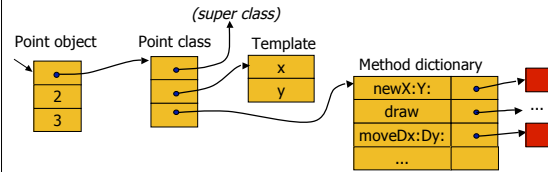


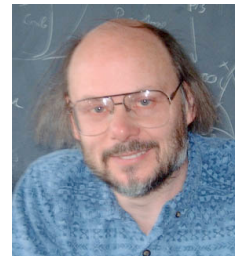
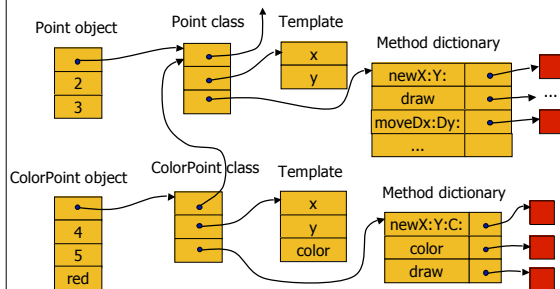
C++

CSCI 334
Stephen Freund

Run-time Representation



Run-time Representation



Stroustrup Quotes

- "C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do, it blows away your whole leg."
- "I have always wished that my **computer** would be as easy to use as my **telephone**. My wish has come true. I no longer know how to use my **telephone**."

... in my group ... we do use C++ regularly and find it very useful but certainly not perfect. Every full moon, however, we sacrifice a virgin disk to the language gods in hopes that the True Object-Oriented Language will someday be manifest on earth, or at least on all major platforms...

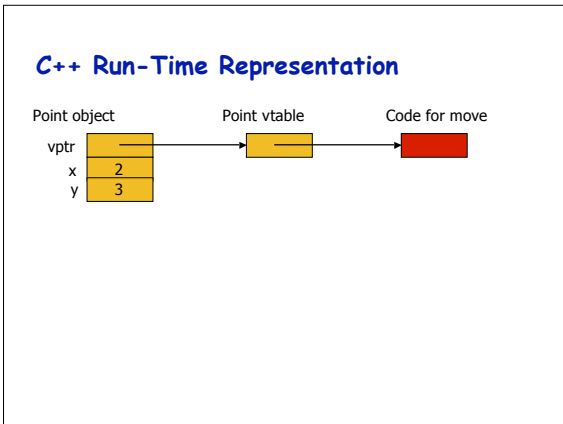
Rick Pember, LLNL

```

class Point {
private:
    int x,y;
public:
    Point(int xv, int yv);
    int getX();
    int getY();
    virtual void move(int dx, int dy);
protected:
    void setLocation(int xv, int yv);
};

Point::Point(int xv, int yv) { x = xv; y = yv; }
int Point::getX() { return x; }
int Point::getY() { return y; }
void Point::setLocation(int xv, int yv) { x = xv; y = yv;}
void Point::move(int dx, int dy) {setLocation(x+dx,y+dy);}

```



```

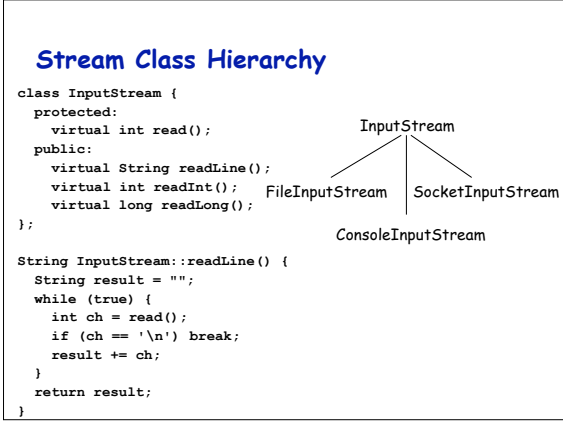
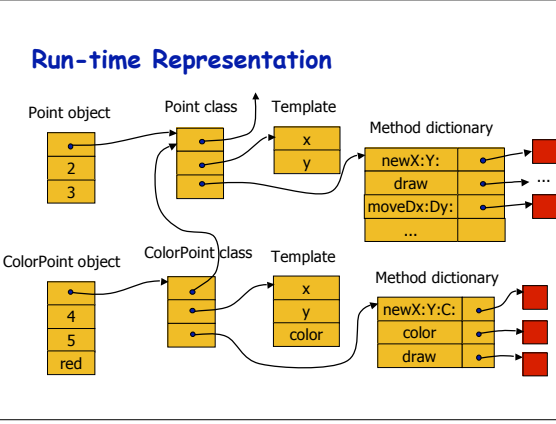
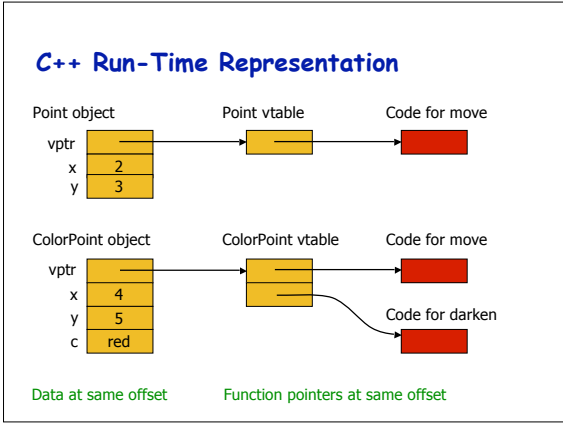
class ColorPoint: public Point {
private:
    int c;
public:
    ColorPoint(int xv, int yv, int cv);
    int getColor(void);
    virtual void move(int dx, int dy);
    virtual void darken(int tint);
};

ColorPoint::ColorPoint(int xv, int yv, int cv):
    Point(xv, yv)
    { c = cv; }

int ColorPoint::getColor(void) { return c; }

void ColorPoint::move(int dx, int dy)
    { Point::move(dx, dy); darken(1); }

```



Stream Class Hierarchy

```
class InputStream {
protected:
    virtual int read();
public:
    String readLine();
    int readInt();
    long readLong();
};

String InputStream::readLine() {
    String result = "";
    while (true) {
        int ch = read();
        if (ch == '\n') break;
        result += ch;
    }
    return result;
}
```

```
graph TD
    InputStream --> FileInputStream
    InputStream --> SocketInputStream
    InputStream --> ConsoleInputStream
```

Stream Class Hierarchy

```
class FileInputStream : public InputStream {
protected:
    virtual int read();
};

int FileInputStream::read() { ... }

class SocketInputStream : public InputStream {
protected:
    virtual int read();
};

int SocketInputStream::read() { ... }
```

Overloading vs. Virtual Methods

```
class Point {
    void printClass() { cout << "Point"; }
    virtual void printClassVirtual() { cout << "Point"; }
};

class ColorPoint : public Point {
    void printClass() { cout << "ColorPoint"; }
    virtual void printClassVirtual() { cout << "ColorPoint"; }
};

Point *p = new Point();
p->printClass();           Point
p->printClassVirtual();   Point

ColorPoint *cp = new ColorPoint();
cp->printClass();         ColorPoint
cp->printClassVirtual(); ColorPoint
```

Overloading vs. Virtual Methods

```
class Point {
    void printClass() { cout << "Point"; }
    virtual void printClassVirtual() { cout << "Point"; }
};

class ColorPoint : public Point {
    void printClass() { cout << "ColorPoint"; }
    virtual void printClassVirtual() { cout << "ColorPoint"; }
};

Point *p = new ColorPoint();

p->printClass();           Point
p->printClassVirtual();   ColorPoint
```

Overriding vs. Overloading

```
class Point {
    virtual boolean equals(Point *p) // 1
    { ... }
};

class ColorPoint : public Point {
    virtual boolean equals(Point *p) // 2
    { ... }

    virtual boolean equals(ColorPoint *cp) // 3
    { ... }
};
```

```
Point *p1 = new Point();
Point *p2 = new ColorPoint();
ColorPoint *cp = new ColorPoint();

p1->equals(p1);
p1->equals(p2);
p1->equals(cp);

p2->equals(p1);
p2->equals(p2);
p2->equals(cp);

cp->equals(p1);
cp->equals(p2);
cp->equals(cp);
```