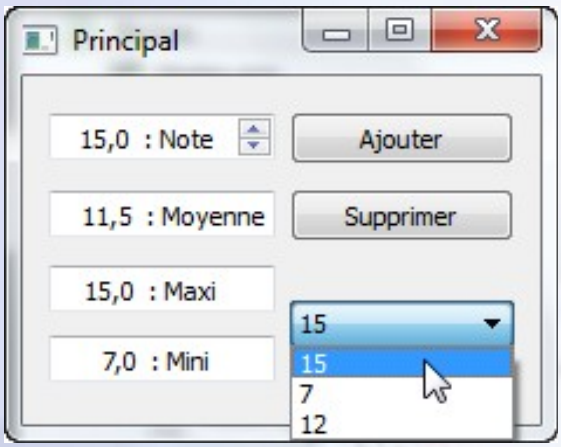
Ce projet permet de visualiser le cours concernant la construction d'un objet au travers de la mise en œuvre d'un projet sur la gestion des notes d'un élève.



x PRINCIPAL.UI

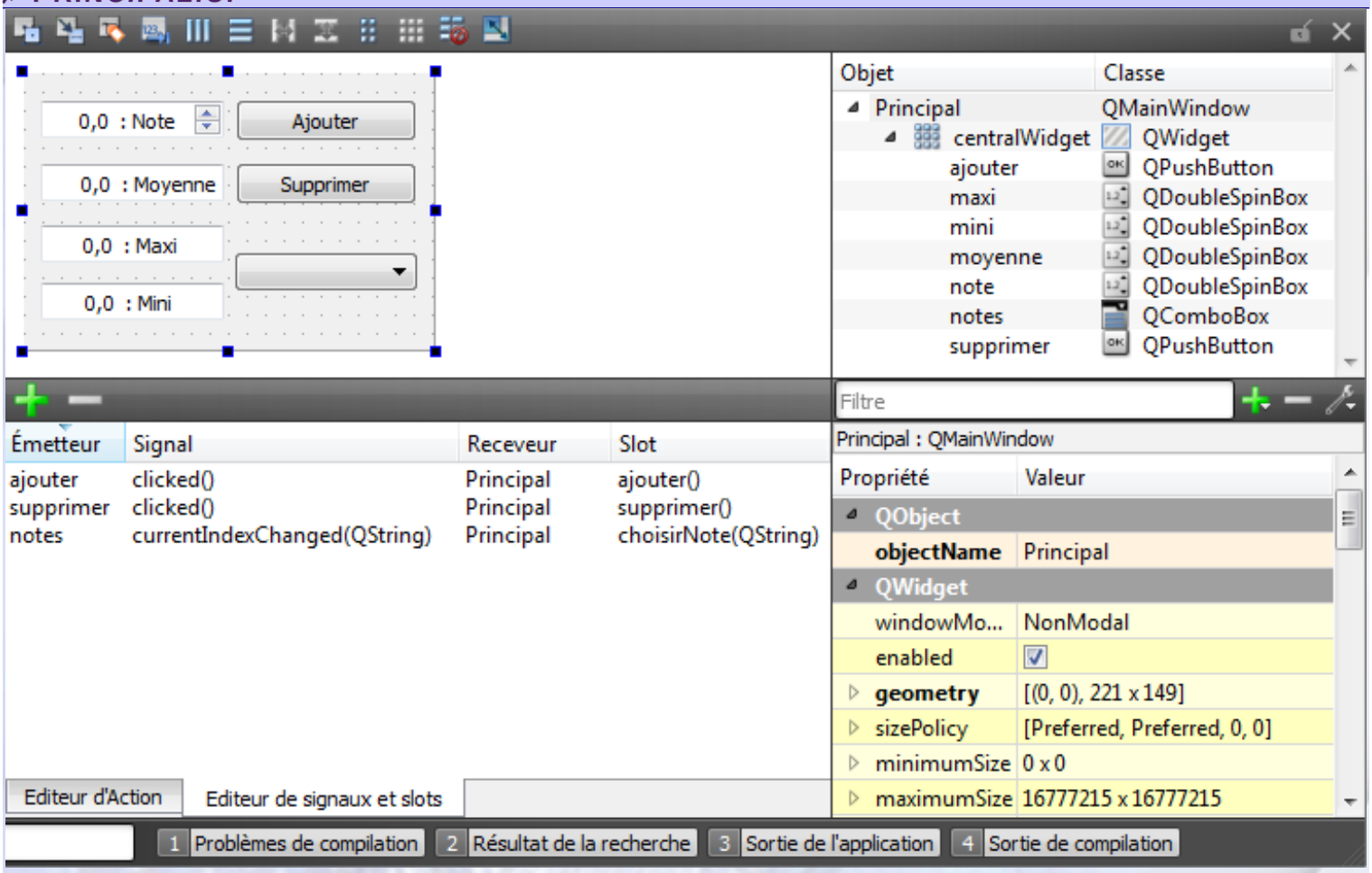


DIAGRAMME DE CLASSES

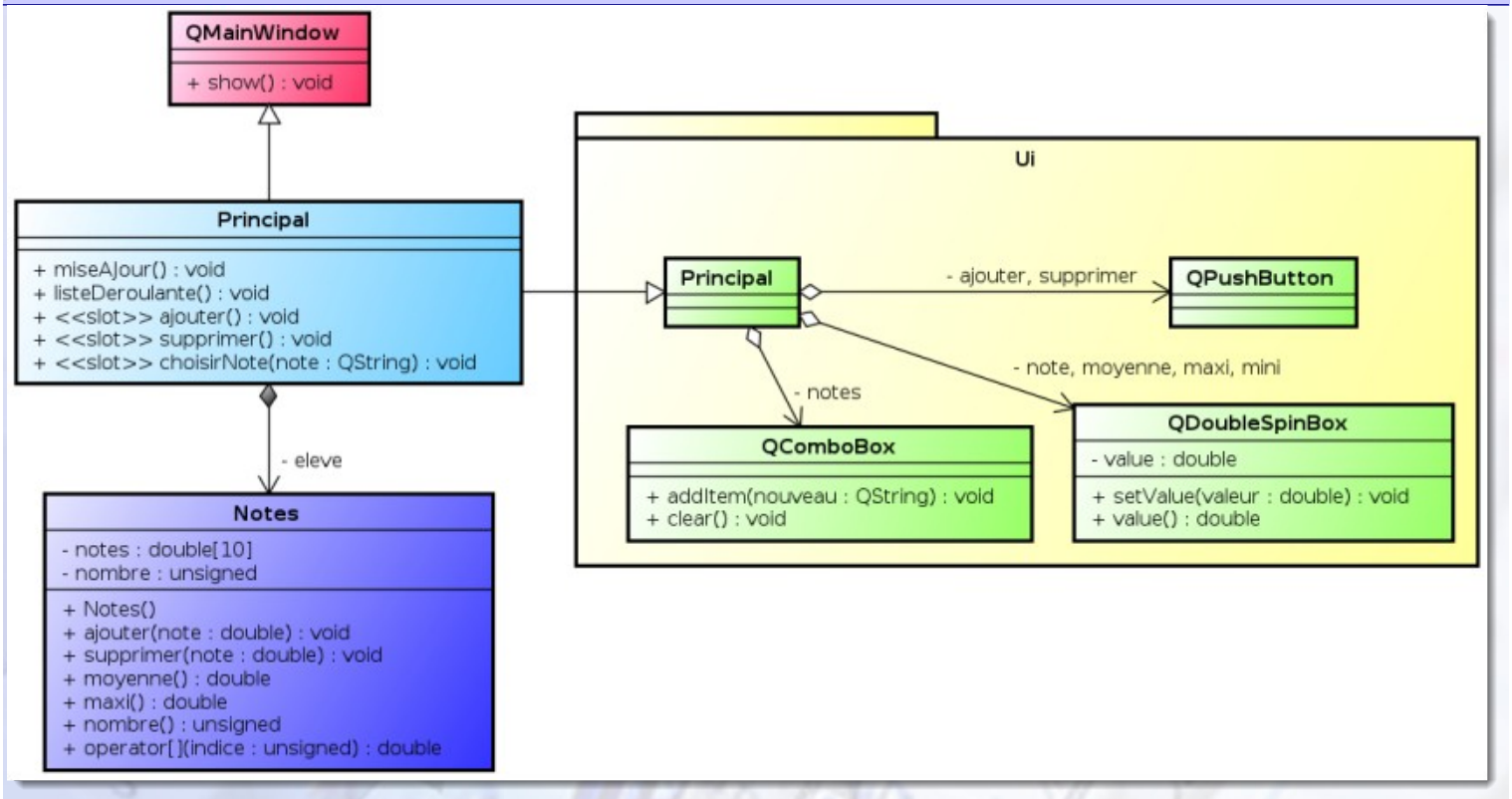
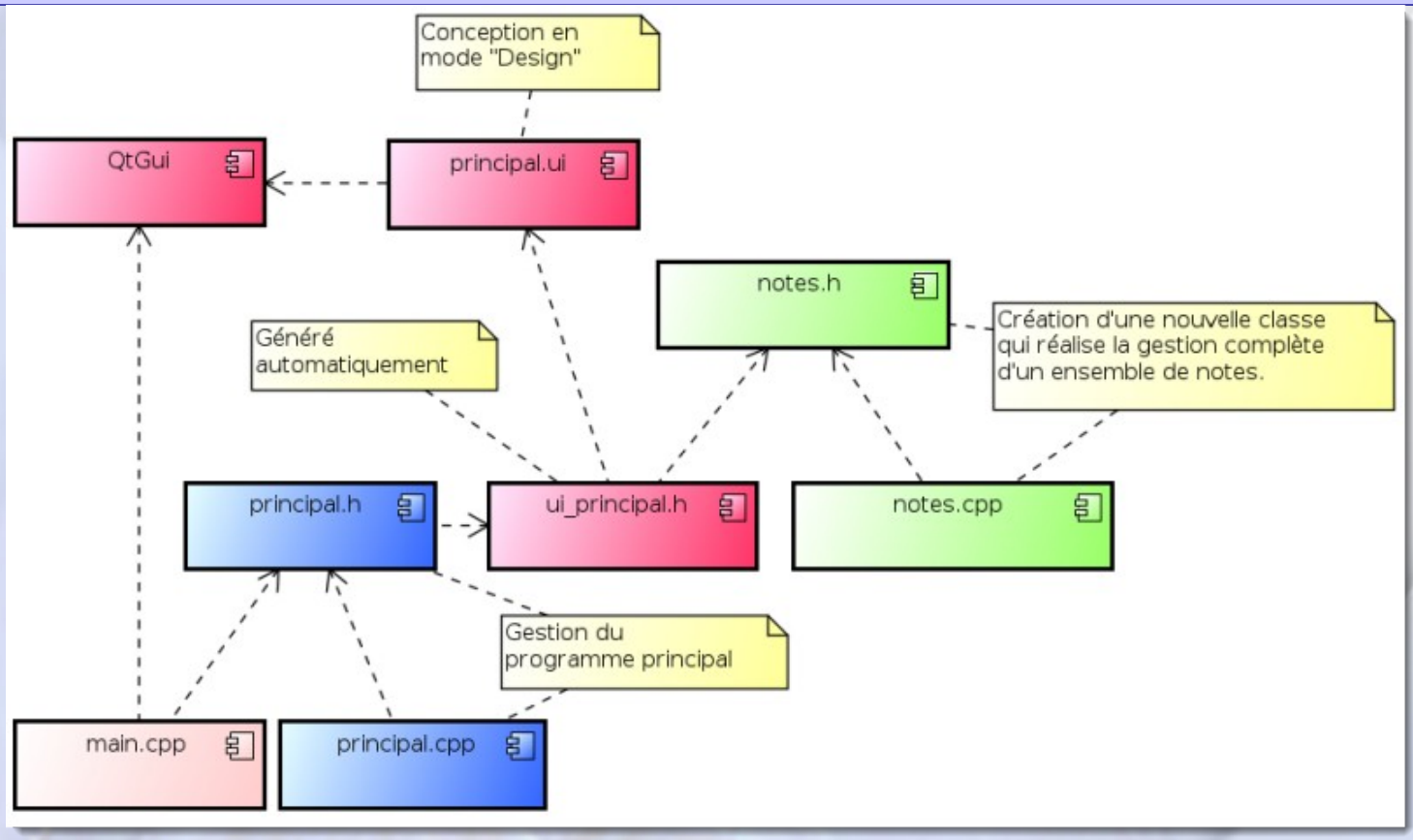


DIAGRAMME DE COMPOSANTS



NOTES.H

```
#ifndef NOTES_H
#define NOTES_H

class Notes
{
   double *notes;
   unsigned nombre, taille;
public:
   Notes(unsigned taille=10);
   ~Notes();
   void ajouter(double note);
   void supprimer(double note);
   double moyenne();
   double mini();
   double maxi();
   double operator[](unsigned indice);
   unsigned getNombre();
};

#endif // NOTES_H
```

NOTES.CPP

```
#include "notes.h"

Notes::Notes(unsigned taille)                { nombre = 0;}
void Notes::ajouter(double note)             { if (nombre<taille) notes[nombre++] = note; }
double Notes::operator [](unsigned indice) { return notes[indice]; }
unsigned Notes::getNombre()                  { return nombre; }

void Notes::supprimer(double note)
{
   bool existe = false;
   unsigned indice = 0;
   while (indice<nombre)
   {
      if (notes[indice]==note) { existe = true; break; }
      indice++;
   }
   if (existe)
   {
      for (unsigned i=indice; i<nombre; i++) notes[i] = notes[i+1];
      nombre--;
   }
}

double Notes::moyenne()
{
   if (nombre==0) return 0.0;
   double somme = notes[0];
   for (unsigned i=1; i<nombre; i++) somme+=notes[i];
   int ajustement = (somme / nombre)*2 + 0.5;
   return  ajustement / 2.0;
}

double Notes::maxi()
{
   if (nombre==0) return 0.0;
   double plusGrande = notes[0];
   for (unsigned i=1; i<nombre; i++) if (notes[i]>plusGrande) plusGrande = notes[i];
   return plusGrande;
}

double Notes::mini()
{
   if (nombre==0) return 0.0;
   double plusPetite = notes[0];
   for (unsigned i=1; i<nombre; i++) if (notes[i]<plusPetite) plusPetite = notes[i];
   return plusPetite;
}
```

PRINCIPAL.H

```
#ifndef PRINCIPAL_H
#define PRINCIPAL_H

#include <QMainWindow>
#include "ui_principal.h"
#include "notes.h"

class Principal : public QMainWindow, public Ui::Principal
{
   Q_OBJECT

public:
   explicit Principal(QWidget *parent = 0);
private:
   Notes eleve;
private:
   void miseAJour();
   void listeDeroulante();
private slots:
   void ajouter();
   void supprimer();
   void choisirNote(QString note);
};

#endif // PRINCIPAL_H
```

PRINCIPAL.CPP

```
#include "principal.h"

Principal::Principal(QWidget *parent) : QmainWindow(parent)  {  setupUi(this);  }

void Principal::ajouter()
{
   eleve.ajouter(note->value());
   miseAJour();
   listeDeroulante();
}

void Principal::supprimer()
{
   eleve.supprimer(note->value());
   miseAJour();
   listeDeroulante();
}

void Principal::miseAJour()
{
   moyenne->setValue(eleve.moyenne());
   maxi->setValue(eleve.maxi());
   mini->setValue(eleve.mini());
}

void Principal::listeDeroulante()
{
   notes->clear();
   for (unsigned i=0; i<eleve.getNombre(); i++)
      notes->addItem(QString::number(eleve[i]));
}

void Principal::choisirNote(QString note) {  this->note->setValue(note.toDouble()); }
```

VARIABLE DYNAMIQUE ET DESTRUCTEUR

Dans la première partie de cette étude, la classe Notes était composée d'un tableau statique, donc figé à une certaine valeur (constante réglée à 10 par défaut). Il est généralement préférable de laisser choisir l'utilisateur quand à la taille qu'il souhaite utiliser.

Dans ce cas là, il est impératif de prévoir un tableau dynamique dont la taille est fixée au moment de la construction. Du coup, nous sommes également obligé d'intégrer un destructeur qui sera utile pour libérer toute la mémoire utilisée par la variable dynamique.

*A chaque **new** doit correspondre un **delete**. Le **new** est prévu pendant la phase de construction. Par symétrie, le **delete** doit être pris en compte au moment de la phase de destruction.*

NOTES.H

```
#ifndef NOTES_H
#define NOTES_H

class Notes
{
   double *notes;
   unsigned nombre, taille;
public:
   Notes(unsigned taille=10);
   ~Notes();
   void ajouter(double note);
   void supprimer(double note);
   double moyenne();
   double mini();
   double maxi();
   double operator[](unsigned indice);
   unsigned getNombre();
};

#endif // NOTES_H
```

NOTES.CPP

```
#include "notes.h"

Notes::Notes(unsigned taille)
{
   nombre = 0;
   notes = new double[this->taille = taille];
}

Notes::~Notes()                                 { delete[] notes; }
double Notes::operator [](unsigned indice) { return notes[indice]; }
unsigned Notes::getNombre()                  { return nombre; }

void Notes::ajouter(double note)              { if (nombre<taille) notes[nombre++] = note; }

void Notes::supprimer(double note)
{
   bool existe = false;
   unsigned indice = 0;
   while (indice<nombre)
   {
      if (notes[indice]==note) { existe = true; break; }
      indice++;
   }
   if (existe)
   {
      for (unsigned i=indice; i<nombre; i++) notes[i] = notes[i+1];
      nombre--;
   }
}

double Notes::moyenne()
{
   if (nombre==0) return 0.0;
   double somme = notes[0];
   for (unsigned i=1; i<nombre; i++) somme+=notes[i];
   int ajustement = (somme / nombre)*2 + 0.5;
   return  ajustement / 2.0;
}

double Notes::maxi()
{
   if (nombre==0) return 0.0;
   double plusGrande = notes[0];
   for (unsigned i=1; i<nombre; i++) if (notes[i]>plusGrande) plusGrande = notes[i];
   return plusGrande;
}
```