Projet POO, Dashboard

Romane Sallio *Pierre Fontaine*

Romane Sallio, Pierre Fontaine
UPPA, Licence, Informatique

1er mai 2017

Sommaire

Introduction

Dashboard est le projet que nous avons conçu en C++ avec le paradigme Objet.
Nous avons utilisé le Framework QT pour développer l'interface utilisateur.
Son objectif est simple : permettre à l'utilisateur d'avoir les éléments essentiels à porté de click.

Pourquoi QT

Paradigme Objet

QT est écrit en C++ et est implémenté selon le paradigme objet. Chacun des composants réfère à une classe particulière qui peut ou non dérivé d'une autre classe mère.

Pourquoi QT

JS LIKE

Lors de l'utilisation d'un nouveau *Framework*, une partie crucial du temps est concacré à l'étude du fonctionnement de celui ci. Il semblait évident qu'après avoir étudier le *JavaScript*, le *QT* qui partage la même philosophie de la gestion d'évènement *(Async/Sync)* serait plus digeste.

Template

Utilité ?

Utilisé dans *List.h*

Pourquoi ?

1. Créer une liste de n'importe quoi
2. Container important

Template

Code


```
template <class T>
class List{...};
```

Forme canonique de coplien

Pourquoi

Il est interressant dans la conception de *container* comme des listes, des sets ... qu'ils puissent s'affecter entre eux, se construire à partir d'un modèle déjà existant.

C'est pourquoi nous intégrons ce modèle dans quelques classes comme :

1. list.h
2. abstractmesureunite.h

Forme canonique de coplien

code 1


```
class AbstractMesureUnite{
protected:
  double _value;
public:
  AbstractMesureUnite();
  AbstractMesureUnite(double);
  AbstractMesureUnite(const AbstractMesureUnite&);
  ~AbstractMesureUnite();
  AbstractMesureUnite &operator=(const AbstractMesureUnite&);
  ...
};
```

Forme canonique de coplien

code 2


```
template <class T>
class List{
protected:
    struct cellule{
        cellule *suivant;
        T valeur;
    };
    typedef cellule* liste;
    liste _l;
public:
    List();
    ~List();
    List(const List<T> &l);
    List<T> operator=(const List<T>);
    ...
};
```

Heritage

QT

Dérivation des composants :

1. Chaque composant du framework QT est une classe
2. Créer un composant spécifique se fait en dérivant un composant général

tips

Il est courant que les classes les plus généralistes ne soient pas instanciables, ce sont des classes *abstraites*.

Heritage

Héritier un composant QT

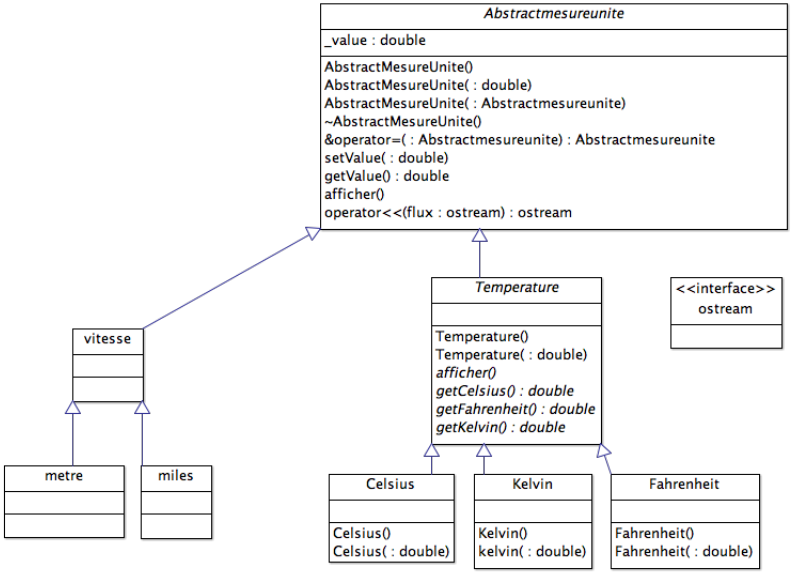

```
#include <QWidget>

class Module : public QWidget{
    Q_OBJECT
protected:
  //méthodes protégées.
public:
  explicit Module(QWidget *parent = 0,double h = 300,
  ↪  double w = 400);
  //méthodes publiques

signals:
  //signals
public slots:
  //slots
};
```

Heritage

En dehors de QT



classe complexe ...

classe abstraite ...