

```
Echec
int main (int argc, char *argv[])
{
Point& p1 = * new ColorPoint (1, 2, "red");
Point& p2 = * new ColorPoint (1, 2, "green");

std::cout << p1.equal (p2) << std::endl;
// => True. ### Wrong!
}

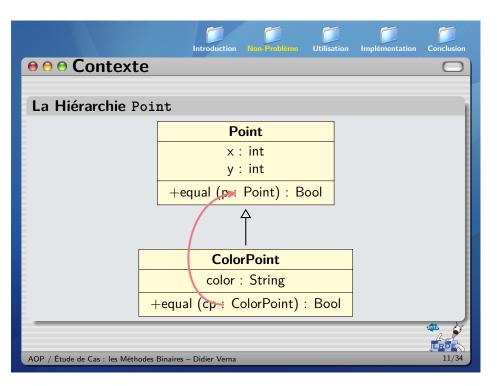
ColorPoint::equal masque Point::equal (statiquement)

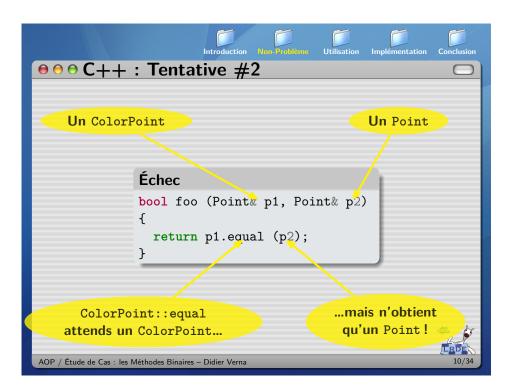
Seule Point::equal est vue depuis la classe de base

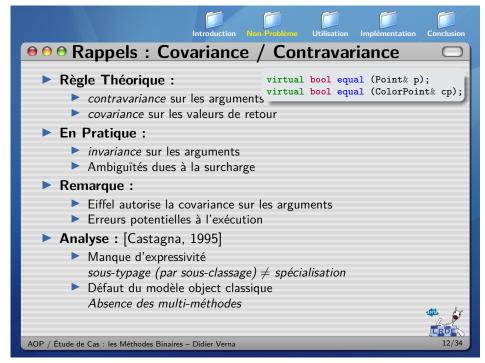
On a besoin de la version exacte
```

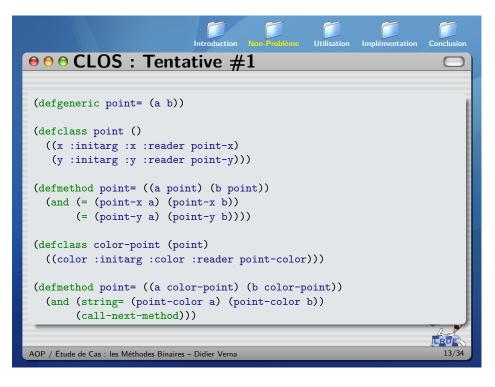
```
class Point
{
  int x, y;
  virtual bool equal (Point& p)
  { return x == p.x && y == p.y; }
};

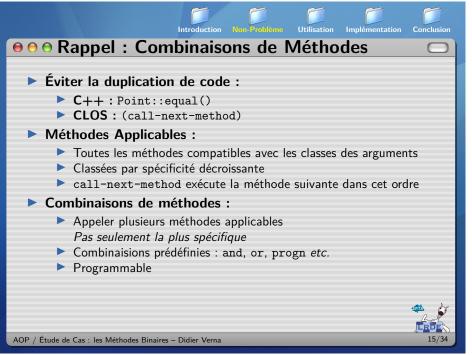
class ColorPoint : public Point
{
  std::string color;
  virtual bool equal (ColorPoint& cp)
  { return color == cp.color && Point::equal (cp); }
};
```

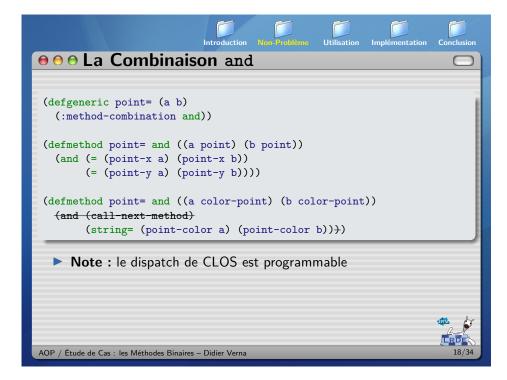


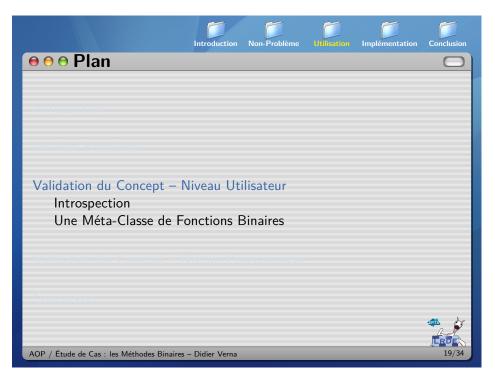


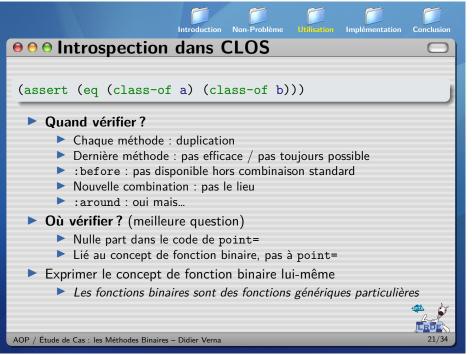


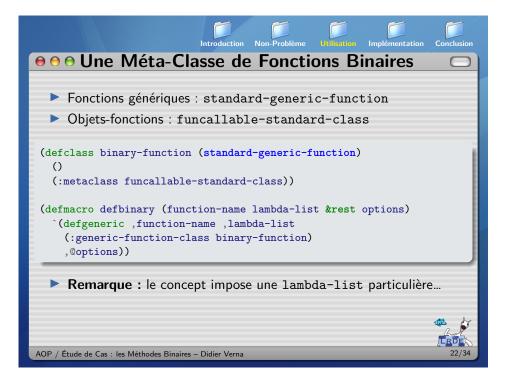


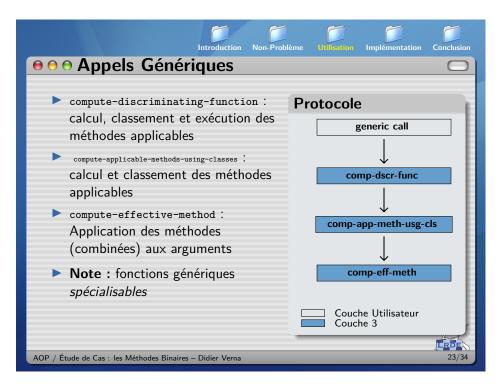


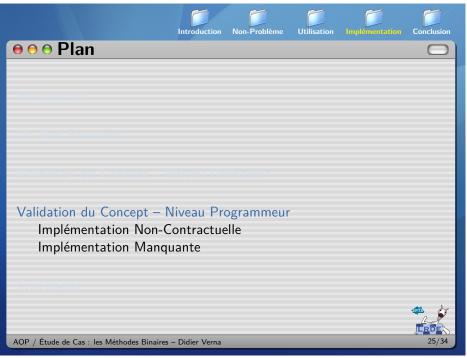








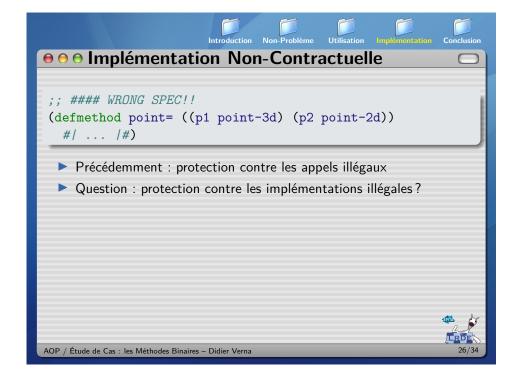


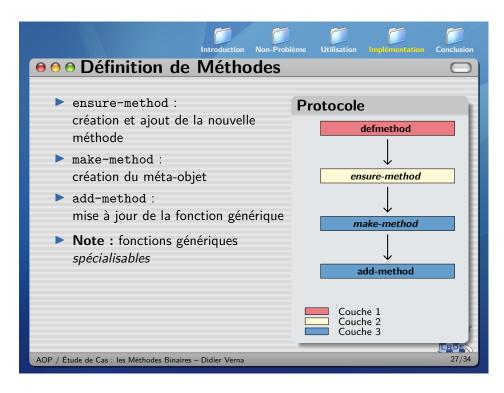


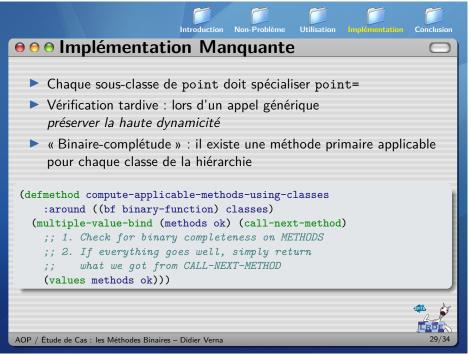
```
Introduction Non-Problème United Implémentation Conclusion

O Application aux Fonctions Binaires

(defmethod compute-applicable-methods-using-classes
:before ((binary-function binary-function) classes)
(assert (= (length classes) 2))
(assert (apply #'eq classes)))
```







```
● ● ● Mise en oeuvre
 (loop :for class :in (class-precedence-list (car classes))
       :until (eq class (find-class 'standard-object))
       :do (assert
            (find-if
             (lambda (method)
               (let ((qualifiers (method-qualifiers method)))
                 (and (equal (method-specializers method)
                             (list class class))
                      (not (member :around qualifiers))
                      (not (member :before qualifiers))
                      (not (member :after qualifiers)))))
             methods)))
  class-precedence-list:accesseur
  ▶ method-qualifiers : accesseur
AOP / Étude de Cas : les Méthodes Binaires - Didier Verna
```

