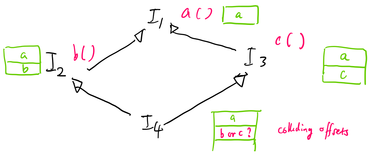
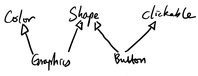
Compiling Interface Method Dispatch



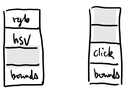
Idea #1. Assign indices to not collide.

```
interface Color {          interface Clickable {      interface Shape {
  RGB  rgb()                 void click()               Box  bounds
  HSV  hsv()               }                          }
}
class Graphics implements Shape, Color { ... }
class Button  ------------ Shape, Clickable { ... }
```

no separate compilation

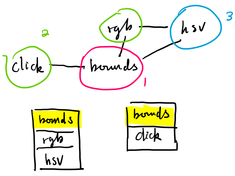


rgb 1
hsv 2
click 3
bounds 4



Want: densely packed DVs

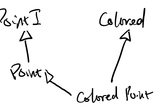
Idea: draw interference graph



Idea #2 Compute indices by hashing.

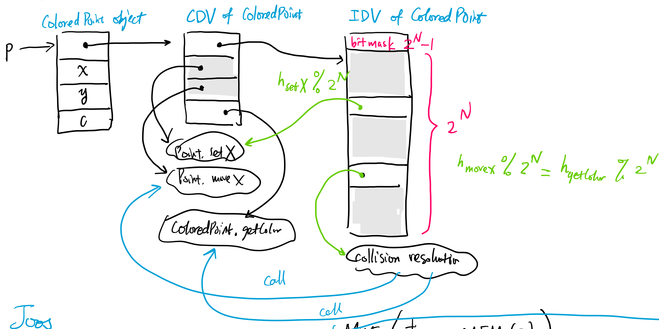
```
interface PointI {
  void setX(int)
  void moveX(int)
}
class Point implements PointI {
  int x, y;
  ...
}
```

```
interface Colored {
  Color getColor();
}
class ColoredPoint
  extends Point implements Colored {
  ...
}
```



IDV ≈ hash table

h_{setX} := precomputed hash of PointI.setX(int)



Java

```
PointI p = new ColoredPoint();
p.setX(42);
```

IR

```
MOVE(t_CDV, MEM(p))
MOVE(t_IDV, MEM(t_CDV))
MOVE(t_setX, AND(h_setX, MEM(t_IDV - 4)))
CALL(MEM(t_IDV + 4 * t_setX), id_setX, p, 42)
```

x86

```
mov  t_CDV, [p]
mov  t_IDV, [t_CDV]
mov  t_setX, h_setX
and  t_setX, [t_IDV - 4]
push 42
push p
push id_setX
call [t_IDV + 4 * t_setX]
```

8 instructions vs. 4 instructions

interface m.d. class m.d.

↯ cache miss

separate compilation ✓

Optimization: inline caching.

slowDispatch(o, m) assumed.

```
line 444 :  o.m()
```

cache is global data

class of o seen last time

code of m invoked ---

```
line 444:  mov  t_id, [o]
           cmp  t_id, [id 444]
           jnz  miss 444
           call [code 444]
done 444:  ---
miss 444:  call slowDispatch(o, m)
           mov  [id 444], t_id
           mov  [code 444], ...
           jmp  done 444
```

update cache

Instance of

o instance of C

(C) o

arr[i] = o

Idea #1.

Use CDV as class identity.

Store in CDV ptrs to parent's CDV,

+ separate compilation

− linear search / tree search

Idea #2.

Use a global M×M table M = # types

table[i][j] = T_i is a subtype of T_j

+ O(1) + multiple inh.

− O(M^2) − separate compilation