

Développement d'applications

version. 1.0 – 24/11/2019

Michel GRIMALDI

<http://grimaldi.univ-tln.fr>
grimaldi@univ-tln.fr

Ce document est sous licence

Creative Commons BY-NC-ND-SA

Attribution : signature de l'auteur initial

Non Commercial : interdiction de tirer un profit commercial de l'œuvre sans autorisation de l'auteur

***No derivative works* : impossibilité d'intégrer tout ou partie dans une œuvre composite**

***Share alike* : partage de l'œuvre, avec obligation de rediffuser selon la même licence**



Organisation des TP

Applications Qt

Exercices sur les
principales
classes de Qt



20 %

Projet personnel

Application Qt de jeu
en réseau avec
interface graphique.



80 %

Projet



Un jeu de Poker en réseau

Règles : <https://www.clubdejeux.com/poker-online/regles>

Hypothèse :

Les deux joueurs sont connectés sur un même réseau local

Les deux joueurs utilisent le même programme.

Scenario :

- 1 - Quand le joueur local lance le programme, il vérifie si l'adversaire est prêt, c'est à dire si un serveur TCP est actif sur sa machine, sur un port prédéfini.
- 2 - S'il l'adversaire n'est pas prêt, il lance lui-même un serveur et attend qu'il vienne se connecter. On l'appellera alors "joueur/serveur".
- 3 - Le joueur/serveur se chargera :
 - d'initialiser la partie
 - de la gestion du jeu de 32 cartes. C'est lui qui distribuera les cartes (5 cartes par joueur), puis le nombre de cartes demandées à la seconde phase du jeu.
 - de la gestion des mises. Il attribuera une somme de départ à chaque joueur.
 - de la comparaison des deux jeux pour définir qui a gagné le pli.
 - de garder un historique horodaté des coups dans un fichier de *log* au format CSV.
 - de la fin de la partie

Ok pour Net Poker !

C'est parti !

Dans quel langage
voulez-vous
que je code ?



Geek de service

Quel langage de programmation ?

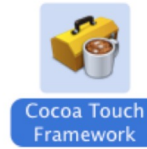


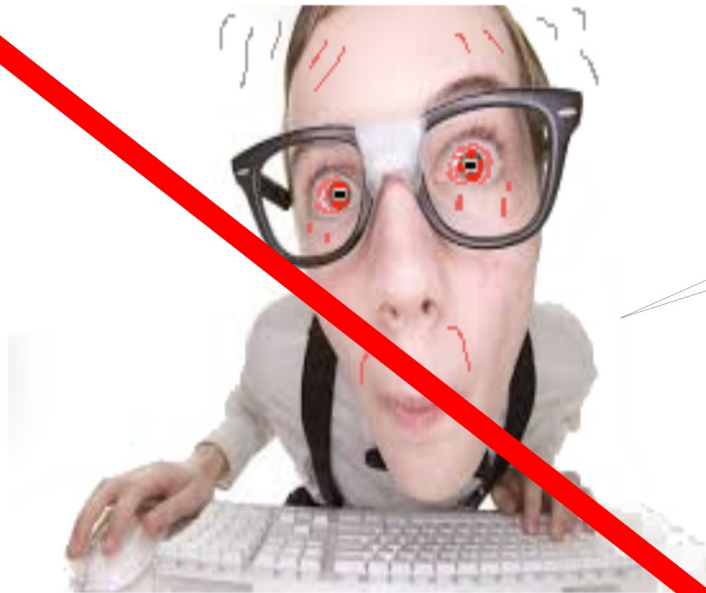
C'est parti pour Net Poker

Sous quel OS ?

Avec quel API
graphique?







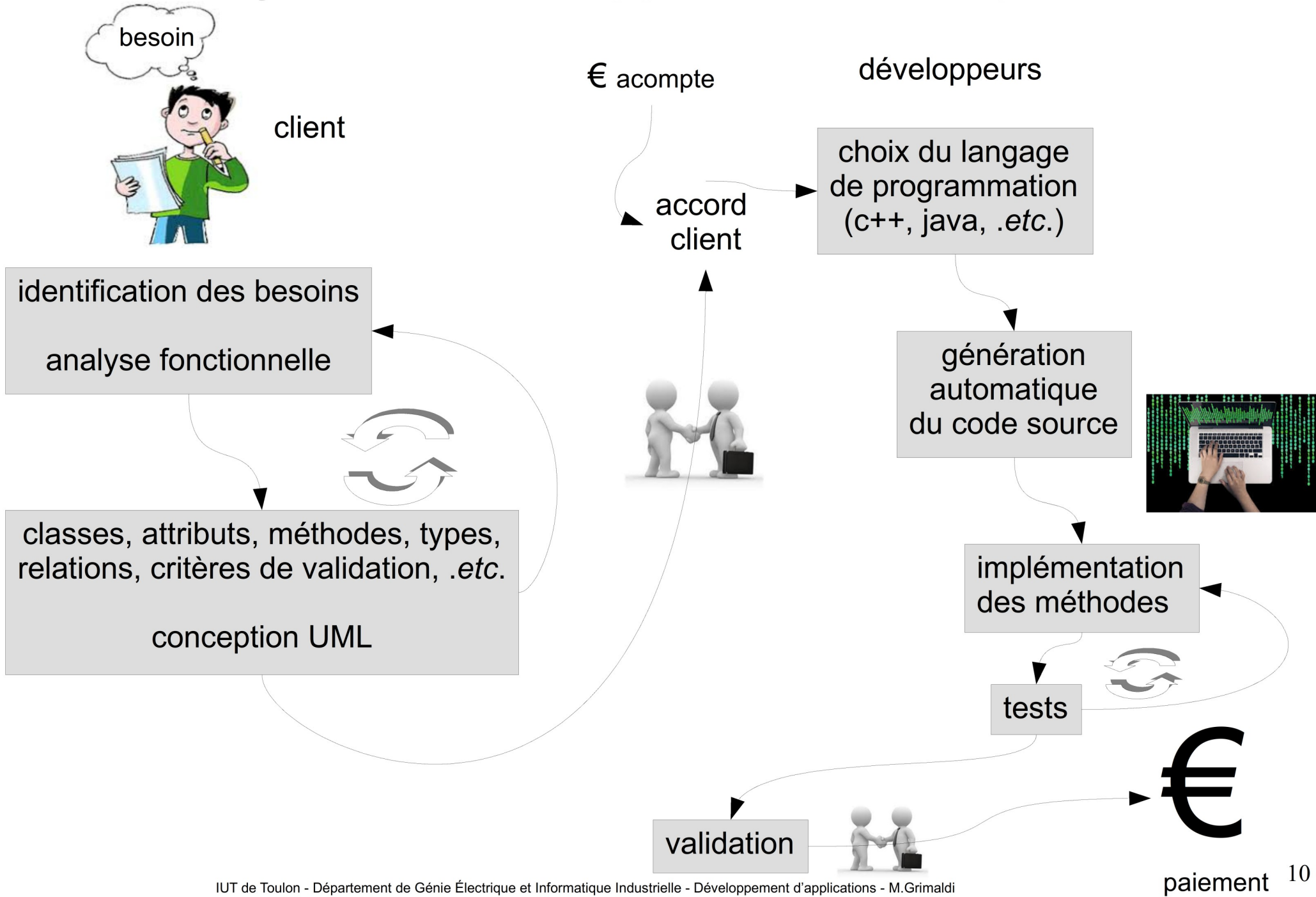
C'est parti...

ça fait trois jours
que je n'ai pas dormi,
rien ne marche,
bououou....

Un mois après ...



Cycle de développement classique



Marche à suivre

- Concepts saillants
- Diagramme des classes primaire
- Génération d'un code intermédiaire, sans réseau ni graphique
- Concepts secondaires, réseau
- Diagramme des classes
- Génération d'un code intermédiaire, sans graphique
- Concepts secondaires, interface
- Génération d'un code intermédiaire
- tests et mise au point

Quels sont les concepts saillants



Joueur



Carte



Jeu



Partie



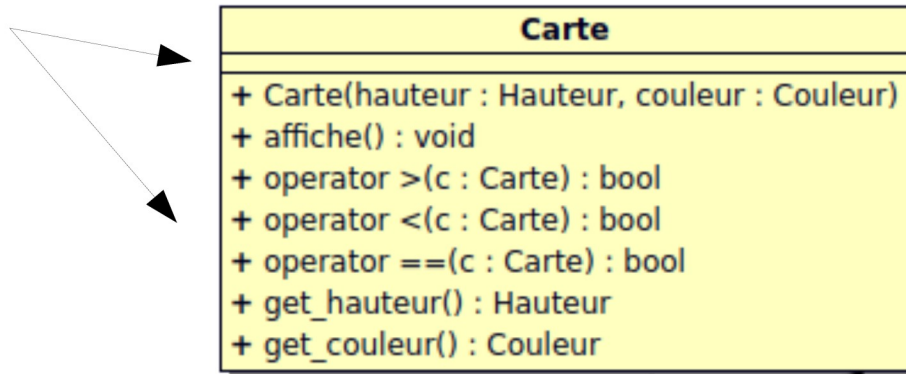
Main

Représentation graphique unifiée, UML

Visibilité :

+ public
- private
protected

Nom de la classe



attributs

méthodes

Analyse conception OO

Découpage du problème
en classes, relations,
objets, .etc.

Génération automatique
du code dans
le langage désiré

Génération de la
documentation

.etc.

BOUML est une suite d'outils UML 2 gratuits
comprenant un modeleur vous permettant de
spécifier et générer du code C++, Java, Idl,
Php, Python et MySQL.

<https://www.bouml.fr>



en UML, le type est spécifié après l'identifiant
x : double ~ x est un double



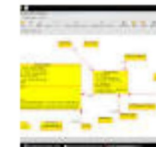
ArgoUML



StarUML



Modelio



Umbrello



Visual Paradigm

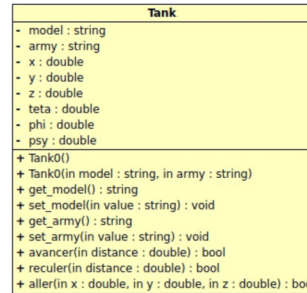
Visual
Paradigm

Unified Modeling Language ~ Indépendance du langage

java

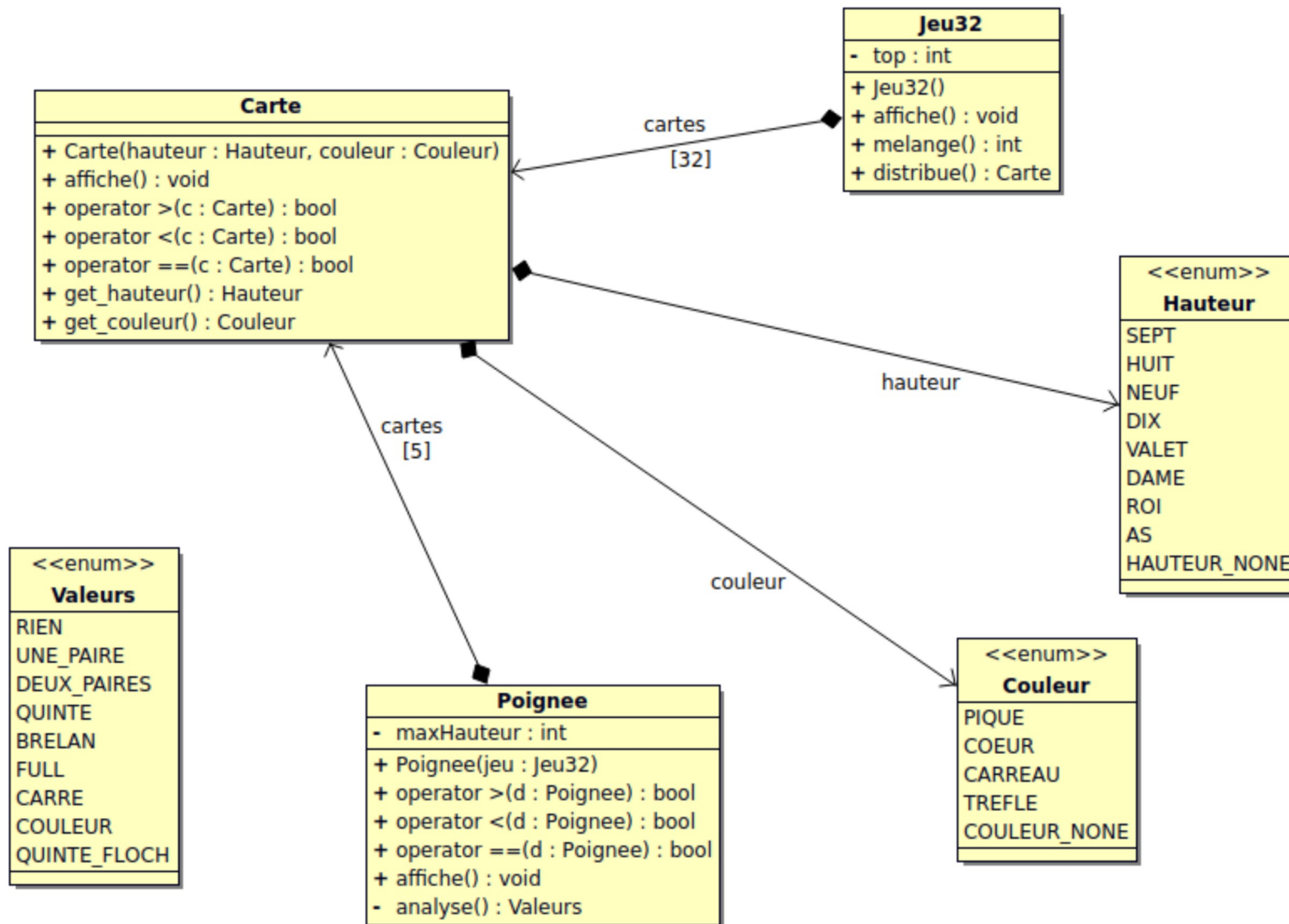
utilisé

c++



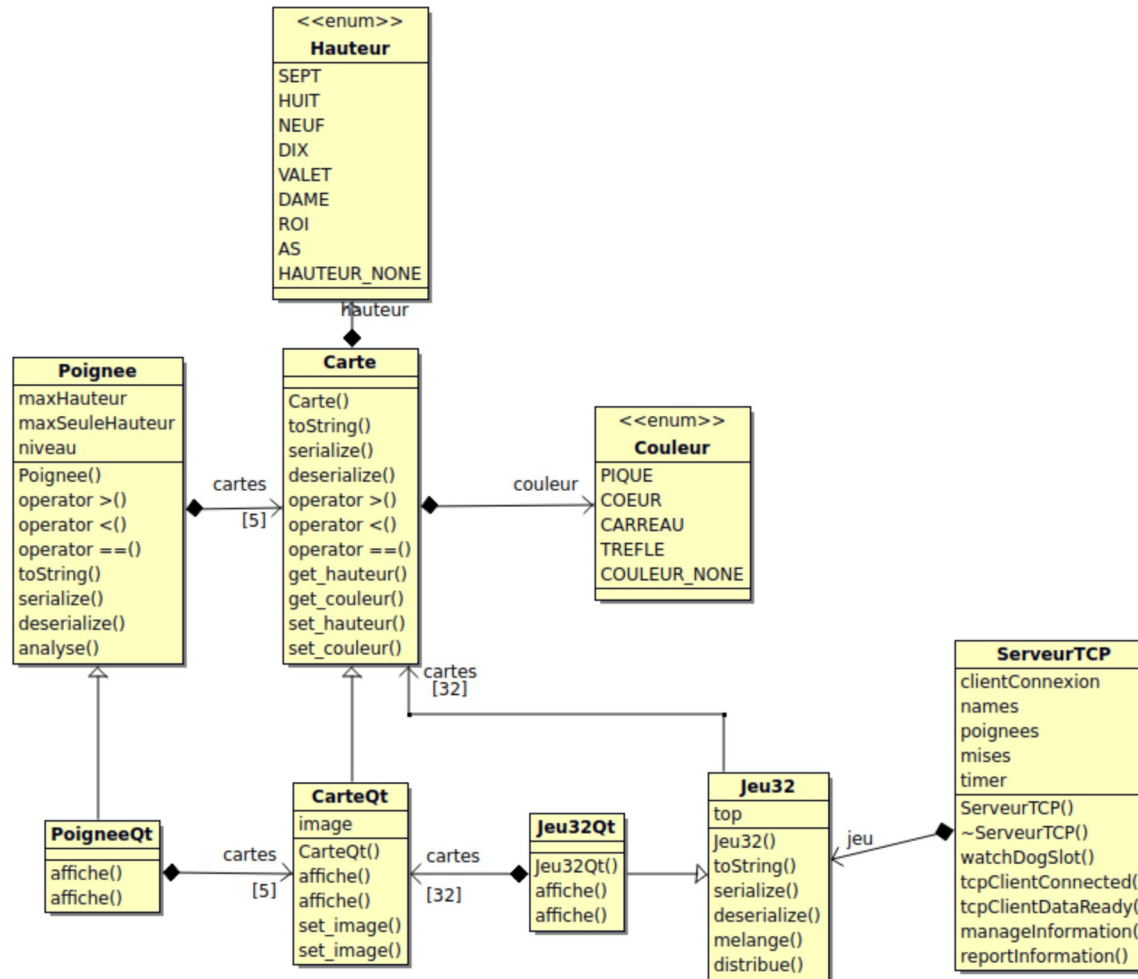
python

Relations entre classes : composition



Règle : **Un seul concept par classe**, une classe pour chaque concept !

Relations entre classes : héritage



Quand on peut dire "est un.e"!

Valeur d'une main au Poker 32

MAINS DE POKER



Quinte flush royale

Rien ne bat la quinte flush royale. C'est la meilleure main au poker et elle est extrêmement rare. Ceci est une quinte flush qui va de 10 à l'As.



Quinte flush

La deuxième meilleure main. Similaire à une suite, mais les cartes sont toutes de la même couleur. Dans cet exemple, toutes des piques qui se suivent.



Carré

Comme son nom l'indique, ce sont quatre cartes du même rang. La main est complétée par la plus haute carte sur la table ou dans votre main.



Full

La combinaison d'un brelan et d'une paire. Si plusieurs joueurs ont un full, c'est celui qui a le plus haut brelan qui gagne.



Couleur

Cinq cartes qui sont toutes de la même couleur. Elles ne doivent pas nécessairement être dans un ordre quelconque. Si plusieurs joueurs ont une couleur, c'est celui qui a la carte la plus haute dans la couleur qui gagne.



Quinte

Une série de cinq cartes qui se suivent, mais qui ne sont pas de la même couleur. Un As peut suivre un Roi, ou être suivi par un 2.



Brelan

Trois cartes du même rang, par exemple trois As. La main est complétée par les deux plus hautes cartes disponibles.



Deux paires

Deux combinaisons de deux cartes du même rang. Par exemple, deux Rois et deux Dames, la main étant complétée par la carte la plus haute qui reste disponible.



Paire

Deux cartes du même rang, par exemple deux As. La main est complétée par les trois cartes les plus élevées qui restent disponibles.



Carte haute

Vous n'avez aucune des combinaisons ci-dessus et seulement une seule carte, l'As dans notre exemple. Il est peut-être temps de vous coucher ?

Le framework Qt

Applications de haut niveau :

- graphiques
- en réseau
- bases de données
- mobiles
- etc.*

Le framework Qt



Qt est un **framework** orienté objet et développé en C++ par Qt Development Frameworks, filiale de Nokia.

Elle offre des composants d'interface graphique (**widgets**), d'accès aux données, de connexions réseaux, de gestion des fils d'exécution, d'analyse XML, etc.

Qt est par certains aspects un framework lorsqu'on l'utilise pour concevoir des **interfaces graphiques** ou que l'on architecture son application en utilisant les mécanismes des **signaux et slots**.

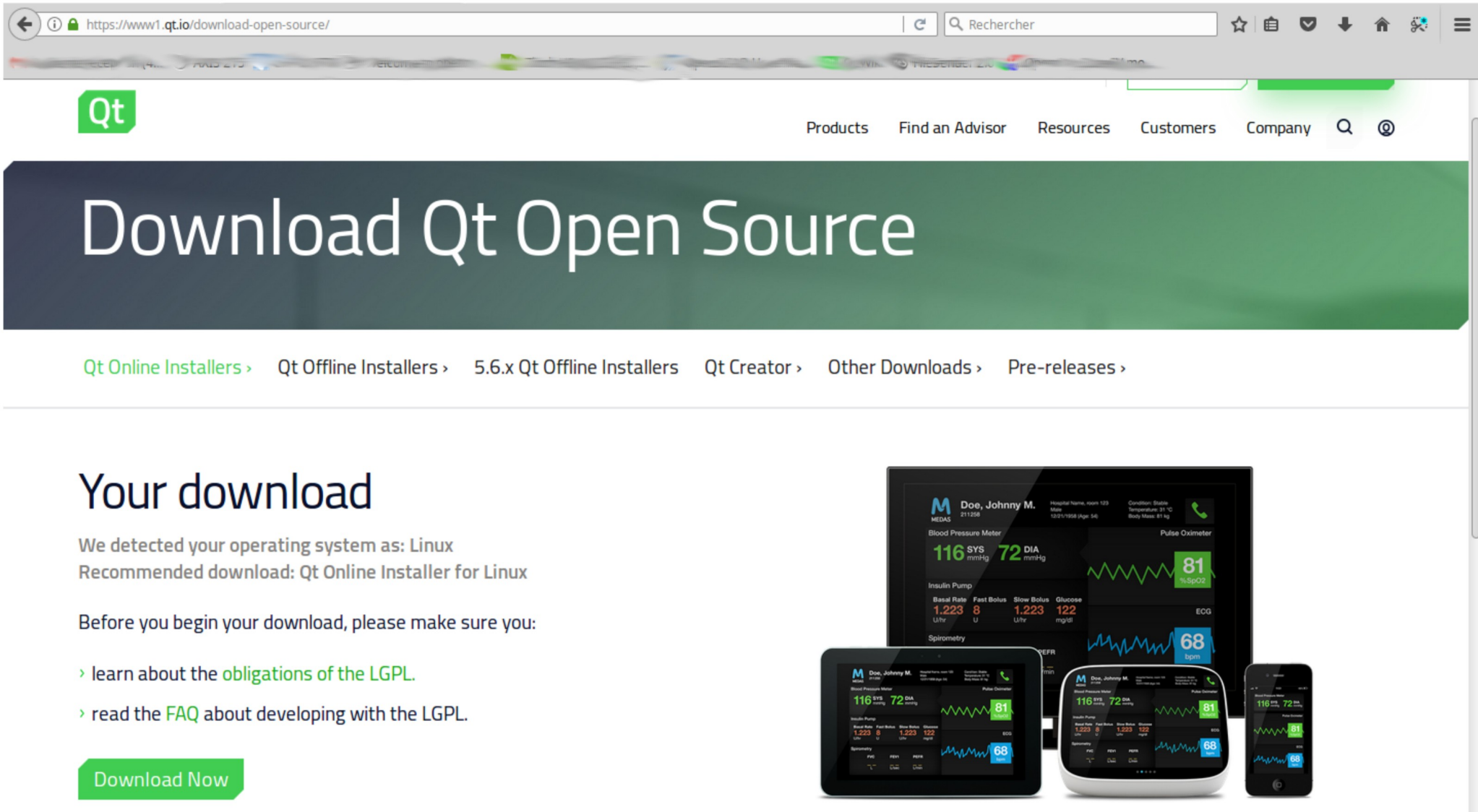
Qt **permet la portabilité des applications** qui n'utilisent que ses composants par simple recompilation du code source. Les environnements supportés sont les **Unix** (dont Linux) qui utilisent le système graphique X Window System, **Windows** et **Mac OS X**.

source : <http://fr.wikipedia.org/wiki/Qt>

Obtenir le SDK de Qt

<https://www1.qt.io/download-open-source/>

<https://www1.qt.io/download-open-source/#section-2>



The screenshot shows the Qt Open Source download page in a web browser. The browser's address bar displays the URL <https://www1.qt.io/download-open-source/>. The page features the Qt logo in the top left corner and a navigation menu with links to Products, Find an Advisor, Resources, Customers, and Company. A large green banner with the text "Download Qt Open Source" is prominently displayed. Below the banner, a horizontal menu lists various download options: Qt Online Installers, Qt Offline Installers, 5.6.x Qt Offline Installers, Qt Creator, Other Downloads, and Pre-releases. The main content area is titled "Your download" and informs the user that the operating system detected is Linux, recommending the Qt Online Installer for Linux. It also provides instructions on what to do before downloading, such as learning about the obligations of the LGPL and reading the FAQ. A green "Download Now" button is located at the bottom left. On the right side, there is a visual representation of a medical monitoring application running on multiple devices (desktop, tablet, and smartphone), showing various health metrics like blood pressure, pulse, and glucose levels.

Qt

Products Find an Advisor Resources Customers Company

Download Qt Open Source

[Qt Online Installers >](#) [Qt Offline Installers >](#) [5.6.x Qt Offline Installers](#) [Qt Creator >](#) [Other Downloads >](#) [Pre-releases >](#)

Your download

We detected your operating system as: Linux
Recommended download: Qt Online Installer for Linux

Before you begin your download, please make sure you:

- › learn about the [obligations of the LGPL](#).
- › read the [FAQ](#) about developing with the LGPL.

[Download Now](#)



The image shows a medical monitoring application running on three devices: a desktop monitor, a tablet, and a smartphone. The application displays various health metrics for a patient named "Doe, Johnny M." (ID: 211256). The metrics include Blood Pressure (116 SYS, 72 DIA mmHg), Pulse Oximeter (81 %SpO2), Insulin Pump (Basal Rate: 1.223 U/hr, Fast Bolus: 8 U, Slow Bolus: 1.223 U/hr, Glucose: 122 mg/dl), and Spirometry (68 bpm). The application also shows a graph of the patient's condition over time.

Environnement de développement : Qt Creator

Qt Creator

Qt Creator est un environnement de développement intégré multiplate-forme faisant partie du framework Qt. Il est donc orienté pour la programmation en C++.

Il intègre directement dans l'interface un **débogueur**, un **outil de création d'interfaces graphiques**, des **outils pour la publication de code sur Git et Mercurial** ainsi que la **documentation Qt**.

L'éditeur de texte intégré permet l'autocomplétion ainsi que la coloration syntaxique.

Qt Creator utilise sous Linux le compilateur gcc et MinGW par défaut sous Windows.

Les classes du framework Qt

Le framework Qt est basé sur une hiérarchie de plusieurs milliers de classes toutes héritées d'une classe QObject, et regroupées par module :

Module	Description
Qt Core	Core non-graphical classes used by other modules.
Qt GUI	Base classes for graphical user interface (GUI) components. Includes OpenGL.
Qt Multimedia	Classes for audio, video, radio and camera functionality.
Qt Multimedia Widgets	Widget-based classes for implementing multimedia functionality.
Qt Network	Classes to make network programming easier and more portable.
Qt QML	Classes for QML and JavaScript languages.
Qt Quick	A declarative framework for building highly dynamic applications with custom user interfaces.
Qt Quick Controls	Reusable Qt Quick based UI controls to create classic desktop-style user interfaces.
Qt Quick Dialogs	Types for creating and interacting with system dialogs from a Qt Quick application.
Qt Quick Layouts	Layouts are items that are used to arrange Qt Quick 2 based items in the user interface.
Qt SQL	Classes for database integration using SQL.
Qt Test	Classes for unit testing Qt applications and libraries.
Qt Widgets	Classes to extend Qt GUI with C++ widgets.

<http://doc.qt.io/qt-5/classes.html>



Quelques classes de base
prises au hasard...

La classe QString

Une chaîne de caractère internationale UNICODE

```
#include "QString.h"

const QString dept = "GEII";
QString message;

message = "Vous avez réussi !";
message = dept;
message = 'W';

QString phrase = "Il était une fois une jolie princesse";
phrase.insert(20, " très");
//phrase contient donc "Il était une fois une très jolie princesse"

phrase.remove(21, 5);
//On revient donc à "Il était une fois une jolie princesse"

phrase.replace(21, 5, "affreuse");
//phrase contient maintenant "Il était une fois une affreuse princesse"

QString suite;
suite = phrase + " qui attendait son prince" ;
//suite contient maintenant "Il était une fois une affreuse princesse qui attendait son prince"
```

Source : <http://sites.univ-provence.fr/wcpp/V2/index.htm>

QString : conversions numériques

Conversion d'une chaîne en nombre
les fonctions toShort(), toInt(), toLong(), toFloat() et toDouble()

```
QString chiffres = "123";  
int n = chiffres.toInt();           // n contient maintenant 123  
chiffres.replace(1, 1, ".");        //chiffres contient maintenant "1.3"  
double k = chiffres.toDouble();     //k contient maintenant 1.3  
n = chiffres.toInt();               //n contient maintenant 0  
  
bool ok ;  
chiffre = "1S23";  
n = chiffres.toInt(&ok);            // on tente la conversion  
  
// ok devient faux si impossible
```

Conversion d'un nombre en chaîne
la fonction setNum(x)

```
QString resultat;  
double pi = 3.14159 ;  
resultat.setNum(pi) ;               //resultat  contient maintenant 3.14159
```

Source : <http://sites.univ-provence.fr/wcpp/V2/index.htm>

La classe QPoint et QPointF

Un point du plan en précision entière (QPoint) ou réelle (QPointF)

QPoint()
QPoint(int xpos, int ypos)
bool isNull() const
int manhattanLength() const
int & rx()
int & ry()
void setX(int x)
void setY(int y)
int x() const
int y() const
QPoint & operator*=(float factor)
QPoint & operator*=(double factor)
QPoint & operator*=(int factor)
QPoint & operator+=(const QPoint & point)
QPoint & operator-=(const QPoint & point)
QPoint & operator/=(qreal divisor)

bool operator!==(const QPoint & p1, const QPoint & p2)
const QPoint operator*(const QPoint & point, float factor)
const QPoint operator*(const QPoint & point, double factor)
const QPoint operator*(const QPoint & point, int factor)
const QPoint operator*(float factor, const QPoint & point)
const QPoint operator*(double factor, const QPoint & point)
const QPoint operator*(int factor, const QPoint & point)
const QPoint operator+(const QPoint & p1, const QPoint & p2)
const QPoint operator+(const QPoint & point)
const QPoint operator-(const QPoint & p1, const QPoint & p2)
const QPoint operator-(const QPoint & point)
const QPoint operator/(const QPoint & point, qreal divisor)
QDataStream & operator<<(QDataStream & stream, const QPoint & point)
bool operator==(const QPoint & p1, const QPoint & p2)
QDataStream & operator>>(QDataStream & stream, QPoint & point)

```
#include <QPoint>
#include <iostream>
using namespace std;
```

```
int main()
```

```
{
    QPoint p1, p2(100, 230);
```

getters

```
    cout<<"p1= ("<<p1.x()<<' , '<<p1.y()<<' ) '<<endl;
    cout<<"p2= ("<<p2.x()<<' , '<<p2.y()<<' ) '<<endl;
```

```
    cout<<"distance de Mahattann de
    p1="<<p2.manhattanLength()<<endl;
```

```
    p1 = p2*10;
    cout<<"p1= ("<<p1.x()<<' , '<<p1.y()<<' ) '<<endl;
```

```
    p1 = p1/10;
    cout<<"p1= ("<<p1.x()<<' , '<<p1.y()<<' ) '<<endl;
```

```
    p1 = p2+QPoint(-20, 50);
    cout<<"p1= ("<<p1.x()<<' , '<<p1.y()<<' ) '<<endl;
```

```
    return 0;
}
```

```
p1=(0,0)
p2=(100,230)
distance de Mahanattann de p1=330
p1=(1000,2300)
p1=(100,230)
p1=(80,280)
```

Les classes QRect et QSize

Un rectangle du plan, une taille (largeur, hauteur)

```
QRect()
QRect(const QPoint & topLeft, const QPoint & bottomRight)
QRect(const QPoint & topLeft, const QSize & size)
QRect(int x, int y, int width, int height)
void adjust(int dx1, int dy1, int dx2, int dy2)
QRect adjusted(int dx1, int dy1, int dx2, int dy2) const
int bottom() const
QPoint bottomLeft() const
QPoint bottomRight() const
QPoint center() const
bool contains(const QPoint & point, bool proper = false) const
bool contains(int x, int y, bool proper) const
bool contains(int x, int y) const
bool contains(const QRect & rectangle, bool proper = false) const
void getCoords(int * x1, int * y1, int * x2, int * y2) const
void getRect(int * x, int * y, int * width, int * height) const
int height() const
QRect intersected(const QRect & rectangle) const
bool intersects(const QRect & rectangle) const
bool isEmpty() const
bool.isNull() const
bool.isValid() const
int left() const
void moveBottom(int y)
void moveBottomLeft(const QPoint & position)
void moveBottomRight(const QPoint & position)
void moveCenter(const QPoint & position)
void moveLeft(int x)
void moveRight(int x)
void moveTo(int x, int y)
void moveTo(const QPoint & position)
void moveTop(int y)
void moveTopLeft(const QPoint & position)
void moveTopRight(const QPoint & position)
QRect normalized() const
int right() const
void setBottom(int y)
void setBottomLeft(const QPoint & position)
```

```
#include <QPoint>
#include <QRect>
#include <iostream>
using namespace std;
// affichage console de données d'un QRect et d'un QPoint
void affiche_rect(string titre, QRect r){
    cout<<titre<<": ("<<r.x()<<', '<<r.y()<<","largeur="<<r.width()
        <<","hauteur="<<r.height()<<') '<<endl;
}
void affiche_point(string titre, QPoint p){
    cout<<titre<<": ("<<p.x()<<', '<<p.y()<<') '<<endl;
}

int main()
{
    QPoint p1(100, 230), p2(200, 300), p3, p4;
    QRect r1, r2(p1, p2), r3(p1, QSize(20, 30));

    affiche_rect("R1", r1);
    affiche_rect("R2", r2);
    affiche_rect("R3", r3);

    p3=r3.bottomRight(); // point en bas à droite
    affiche_point("P3", p3);
    p4 = r3.bottomLeft(); // point en bas à gauche
    affiche_point("P4", p4);

    r3.translate(20, -20); // translation du rectangle
    affiche_rect("R3", r3);

    if (r2.intersects(r3)){
        r1=r2.intersected(r3); // intersection de rectangles
        affiche_rect("R1", r1);
    }
    else cout<<"pas d'intersection R2, R3"<<endl;

    if (r1.contains(p3))cout<<"P3 dans R1"; else cout<<"P3 n'est pas dans R1";
    cout<<endl;
    return 0;
}
```

```
R1: (0,0,largeur=0,hauteur=0)
R2: (100,230,largeur=101,hauteur=71)
R3: (100,230,largeur=20,hauteur=30)
P3=(119,259)
P4=(100,259)
R3: (120,210,largeur=20,hauteur=30)
R1: (120,230,largeur=20,hauteur=10)
P3 n'est pas dans R1
```

La classe QDate

Une date sous toutes ses formes

QDate()

QDate(int y, int m, int d)

QDate addDays(qint64 ndays)

QDate addMonths(int nmonths)

QDate addYears(int nyears)

int day() const

int dayOfWeek() const

int dayOfYear() const

int daysInMonth() const

int daysInYear() const

qint64 daysTo(const QDate &date)

void getDate(int * year, int * month, int * day)

bool isNull() const

bool isValid() const

int month() const

bool setDate(int year, int month, int day)

qint64 toJulianDay() const

QString toString(const QString &format)

QString toString(Qt::DateFormat format)

int weekNumber(int * year, int * month, int * day)

int year() const

bool operator!=(const QDate &date) const

bool operator<(const QDate &date) const

bool operator<=(const QDate &date) const

bool operator==(const QDate &date) const

bool operator>(const QDate &date) const

bool operator>=(const QDate &date) const

date ? : 31/8/2021

d1: 15.08.2019, 227 ème jour de l'année 2019

d2: 31.08.2021, mardi, dans 747 jours/aujourd'hui'

d3: 01.08.2021 dimanche, 717 jours/aujourd'hui, dimanche, 2021 a 365 jours

d2 (

QDate::currentDate()

toString("dd.MM.yyyy")

d1.dayOfYear()

d2.toString

QDate::longDayName d2.dayOfWeek

d1.daysTo

d2.addDays

d3.year

d3.daysInYear

La classe QTime

L'heure sous toutes ses formes

	QTime()
	QTime (int <i>h</i> , int <i>m</i> , int <i>s</i> = 0, int <i>ms</i> = 0)
QTime	addMSecs (int <i>ms</i>) const
QTime	addSecs (int <i>s</i>) const
int	elapsed () const
int	hour () const
bool	isNull () const
bool	isValid () const
int	minute () const
int	msec () const
int	msecsSinceStartOfDay () const
int	msecsTo (const QTime & <i>t</i>) const
int	restart ()
int	second () const
int	secsTo (const QTime & <i>t</i>) const
bool	setHMS (int <i>h</i> , int <i>m</i> , int <i>s</i> , int <i>ms</i> = 0)
void	start ()
QString	toString (const QString & <i>format</i>) const
QString	toString (Qt::DateFormat <i>format</i> = Qt::TextDate)
bool	operator!= (const QTime & <i>t</i>) const
bool	operator< (const QTime & <i>t</i>) const
bool	operator<= (const QTime & <i>t</i>) const
bool	operator== (const QTime & <i>t</i>) const
bool	operator> (const QTime & <i>t</i>) const
bool	operator>= (const QTime & <i>t</i>) const

```
t1: 09:39:27
t2: 15:12:03
t1 est avant t2
il y a 19956 secondes d'écart
```

```
void affiche_time(string titre, QTime t){
cout<<titre<<" : "<<t.toString().toUtf8().constData()<<endl;
}
```

t2 (

QTime::currentTime

t1>

t1==

t1.secsTo (

La classe QTime

L'heure sous toutes ses formes

QTime()

QTime(int *h*, int *m*, int *s* = 0, int *ms* = 0)

QTime **addMsecs**(int *ms*) const

QTime **addSecs**(int *s*) const

int **elapsed**() const

int **hour**() const

bool **isNull**() const

bool **isValid**() const

int **minute**() const

int **msec**() const

int **msecsSinceStartOfDay**() const

int **msecsTo**(const **QTime** & *t*) const

int **restart**()

int **second**() const

int **secsTo**(const **QTime** & *t*) const

bool **setHMS**(int *h*, int *m*, int *s*, int *ms* = 0)

void **start**()

QString **toString**(const **QString** & *format*) const

QString **toString**(Qt::DateFormat *format* = Qt::Date)

bool **operator!=**(const **QTime** & *t*) const

bool **operator<**(const **QTime** & *t*) const

bool **operator<=**(const **QTime** & *t*) const

bool **operator==**(const **QTime** & *t*) const

bool **operator>**(const **QTime** & *t*) const

bool **operator>=**(const **QTime** & *t*) const

les boucles seules ont pris 24 ms
la fonction rand toute seule prend 8.6 ns

```
void affiche_time(string titre, QTime t){  
    cout<<titre<<": "<<t.toString().toUtf8().constData()<<endl;  
}
```

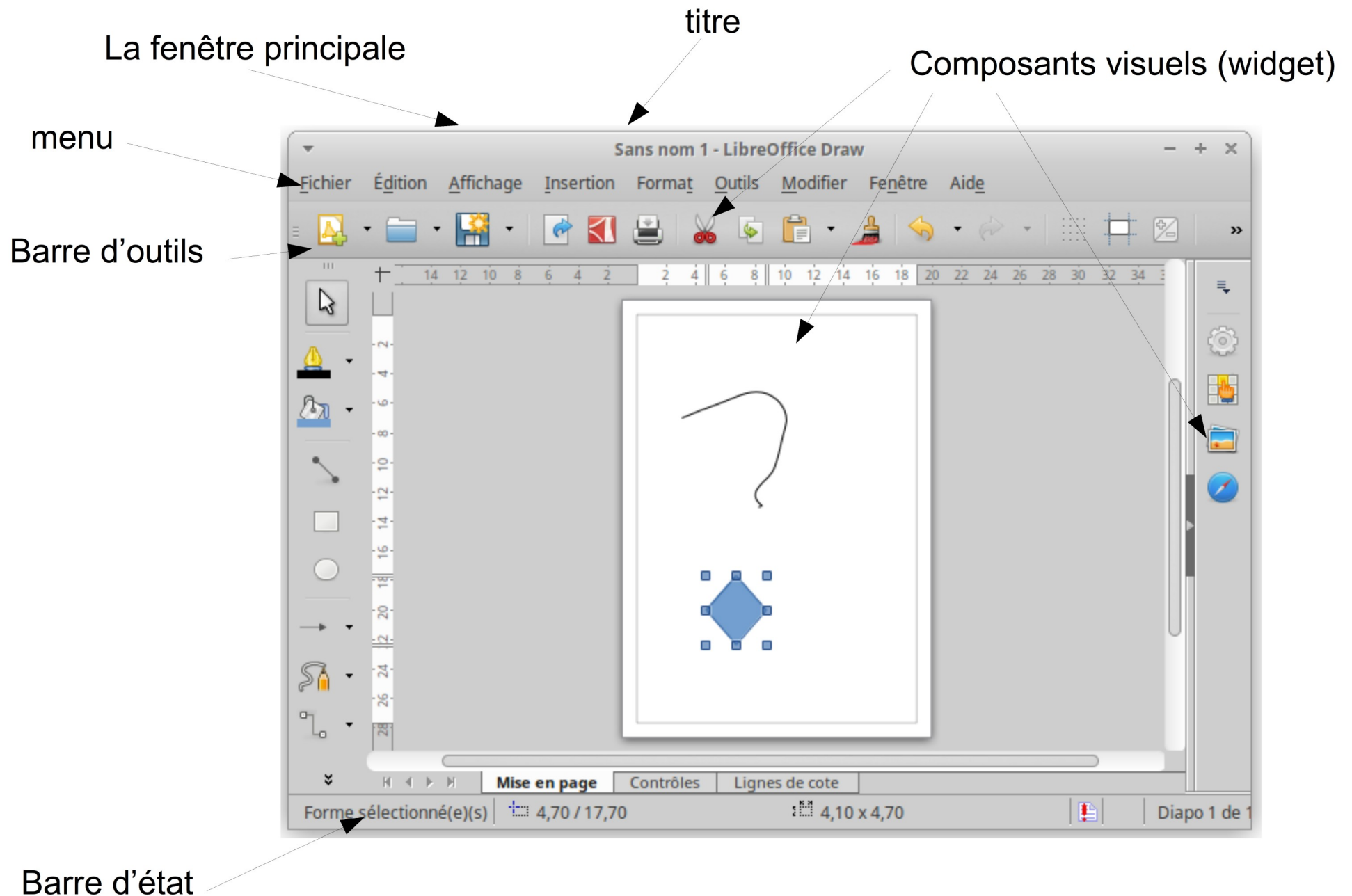
t3.start()

t3.restart

t3.elapsed

Application graphique Qt

Application graphique : exemple



Création d'une application graphique Qt

Application graphique Qt

Information sur la classe

Informations de base des classes pour lesquelles vous créez des fichiers squelettes de code source.

ApplicationDemo
QWidget

applicationdemo.h
applicationdemo.cpp
applicationdemo.ui

Interface graphique: ☒

< Précédent Suivant > Annuler

```
#ifndef APPLICATIONDEMO_H
#define APPLICATIONDEMO_H

#include <QWidget>

namespace Ui {
class ApplicationDemo;
}

class ApplicationDemo : public QWidget
{
    Q_OBJECT

public:
    explicit ApplicationDemo(QWidget *parent = 0);
    ~ApplicationDemo();

private:
    Ui::ApplicationDemo *ui;
};

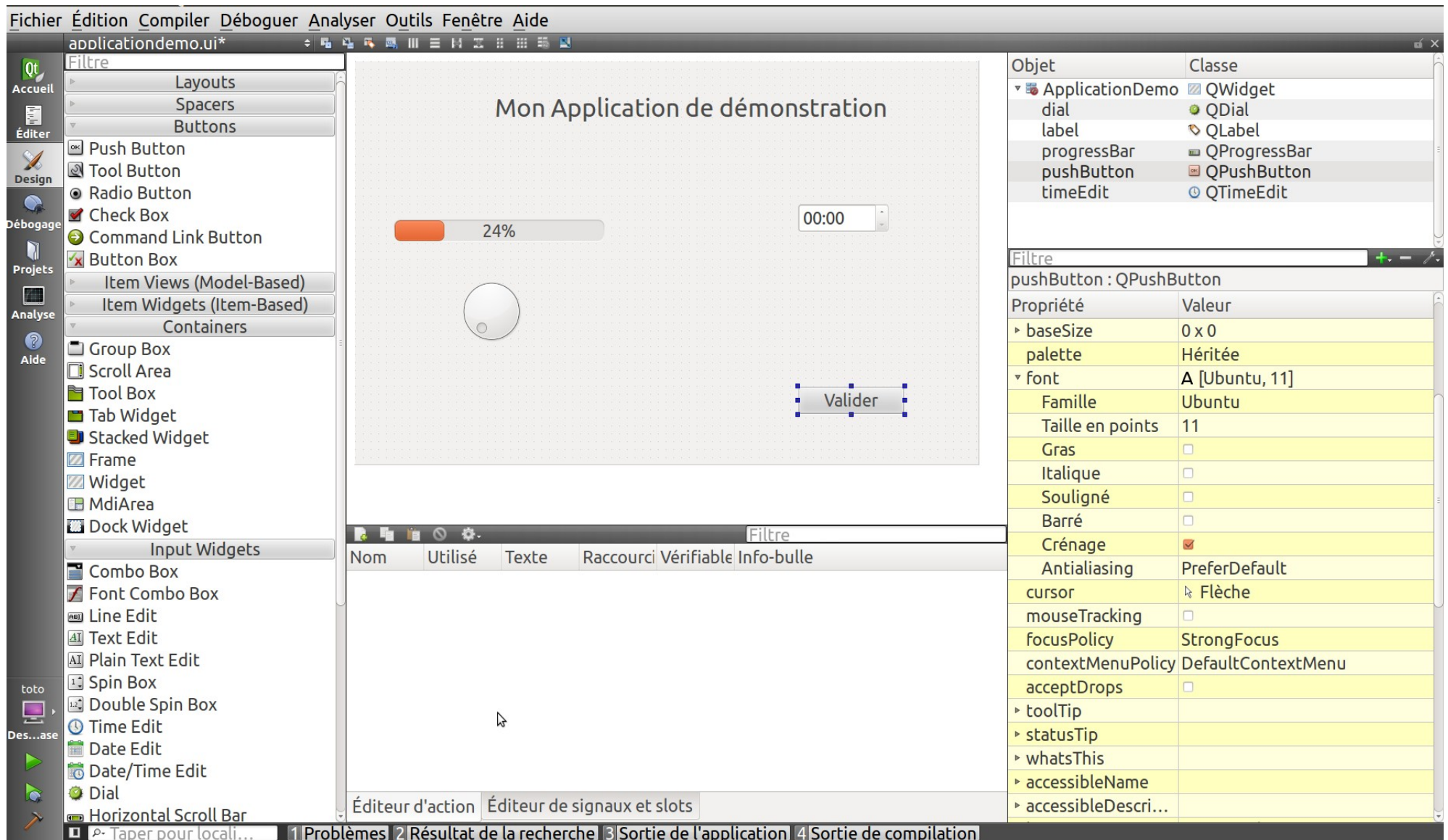
#endif // APPLICATIONDEMO_H
```

```
#include "applicationdemo.h"
#include "ui_applicationdemo.h"

ApplicationDemo::ApplicationDemo(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::ApplicationDemo)
{
    ui->setupUi(this);
}

ApplicationDemo::~ApplicationDemo()
{
    delete ui;
}
```


L'interface utilisateur (*user interface*)

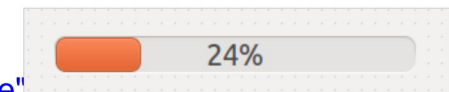


Qt Designer : dessin de l'interface (fichier : applicationdemo.ui)

Le fichier de design de l'UI

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>ApplicationDemo</class>
  <widget class="QWidget" name="ApplicationDemo">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>803</width>
        <height>509</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>ApplicationDemo</string>
    </property>
    <widget class="QLabel" name="label">
      <property name="geometry">
        <rect>
          <x>180</x>
          <y>40</y>
          <width>505</width>
          <height>37</height>
        </rect>
      </property>
      <property name="font">
        <font>
          <pointsize>16</pointsize>
        </font>
      </property>
      <property name="text">
        <string>Mon Application de démonstration</string>
      </property>
    </widget>
  </widget>
</ui>
```

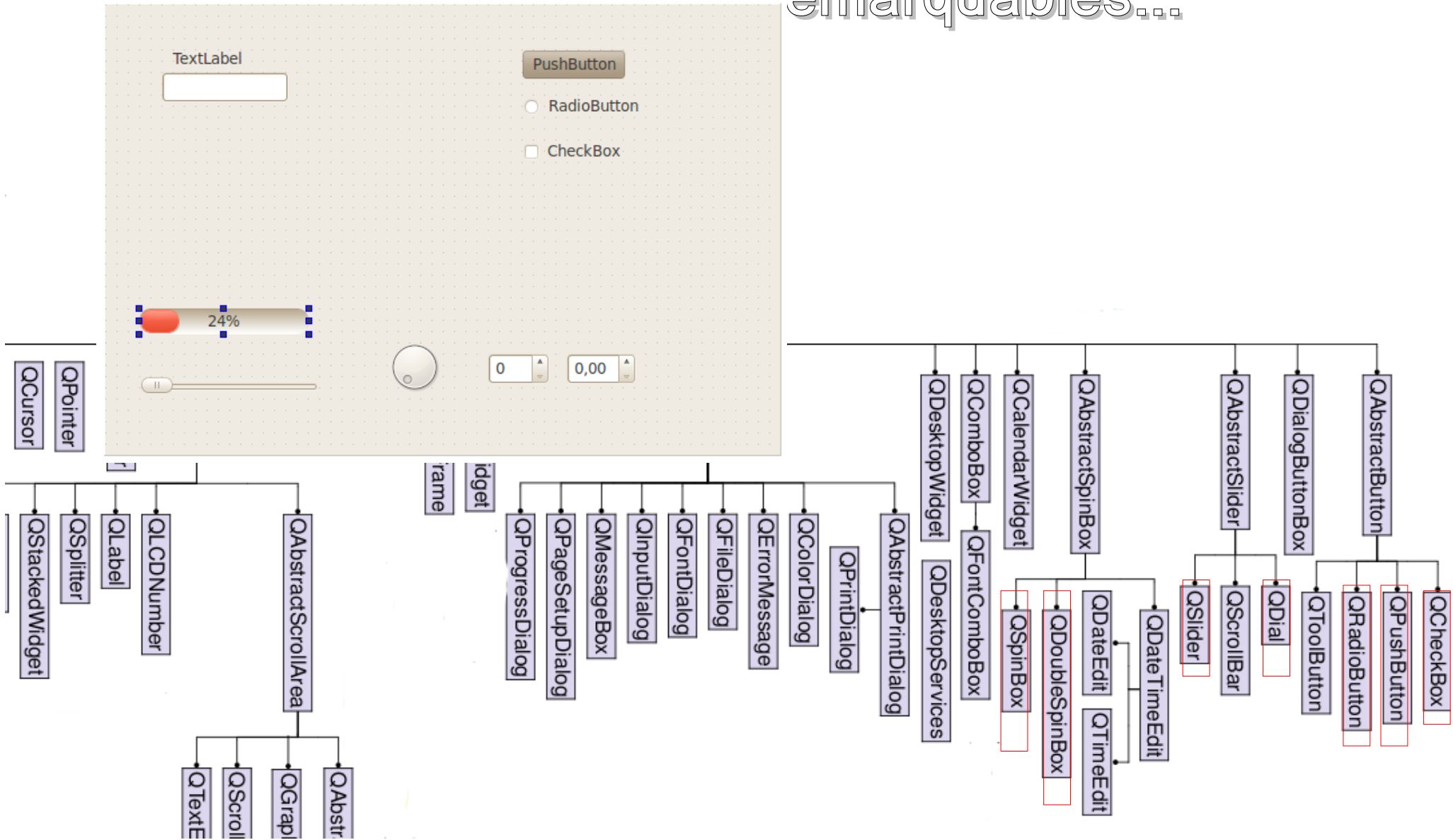
```
<?xml version="1.0" encoding="UTF-8"?>
  </property>
</widget>
  <widget class="QDial" name="dial">
    <property name="geometry">
      <rect>
        <x>130</x>
        <y>270</y>
        <width>91</width>
        <height>91</height>
      </rect>
    </property>
  </widget>
  <widget class="QProgressBar" name="progressBar">
    <property name="geometry">
      <rect>
        <x>50</x>
        <y>200</y>
        <width>271</width>
        <height>28</height>
      </rect>
    </property>
    <property name="value">
      <number>24</number>
    </property>
  </widget>
  <widget class="QTimeEdit" name="timeEdit">
    <property name="geometry">
      <rect . . . . .
    </property>
  </widget>
```



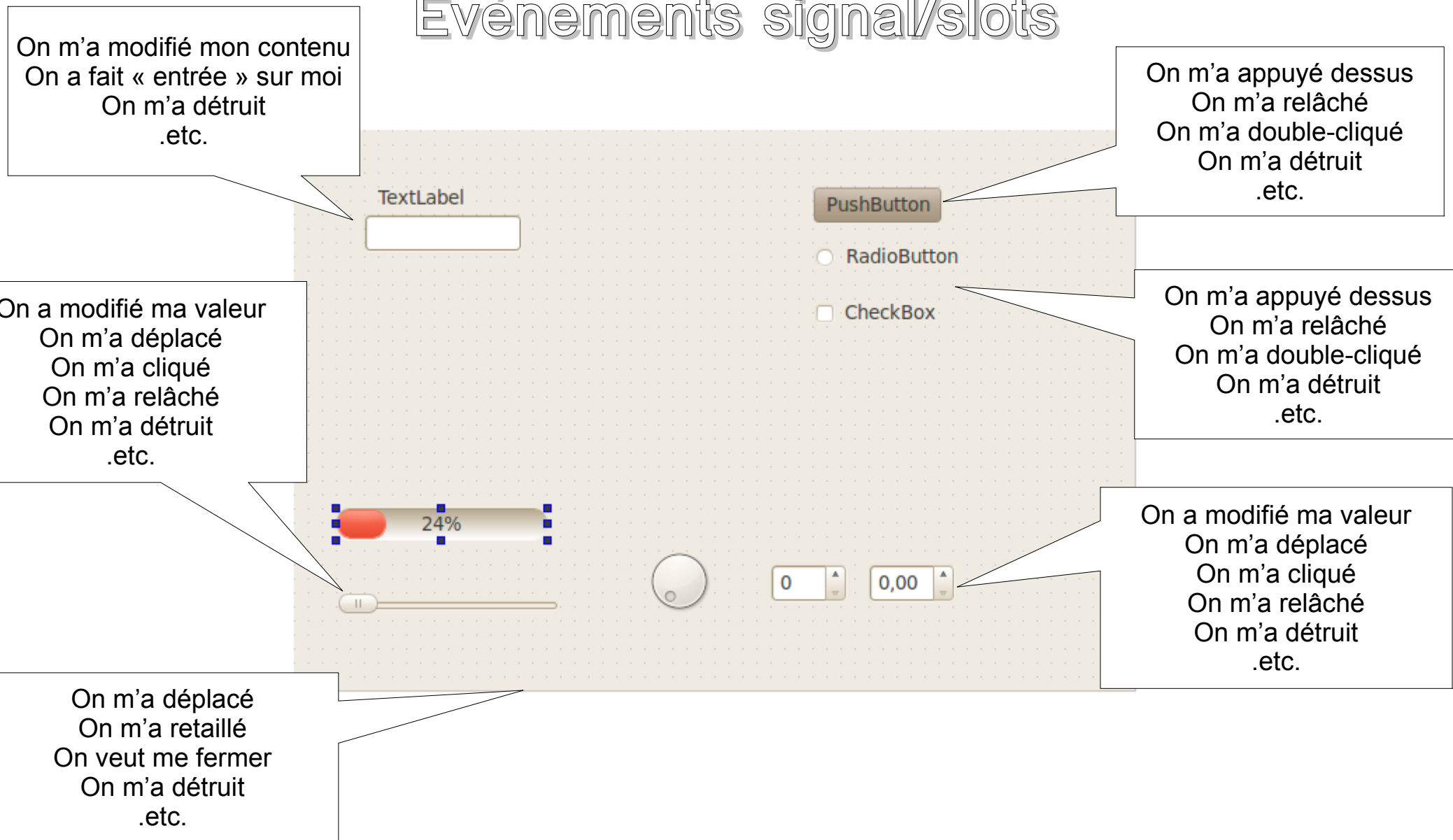
.....

Mon Application de démonstration

Quelques classes remarquables...



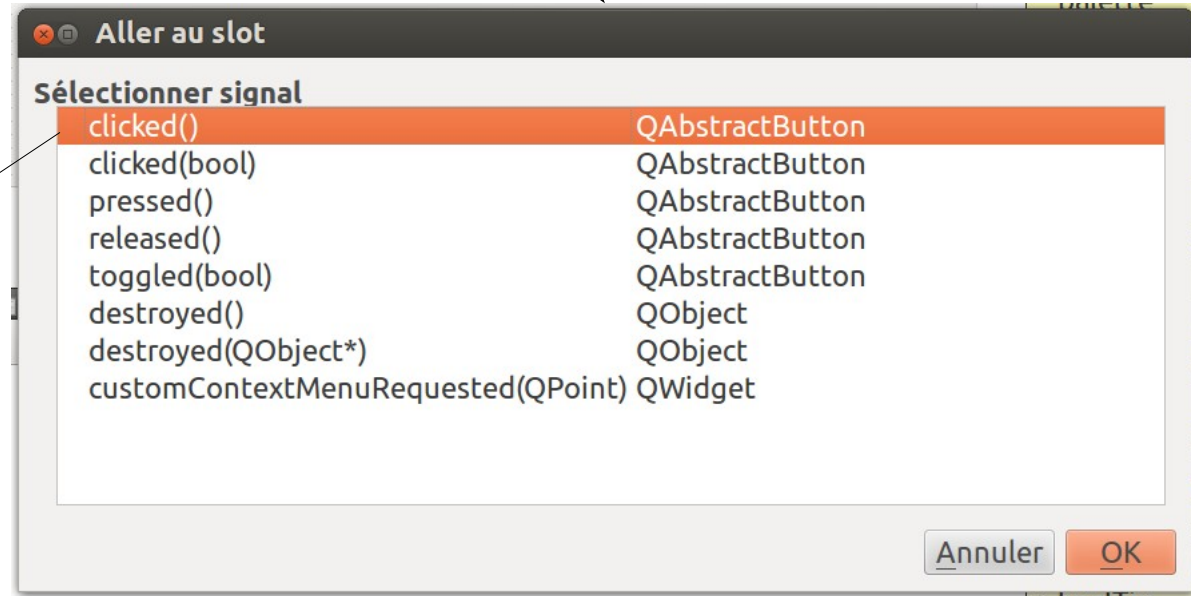
Événements signal/slots



Chaque composant Qt (représenté par une classe) peut émettre des **signaux**, correspondant à des actions, au noyau de l'application
On peut connecter un de ces signaux à une méthode d'une classe quelconque (**slot**)

connexion d'un signal à un slot dans le designer

Click droit



On choisit le signal
que l'on veut gérer,

Fichier d'entête :
une nouvelle
méthode est créée

Fichier source : implémentation de la méthode

```
private slots:  
    void on_toolButton_clicked();
```

```
void ApplicationDemo::on_toolButton_clicked()  
{  
  
    // Écrire ici le code qui se déroulera lorsqu'on  
    // aura cliqué sur le bouton « Valider »  
}
```

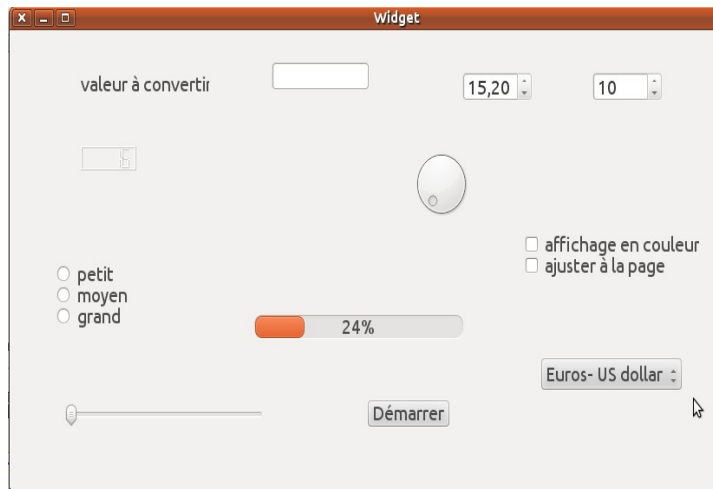
On pourrait également le faire par programme :

```
connect (objet_émetteur, signal, objet_récepteur, slot)
```

Actions courantes sur les composants de base



```
this->ui->composant->methode(arg1, arg2, ...);
```



QWidget

La fenêtre principale de l'application

```

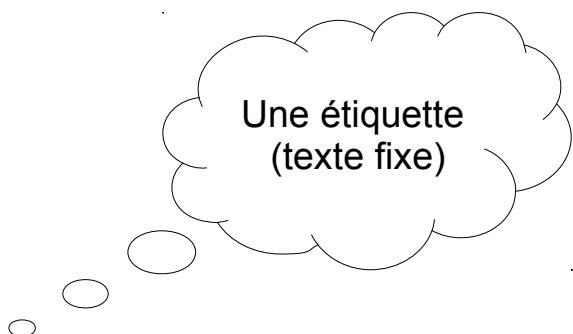
this->setVisible(true);           // rend visible
this->setVisible(false);          // rend invisible
...
QFont serifFont("Times", 10, QFont::Bold);
this->setFont(serifFont);
...

this->setCursor(Qt::WaitCursor);  // curseur sablier
...
// la taille et la position par rapport au coin haut gauche de l'écran
QRect r;                          // un rectangle
r = this->geometry();
r.setWidth(r.width()+30);          // augmente la largeur de 30 pixels
this->setGeometry(r);
...
this->setWindowTitle("Essai - version 1.00");
...
this->close();                    // ferme le widget
...

```


QLabel

valeur à convertir



Une étiquette
(texte fixe)

```
ui->label->setText("salut");    // modifie le texte

ui->label->setVisible(true);     // visible
ui->label->setVisible(false);    // invisible

QString contenu = ui->label->text();    // le texte
contenu = contenu + " les GEII";      // concaténation
ui->label->setText(contenu);         // salut les GEII

if (ui->label->isVisible()) {        // est il visible?
    }
    else { // pas visible
    }
```

QLineEdit



Attention,
un QLineEdit
contient du texte
(QString)

```
ui->lineEdit->setVisible(true);  
  
QRect r = ui->lineEdit->geometry();  
  
ui->lineEdit->setCursor(Qt::CrossCursor);  
  
double valeur;  
QString chaine;  
bool conversionOk;  
chaine = ui->lineEdit->text();  
  
valeur = chaine.toDouble(&conversionOk);  
  
if (conversionOk && valeur>=0){  
    valeur = sqrt(valeur);  
    chaine.setNum(valeur);  
    ui->lineEdit->setText(chaine);  
}  
else ui->lineEdit->setText("impossible");
```

QPushButton

A QPushButton widget with the text "Démarrer". The button is rectangular with rounded corners, a light gray background, and a thin gray border. The text is in a dark gray, sans-serif font.

```
ui->pushButton->setText("start");

ui->pushButton->setVisible(true);    // ou true

ui->pushButton->setEnabled(false);    // ou true

QRect r = ui->pushButton->geometry();

if (ui->pushButton->text()=="start")
    ui->pushButton ->setText("stop");
else ui->pushButton->setText("start");
```

QRadioButton

- ☐ petit
- ☐ moyen
- ☐ grand

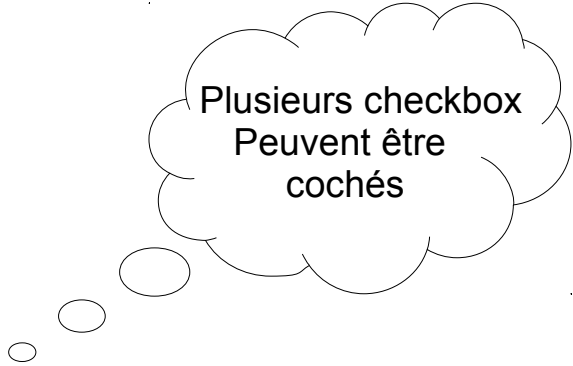


Un seul Radio-bouton enfoncé à la fois

```
ui->radioButton->setVisible(true);  
  
ui->radioButton->setText("trés grand");  
  
ui->radioButton->setChecked(true); // ou false  
  
if (ui->radioButton->isChecked()){ // il est enfoncé  
    ...  
}  
else { // il n'est pas enfoncé  
    ...  
}
```

QCheckBox

- ☐ affichage en couleur
- ☐ ajuster à la page



Plusieurs checkbox
Peuvent être
cochés

```
ui->checkBox->setVisible(true);

ui->checkBox->setText("affichage en couleur");

ui->checkBox->setChecked(true); // ou false

if (ui->checkBox->isChecked()){ // il est enfoncé
    }
else { // il n'est pas enfoncé
    }
```

QSpinBox



Pour saisir une
Valeur entière

```
ui->spinBox->setVisible(true); // ou false

// intervalle de définition
ui->spinBox->setMaximum(360);
ui->spinBox->setMinimum(-360);

// récupération de la valeur
int valeur = ui->spinBox->value();

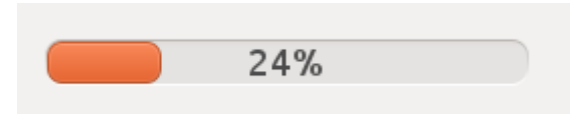
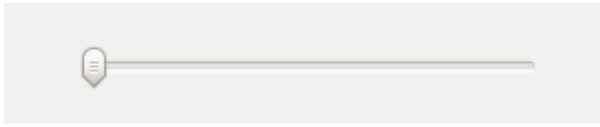
valeur *=2;

ui->spinBox->setValue(valeur);
```



QDoubleSpinBox idem mais en double

QSlider- QDial - QProgressBar



```
ui->horizontalSlider->setMaximum(200); // on minimum
ui->progressBar->setMaximum(200);

int valeur = ui->horizontalSlider->value();

ui->progressBar->setValue(valeur);

ui->dial->setValue(200-valeur);

// mais aussi :
// la géométrie
// visible ou pas
// le cursor
// la font
// .etc.
```

Les MessageBox prédéfinis

The document has been modified.

OK

```
QMessageBox msgBox;  
msgBox.setText("The document has been modified.");  
msgBox.exec();
```

The document has been modified.

Do you want to save your changes?

Don't Save

Cancel

Save

```
QMessageBox msgBox;  
msgBox.setText("The document has been modified.");  
msgBox.setInformativeText("Do you want to save your changes?");  
msgBox.setStandardButtons(QMessageBox::Save |  
                           QMessageBox::Discard |  
                           QMessageBox::Cancel);  
msgBox.setDefaultButton(QMessageBox::Save);  
int ret = msgBox.exec();
```

```
int ret = QMessageBox::warning(this, tr("My Application"),  
                               tr("The document has been modified.\n",  
                                   "Do you want to save your changes?"),  
                               QMessageBox::Save |  
                               QMessageBox::Discard |  
                               QMessageBox::Cancel,  
                               QMessageBox::Save);
```

My Application



The document has been modified.
Do you want to save your changes?

Close without Saving

Cancel

Save

Les dialogues d'entrée prédéfinis

Static Public Members

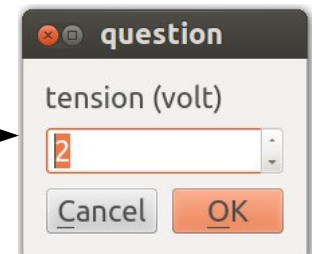
```
double getDouble(QWidget * parent, const QString & title, const QString & label, double value = 0, double min = -2147483647, double max = 2147483647, int decimals = 1, bool * ok = 0, Qt::WindowFlags flags = 0)
int getInt(QWidget * parent, const QString & title, const QString & label, int value = 0, int min = -2147483647, int max = 2147483647, int step = 1, bool * ok = 0, Qt::WindowFlags flags = 0)
int getInteger(QWidget * parent, const QString & title, const QString & label, int value = 0, int min = -2147483647, int max = 2147483647, int step = 1, bool * ok = 0, Qt::WindowFlags flags = 0) (deprecated)
QString getItem(QWidget * parent, const QString & title, const QString & label, const QStringList & items, int current = 0, bool editable = true, bool * ok = 0, Qt::WindowFlags flags = 0, Qt::InputMethodHints inputMethodHints = Qt::ImhNone)
QString getText(QWidget * parent, const QString & title, const QString & label, QLineEdit::EchoMode mode = QLineEdit::Normal, const QString & text = QString(), bool * ok = 0, Qt::WindowFlags flags = 0, Qt::InputMethodHints inputMethodHints = Qt::ImhNone)
```



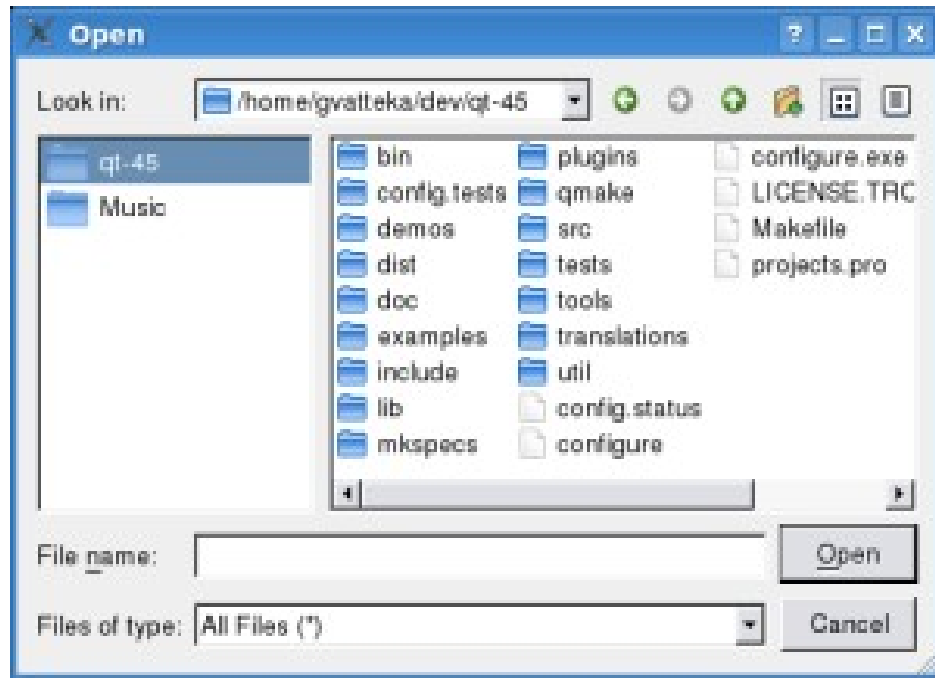
```
#include <QInputDialog>
```

```
...
```

```
int x = QInputDialog::getInt(this, "question", "tension (volt)");
```



Les dialogues de fichiers prédéfinis



```
QList<QUrl> urls;
urls << QUrl::fromLocalFile("/home/grimaldi/qt-45")
    << QUrl::fromLocalFile(QDesktopServices::storageLocation(QDesktopServices::MusicLocation));

QFileDialog dialog;
dialog.setSidebarUrls(urls);
dialog.setFileMode(QFileDialog::AnyFile);
if(dialog.exec()) {
    // ...
}
```

Pour le reste voir l'aide de Qt

Qt Creator

Fichier Édition Compiler Débugger Analyser Outils Fenêtre Aide

Index

Rechercher : QString

Qt 4.8: QString Class Reference

Filtre : Sans filtre

Public Functions

QString ()
QString (const QChar * unicode, int size)
QString (const QChar * unicode)
QString (QChar ch)
QString (int size, QChar ch)
QString (const QLatin1String & str)
QString (const QString & other)
QString (const char * str)
QString (const QByteArray & ba)
~QString ()

QString & append (const QString & str)
QString & append (const QStringRef & reference)
QString & append (const QLatin1String & str)
QString & append (const QByteArray & ba)
QString & append (const char * str)
QString & append (QChar ch)

QString arg (const QString & a, int fieldWidth = 0, const QChar & fillChar = QLatin1Char(' ')) const
QString arg (const QString & a1, const QString & a2) const
QString arg (const QString & a1, const QString & a2, const QString & a3) const
QString arg (const QString & a1, const QString & a2, const QString & a3, const QString & a4) const
QString arg (const QString & a1, const QString & a2, const QString & a3, const QString & a4, const QString & a5) const
QString arg (const QString & a1, const QString & a2, const QString & a3, const QString & a4, const QString & a5, const QString & a6) const
QString arg (const QString & a1, const QString & a2, const QString & a3, const QString & a4, const QString & a5, const QString & a6, const QString & a7) const
QString arg (const QString & a1, const QString & a2, const QString & a3, const QString & a4, const QString & a5, const QString & a6, const QString & a7, const QChar & fillChar = QLatin1Char(' ')) const
QString arg (int a, int fieldWidth = 0, int base = 10, const QChar & fillChar = QLatin1Char(' ')) const
QString arg (uint a, int fieldWidth = 0, int base = 10, const QChar & fillChar = QLatin1Char(' ')) const
QString arg (long a, int fieldWidth = 0, int base = 10, const QChar & fillChar = QLatin1Char(' ')) const
QString arg (ulong a, int fieldWidth = 0, int base = 10, const QChar & fillChar = QLatin1Char(' ')) const
QString arg (qlonglong a, int fieldWidth = 0, int base = 10, const QChar & fillChar = QLatin1Char(' ')) const
QString arg (qulonglong a, int fieldWidth = 0, int base = 10, const QChar & fillChar = QLatin1Char(' ')) const
QString arg (short a, int fieldWidth = 0, int base = 10, const QChar & fillChar = QLatin1Char(' ')) const
QString arg (ushort a, int fieldWidth = 0, int base = 10, const QChar & fillChar = QLatin1Char(' ')) const
QString arg (QChar a, int fieldWidth = 0, const QChar & fillChar = QLatin1Char(' ')) const
QString arg (char a, int fieldWidth = 0, const QChar & fillChar = QLatin1Char(' ')) const
QString arg (double a, int fieldWidth = 0, char format = 'g', int precision = -1, const QChar & fillChar = QLatin1Char(' ')) const
const QChar at (int position) const

Des milliers de classes
Des milliers de méthodes
Des milliers de pages
Des milliers d'exemples

Et en plus

...

En anglais !!!!!!!

