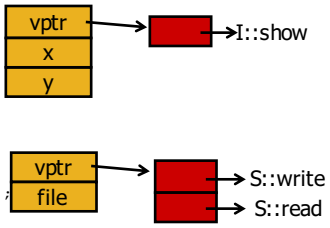
Multiple Inheritance & Java

CSCI 334
Stephen Freund

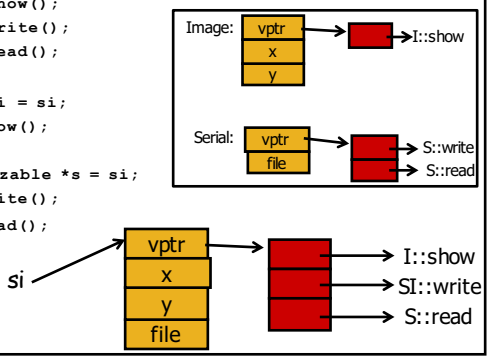
Multiple Inheritance

```
class Image {
  int x,y;
  virtual void show();
};

class Serializable {
  string file;
  virtual void write();
  virtual void read();
};

class SerialImage: public Image, public Serializable {
  virtual void write();
  virtual void show();
};
```



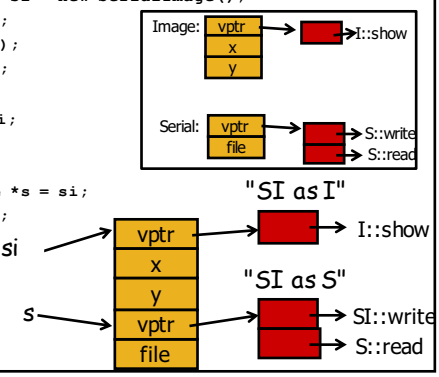
Multiple Inheritance (Not Good Yet...)

```
SerialImage *si = new SerialImage();
si -> show();
si -> write();
si -> read();

Image *i = si;
i -> show();

Serializable *s = si;
s -> write();
s -> read();
```



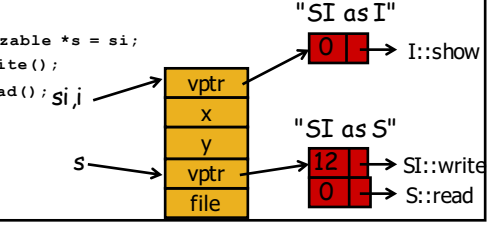
Multiple Inheritance (Not Good Yet...)

```
SerialImage *si = new SerialImage();
si -> show();
si -> write();
si -> read();

Image *i = si;
i -> show();

Serializable *s = si;
s -> write();
s -> read();
```



Multiple Inheritance

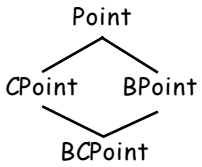
```
SerialImage *si = new SerialImage();
si -> show();
si -> write();
si -> read();

Image *i = si;
i -> show();

Serializable *s = si;
s -> write();
s -> read();
```



Name Clashes

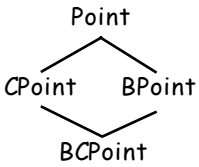
```
class Image {
  virtual void print();
};

class Serializable {
  virtual void print();
};

class SerialImage: public Image, public Serializable {
 ...
};

SerialImage *si = new SerialImage();
si -> print();
```

Diamond Pattern

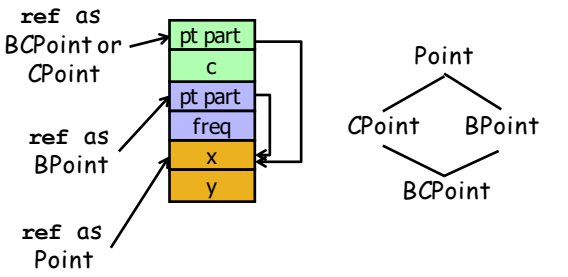



```
class Point {
  int x,y;
}
class ColorPoint : public Point {
  int color;
}
class BlinkingPoint : public Point {
  int freq;
}
class BlinkingColorCPoint :
        public ColorPoint, BlinkingPoint {
  ...
}
```

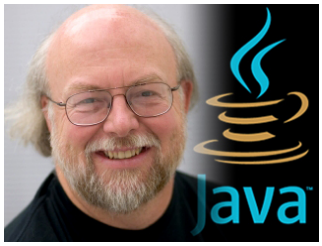
Diamond Pattern




```
class Point {
  int x,y;
}
class ColorPoint : public virtual Point {
  int color;
}
class BlinkingPoint : public virtual Point {
  int freq;
}
class BlinkingColorCPoint :
        public ColorPoint, BlinkingPoint {
  ...
}
```




```
void print(BPoint *p) { ... }
...

BCPoint *ref = new BCPoint(...);
print(ref);
```



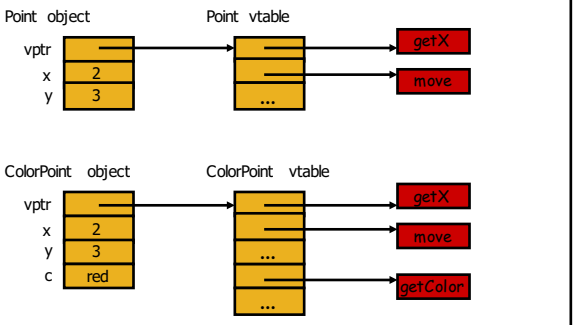
Java Class

```
class Point {
  private int x,y;

  public Point(int xv, int yv) {
    x = xv; y = yv;
  }

  public int getX() { return x; }
  public int getY() { return y; }

  public void move(int dx, int dy) {
    setLocation(x+dx, y+dy);
  }

  protected void setLocation(int xv, int yv) {
    x = xv; y = yv;
  }
```

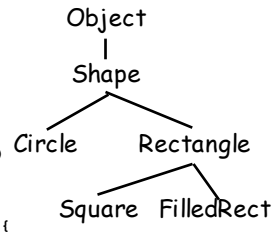
Java Derived Class

```
class ColorPoint extends Point {
  private int c;

  public ColorPoint(int xv, int yv, int cv) {
    super(xv, yv);
    c = cv;
  }

  public int getColor() { return c; }

  public void move(int dx, int dy) {
    super.move(dx, dy);
    darken(1);
  }

  public void darken(int tint) { ... }
}
```

Java Run-Time Representation (more later…)

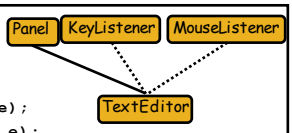


Final Methods




```
class Shape {
  boolean contains(Point p)
}

class Circle extends Shape {
  boolean contains(Point p)
}

class Rectangle extends Shape {
  final boolean contains(Point p)
}
```

Interfaces




```
interface KeyListener {
  void keyPressed(KeyEvent e);
  void keyReleased(KeyEvent e);
  void keyTyped(KeyEvent e);
}

interface MouseListener {
  void buttonClicked(MouseEvent e);
}

class TextEditor extends Panel
                 implements KeyListener,MouseListener {
  void keyPressed(KeyEvent e) { /* code */ }
  void keyReleased(KeyEvent e) { /* code */ }
  void keyTyped(KeyEvent e) { /* code */ }
  void buttonClicked(MouseEvent e) { /* code */ }
}
```

Interfaces vs. Multiple Inheritance

```
class DefaultKeyListener {
  // default is to do nothing
  void keyPressed(KeyEvent e)  { }
  void keyReleased(KeyEvent e) { }
  void keyTyped(KeyEvent e) { }
}

class TextField : public Panel, DefaultKeyListener {
  void keyTyped(KeyEvent e) { /* code here */ }
  ...
}

class TerminalWindow : public Panel, DefaultKeyListener
  void keyTyped(KeyEvent e) { /* code here */ }
  ...
}
```

Interfaces vs. Multiple Inheritance

```
interface KeyListener {
  void keyPressed(KeyEvent e);
  void keyReleased(KeyEvent e);
  void keyTyped(KeyEvent e);
}

class TextField extends Panel implements KeyListener {
  void keyPressed(KeyEvent e) { }
  void keyTyped(KeyEvent e) { /* code here */ }
  void keyReleased(KeyEvent e) { }
}

class TerminalWindow extends Panel implements KeyListener
  void keyPressed(KeyEvent e) { }
  void keyTyped(KeyEvent e) { /* code here */ }
  void keyReleased(KeyEvent e) { }
}
```