Decorator Pattern

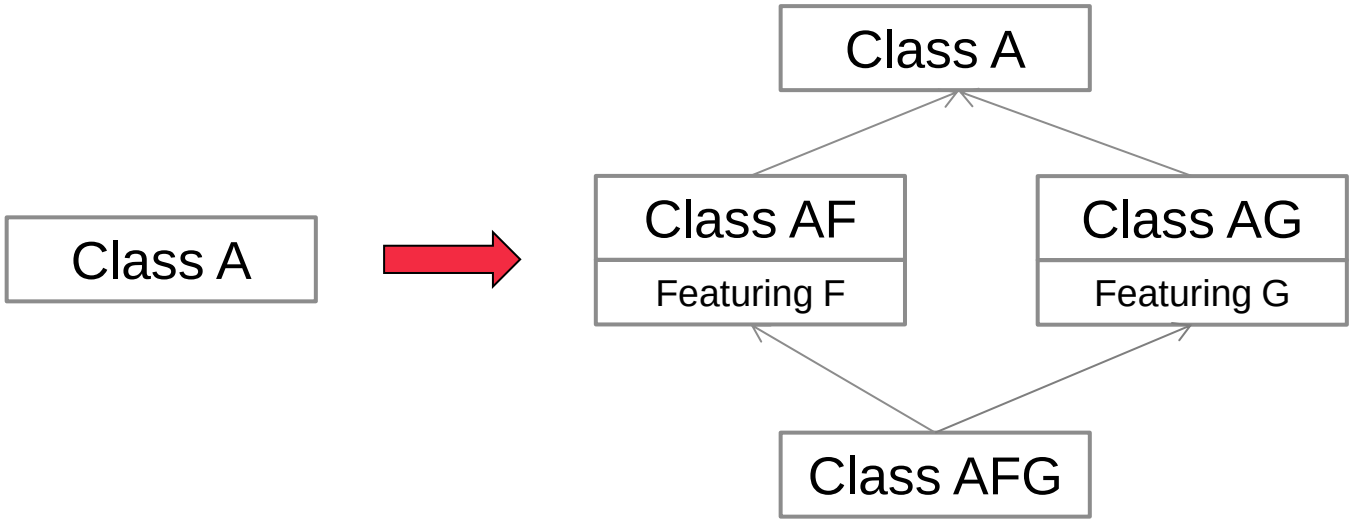
BugSquashing Seminar 26.11.14

Tobias Norajitra

dkfz.
GERMAN
CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

Motivation

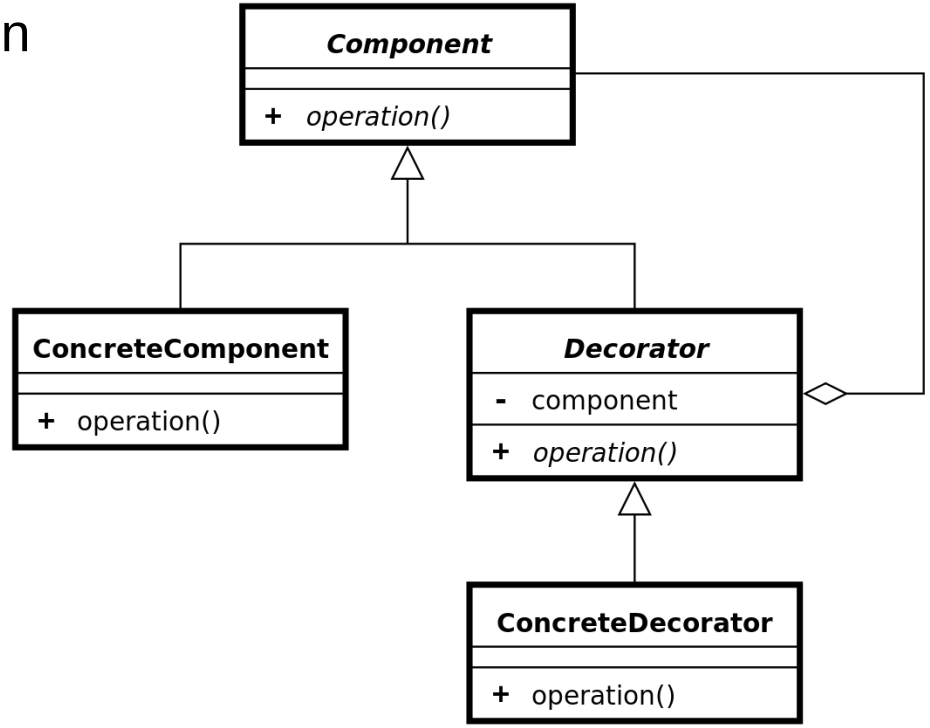
- Add optional extensions F and G to class A:



- Subclassing: complex combinatorial inheritance
- Use decorators instead

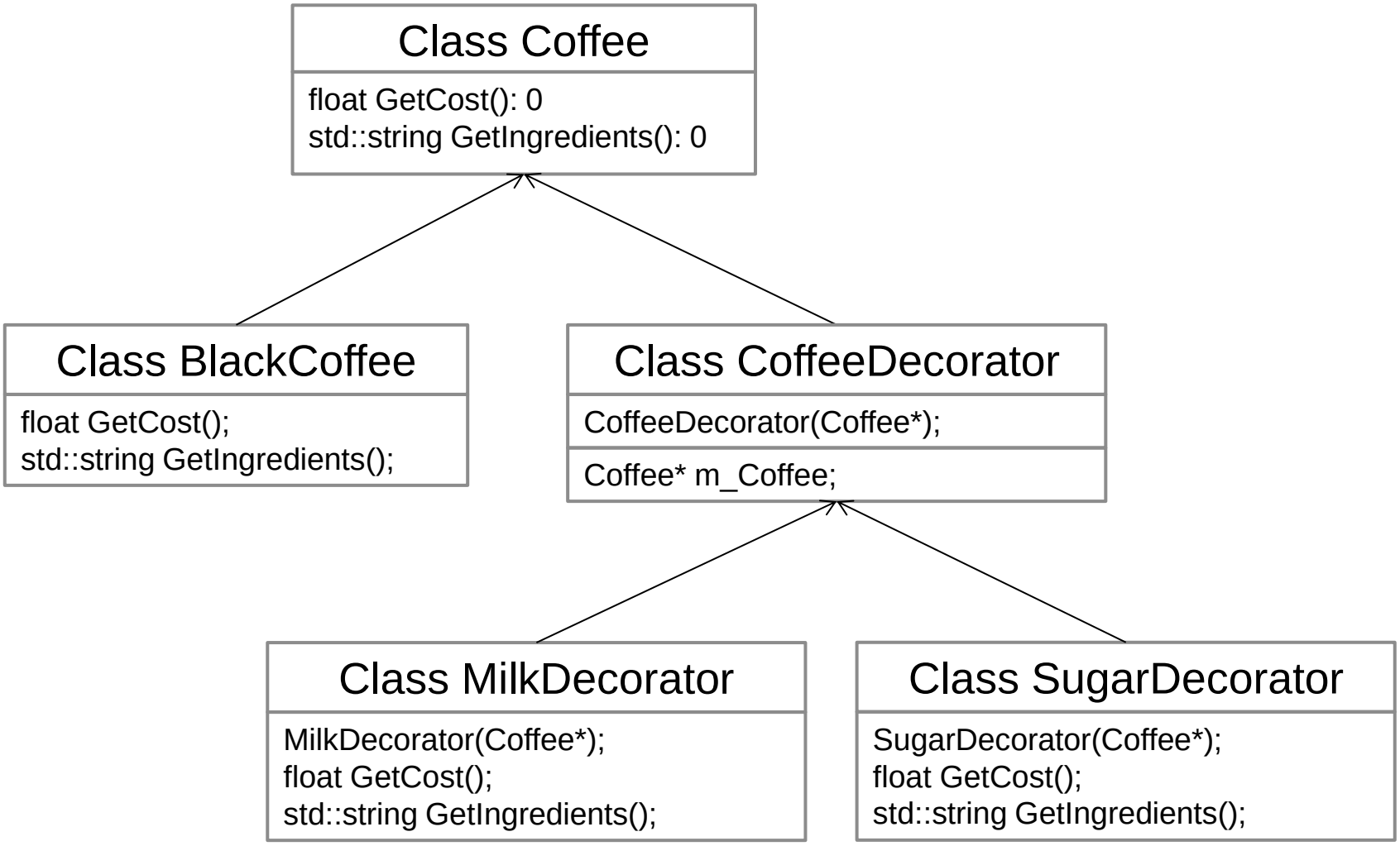
Decorator Pattern

- Decorator pattern:
  - Extend functionality (decorate) on class instance
  - Decorators can be stacked: combine extensions
  - Statically or at runtime



Example I

- Extend Class Coffee by options Milk and Sugar



Example II

```
class BlackCoffee : public Coffee
{
  virtual double getCost() { return 1.0; }
  virtual std::string getIngredients() { return "Coffee"; }
};

class CoffeeDecorator : public Coffee
{
  CoffeeDecorator ( Coffee* basicCoffee )
  : m_BasicCoffee = basicCoffee;

  Coffee* m_BasicCoffee;
}

class MilkDecorator : public CoffeeDecorator
{
  MilkDecorator ( Coffee *basicCoffee ) {};
  virtual float getCost() { return m_BasicCoffee->getCost() + 0.5; }
  virtual std::string getIngredients() { return m_BasicCoffee->getIngredients() + ", " + "Milk"; }
};

class SugarDecorator : public CoffeeDecorator
{
  SugarDecorator (Coffee *basicCoffee) {};
  virtual float getCost() { return m_BasicCoffee->getCost() + 0.3; }
  virtual std::string getIngredients() { return m_BasicCoffee->getIngredients() + ", " + "Sugar"; }
};

int main()
{
  BlackCoffee coffee;
  std::cout << "Cost: " << coffee.getCost() << "; Ingredients: " << coffee.getIngredients() << std::endl;

  MilkDecorator coffeeWithMilk(&coffee);
  std::cout << "Cost: " << coffeeWithMilk.getCost() << "; Ingredients: " << coffeeWithMilk.getIngredients() << std::endl;

  SugarDecorator coffeeWithSugar(&coffee);
  std::cout << "Cost: " << coffeeWithSugar.getCost() << "; Ingredients: " << coffeeWithSugar.getIngredients() << std::endl;

  // Stacked
  MilkDecorator coffeeWithMilkAndSugar(&coffeeWithSugar);
  std::cout << "Cost: " << coffeeWithMilkAndSugar.getCost() << "; Ingredients: " << coffeeWithMilkAndSugar.getIngredients() << std::endl;
}

Cost: 1.0; Ingredients: Coffee
Cost: 1.5; Ingredients: Coffee, Milk
Cost: 1.3; Ingredients: Coffee, Sugar
Cost: 1.8; Ingredients: Coffee, Sugar, Milk
```

Decorators in itk

- Make objects not derived from itk::DataObject available for itk pipelining
  - Decorate with itk::DataObject API
  - Access via Get() and Set()

- Itk::DataObjectDecorator< T >: for subclasses of itk::Object

- SimpleDataObjectDecorator< T >
  - For classes not derived from itk::Object
  - int, float, …

- AutoPointerDataObjectDecorator< T >: Decorate object pointers
  - Pointer Deletion upon decorator destruction

Questions?