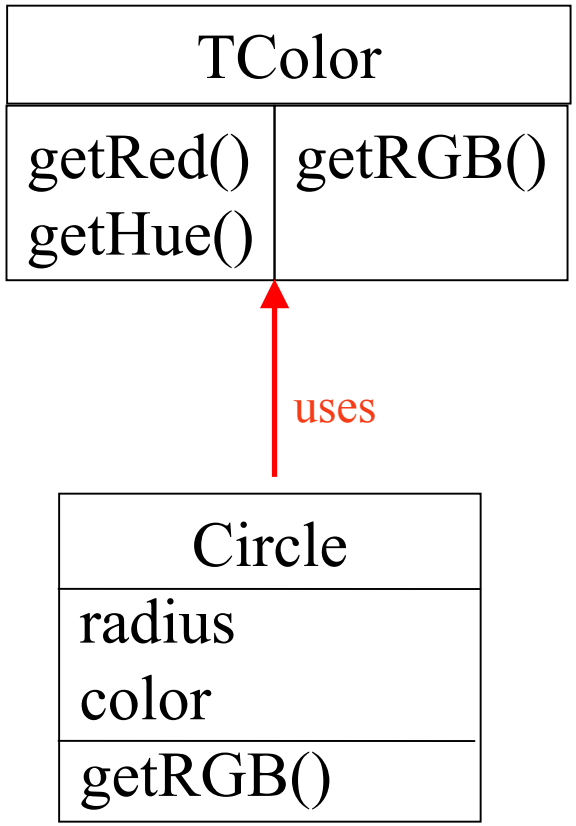
Introducing a flavor of Traits in Java with AspectJ

P. Cointe and S. Denier
OBASCO group

Bern 04/06/23

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE


TColor
getRed()
getHue()
getRGB()
uses
Circle
radius
color
getRGB()

Squeak Trait = Java Interface + AspectJ Introduction

```
interface TColorRequirement {
   public java.awt.Color getRGB();
}

interface TColor extends TColorRequirement {
   public int getRed();
   pubic float getHue();
}
```

AspectJ ’introduction

```
aspect TColorRepository {
   public int TColor.getRed() {…};
   public int TColor.getHue() {…};
}
```

AspectJ ’introduction

```
aspect TColorRepository {
   public int TColor.getRed() {…};
   public int TColor.getHue() {…};
}
```

uses = implements

```
class Circle implements TColor {
   private java.awt.Color,
   private int radius,

   public java.awt.Color getRGB() {
      return color;
}
```

uses = implements

```
class Circle implements TColor {
   private java.awt.Color,
   private int radius,
   public int getRed() {…};
   public int getHue() {…};
   public java.awt.Color getRGB() {
      return color;
}
```

Composition?

```
class Circle implements TColor, TLocation, ... {
    ...
}
```

Conflicts detection / resolution

- done by the Aspectj 'weaver in a given (ad hoc) way
- need to be explicited (dark side of AspectJ)
  - typing issue when two « weaved » methods only differ by their return type.
  - Introducing aliasing at the AspectJ user'level