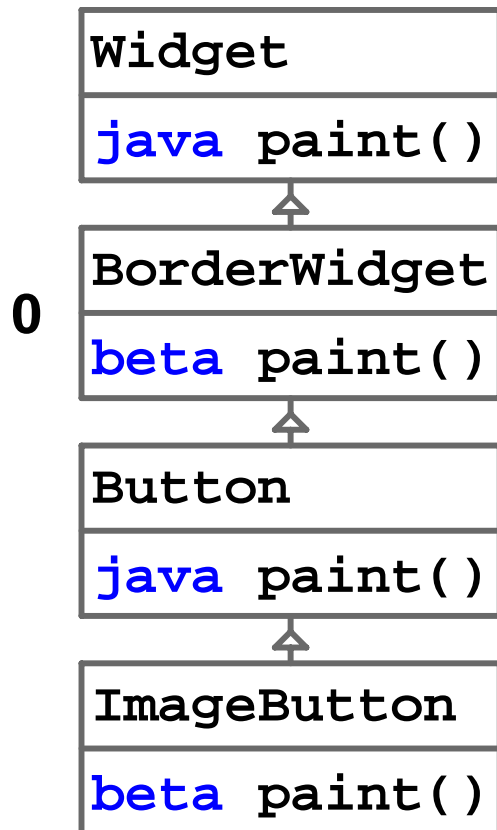
Compiling inner Calls



Compiling inner Calls

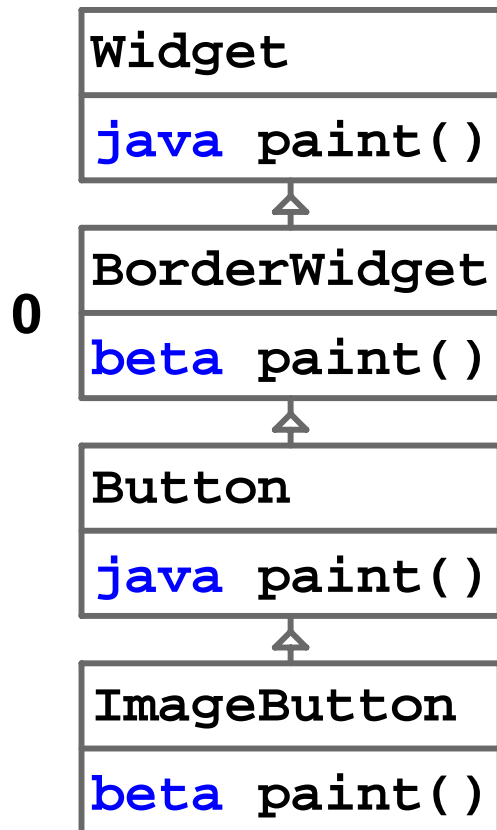


Compiling inner Calls



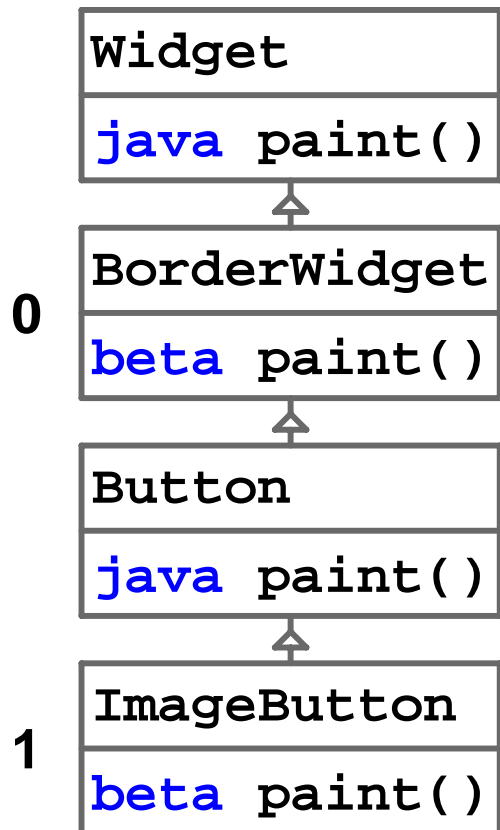
```
class BorderWidget ...
    beta void paint() {
        super.paint();
        ...
        inner.paint() else ...;
    }
```

Compiling inner Calls



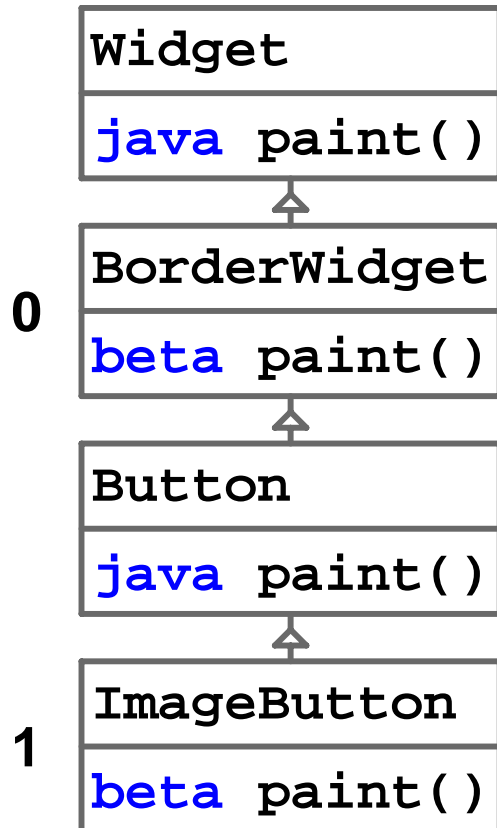
```
class BorderWidget ...
    beta void paint() {
        super.paint();
        ...
        paint() at index 1
    }
```

Compiling inner Calls



```
class BorderWidget ...  
    beta void paint() {  
        super.paint();  
        ...  
        paint() at index 1  
    }
```

Compiling inner Calls



paint	0
	<i>Widget</i>

paint	0	1
	<i>BorderWidget</i>	<i>null</i>

paint	0	1
	<i>BorderWidget</i>	<i>Button</i>

paint	0	1	2
	<i>BorderWidget</i>	<i>ImageButton</i>	<i>null</i>

Performance

- Performance unchanged with these modifications
 - All `java` methods behave exactly like normal Java
- Additional 250 lines of code to 2800-line object system

Model and Implementation

- Model based on CLASSICJAVA [Flatt, Krishnamurthi, Felleisen '98]
 - Soundness proof in paper
- Implemented in v299 of PLT Scheme

Use in Practice: on-close and render-value Methods

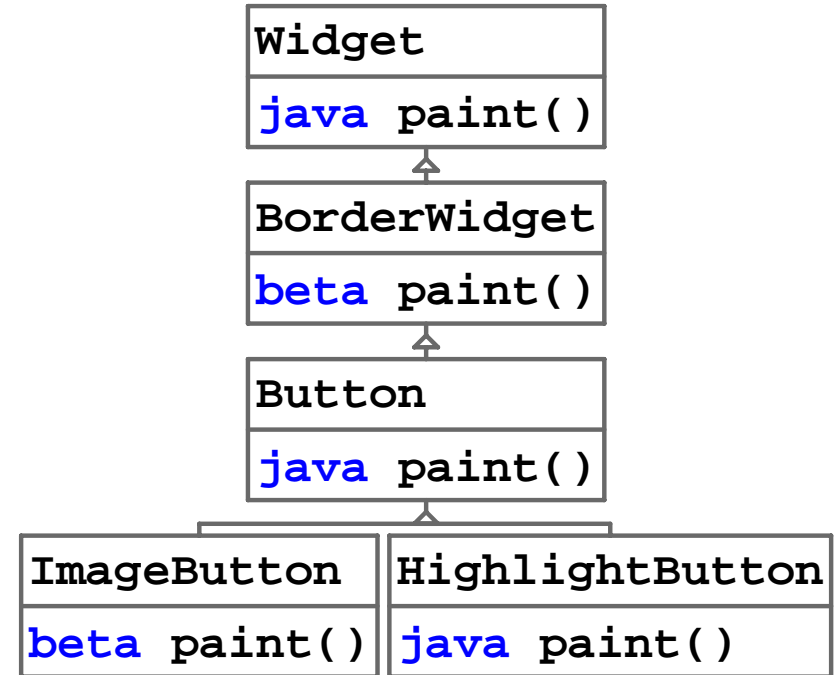
- **on-close** method of editor called when window closed
 - Includes important code to be executed
 - Must be in control
- **render-value** method of language converts values to strings
 - Sometimes language wants control — certain values always displayed the same
 - Sometimes gives up control to subclasses

Related Work

- CLOS [Keene, 1988],
- gbeta [Ernst, Ph.D. Thesis, 1999]
- Cook [Ph.D. Thesis, 1989],
Clark [GEC Journal of Research, 1994]
Bracha and Cook [OOPSLA 1990],
Ancona and Zucca [PLILP 1997, MSCS 1998],
Duggan [ECOOP 2000],
Bettini, et al. [WOOD 2003],
GNU EDMA [Oliveira]

Conclusions

- Programmers need **super** and **inner** in same language
 - Method can maintain control or give control as desired
- Developed simple system of determining control
 - Easy to implement



- Implementations:

<http://drscheme.plt-scheme.org/>

<http://www.cs.utah.edu/plt/super+inner/>