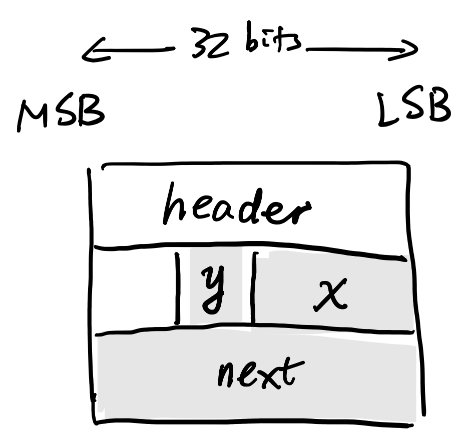
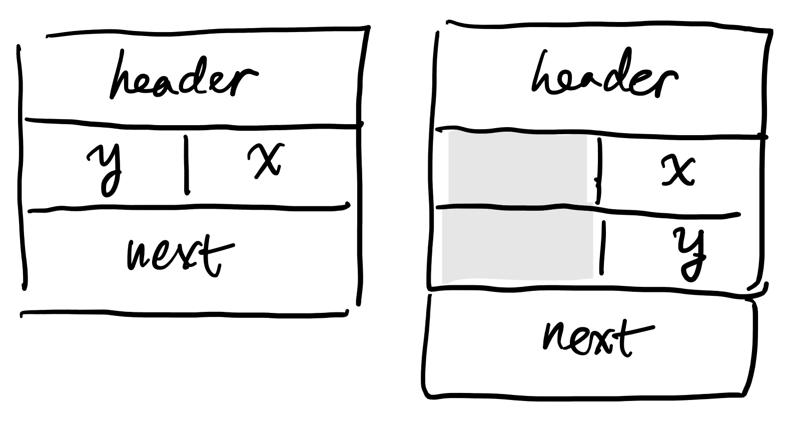
Compiling classes and objects

Layout

C++:
```
class C {
    short x;    16 bits
    char  y;    8 bits
    C*  next;   32 bits
}
```



Java.
```
class C {
    short x;    16 bits
    char  y;    16 bits
    C   next;   32 bits
}
```



Method dispatch

Java.
```
class Point {
    int x, y;
    void setX(int x) { this.x = x; }
    void moveX(int dx)
      { this.setX(this.x + dx); }
}
```

```
class ColorPoint extends Point {
  Color c;
  void setX(int x) {
     this.x = x;
     this.c = Colors.redden();
  }
  Color getColor() { return this.c; }
}
```

```
Point p = new ColorPoint();
p.setX(42);    redder
p.moveX(1);    redder
```

Given o.m(..), find correct code for m based on run-time type of o.

Smalltalk.

Objects point to "class objects"   String.class   java.lang.Class

On a call, walk up hierarchy to find first match method in class object.

+ code sharing

- slow

Java/Joos.

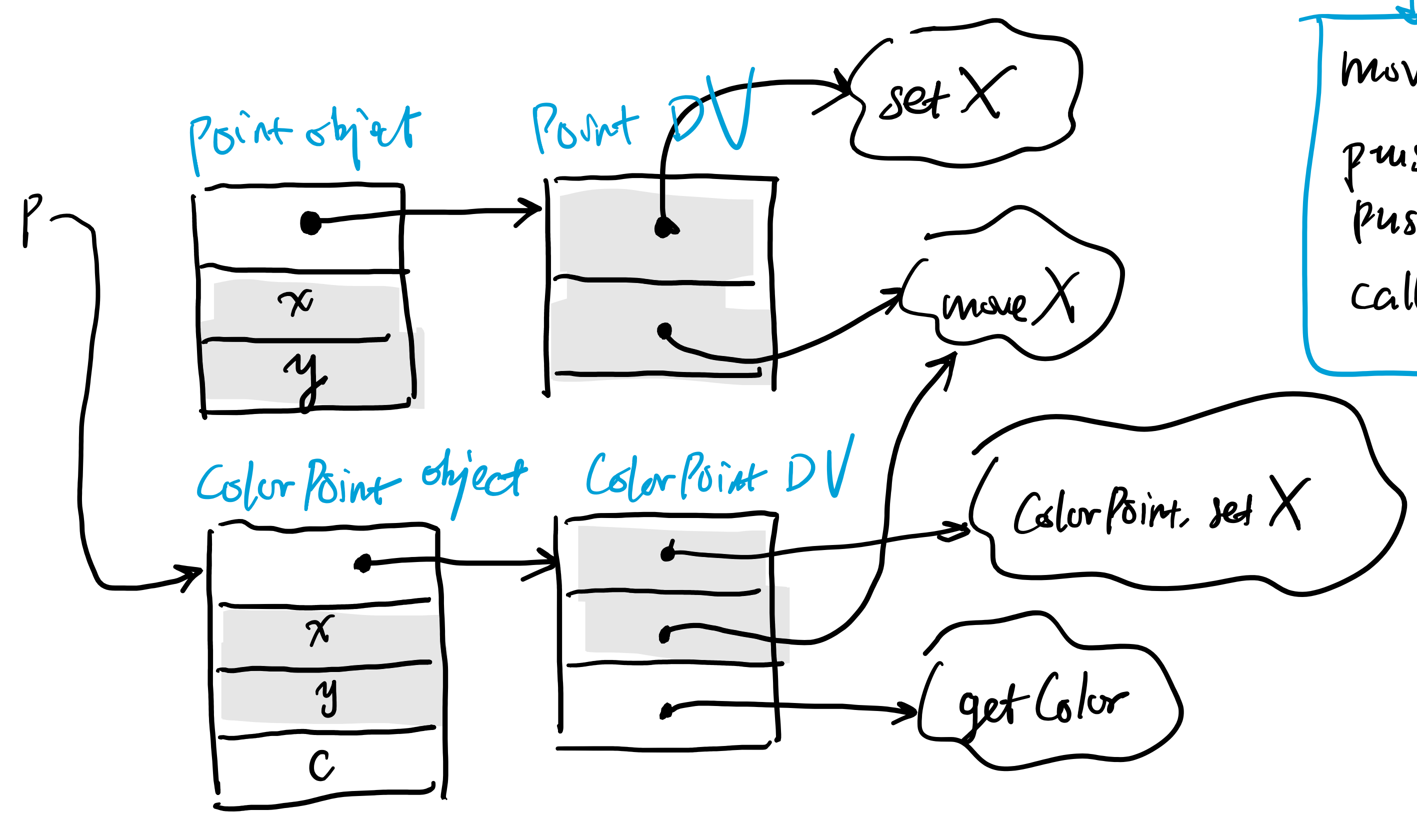
Attach dispatch vectors to object = array of code pointers

aka vtables

```
Point p = new Point();
p = new ColorPoint();
p.moveX(1);
```

TIR
```
MOVE(t_DV, MEM(p))
CALL(MEM(t_DV + 4), p, 1)
```

x86
```
mov t_DV, [p]
push 1
push p
call [t_DV + 4]
```



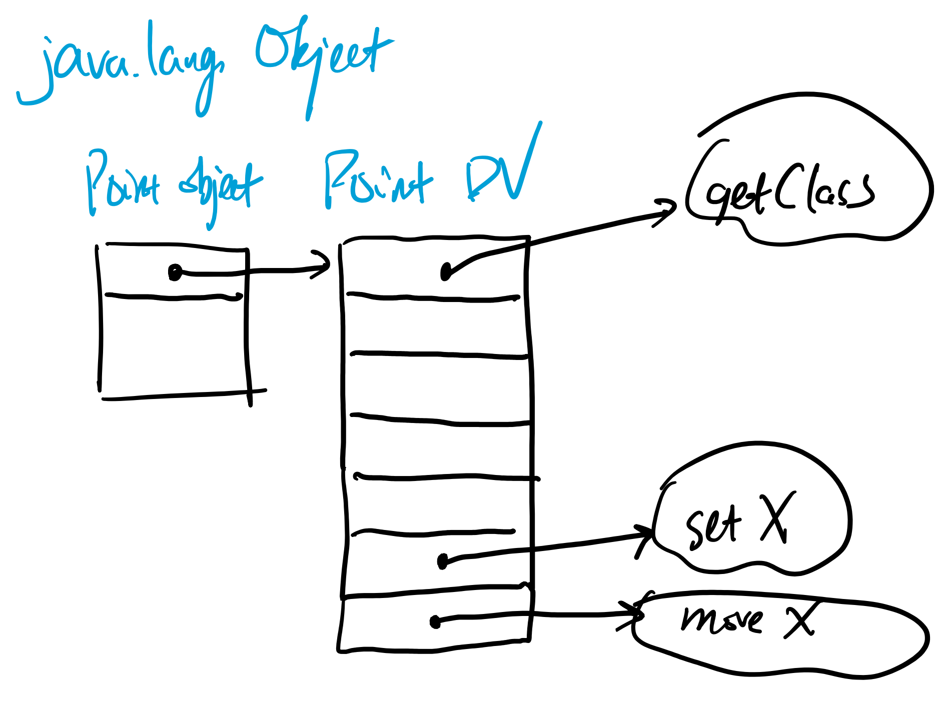
Translate field access

p.y $\rightsquigarrow$ MEM(p+8)

Abstract classes

no DVs. But a DV layout.

compile code for non-abstract methods



JVM

p.getColor()

↓ javac

bytecode invokevirtual "ColorPoint.getColor"

↓ load / rewrite

invokevirtual-quick 3 $\rightsquigarrow$ opt. call getColor directly inline!

↓ JIT

```
move t_DV, [p]
   :
call [t_DV + ...]
```