Sidenotes:

- Aveti numele fisierului in care se afla fiecare tutorial langa nume
  Deci pentru T3 ,T4 ,T5 ,T6 , proiectul se afla in fisierul T3.
- Anumite tutoriale au putine spre deloc notite, tot ce era nevoie se afla in cod si/sau in comentariile codului.
- Tutorialele 43-46 nu mai sunt valabile in versiuni actuale Qt .

T1

Qt este cross platform, poate fi utilizat pe unix, windows :)

T2 hello-t2

```
#include <QCoreApplication>

#include <QDebug> //folosim qdebug pt meniul de debug

int main(int argc, char *argv[])

{

   QCoreApplication a(argc, argv); //creeaza o instanta a aplicatiei

   qDebug() << "Hello World";  //afiseaza hello world

   return a.exec();

}
```

T3 - T3

din QT

MainWindow - numele clasei

.ui – extensia din qt pt form-uri

Structural, toate fisierele sunt xml

Q Object - baselevel pt orice din qt (ca object in java)

Pentru a edita Gui-ul intram in mainwindow.ui

si putem da drag and drop componentelor dorite

apoi mergem in mainwindow.cpp si putem modifica textul

ui->pushButton->setText

in design -> signals and slots -> drag de la componenta -> se deschide Configure Connection

apasam pe clicked() apoi bifam pe show signals si dam close din main window pt a

->> buton va inchide fereastra la apasare

T4 - T3

Signals and slots

-Fiecare signal si slot trb sa aibe acelasi numar de argumente

n cazul nostru avem valueChanged(int) din horizontal slider si setValue(int) , ambele avand 1 arg

-Pentru a sterge o connectiune, o selectam si apasam delete

putem avea mai multe signal-uri conectate la un singur slot;

T5 -T3

in design

Typehere :: File

Typehere :: New File

^ creeaza o actiune (action) careia ii putem da drag and drop in ui

daca dam click dreapta pe action-> go to slot -> triggered -> genereaza cod

pt un form nou -> source files -> add new design form class

T6 - T6

Layouts - butoane in ribbon in .ui - se folosesc cu selectarea componentelor dorite

- se folosesc deoarece qt e cross platform

-Pt butonul de ok am folosit accept() din slots -> acesta accepta optiunea si inchide fereastra de dialog

Buddies - alege ce 2 sau mai multe componente se pot bolda

Tab order - alege ordinea in care se boldeaza un obiect atunci cand se apasa tab

T7 - minimal

HTML Aware Widgets - toate label-urile sunt HTML aware, deci putem folosi sintaxe din html ca si   pt bold si   pt italic

minimal.pro

```
QT += widgets
TARGET = minimal
SOURCES += \
   main.cpp
```

main.cpp

```
#include <QApplication>
#include <QLabel>

int main(int argc, char *argv[]){

   QApplication app(argc,argv);
   //QLabel *label= new QLabel("<b>Hello</b>  <i>Worlddd</i>dddd:)");
   QLabel *label= new QLabel("<h2>Hello</h2>  <font color = red><i>Worlddd</i></font>dddd:)");
   label->show(); //se incapsuleaza automat intr o fereastra

   return app.exec();

}
```

T8 - Layouts prin cod - minimal

```
QWidget *window = new QWidget();
```

```
    //atat v a afisa doar o fereastra mare, pt ca nu are parametrii
    window->setWindowTitle("<My APP>");
    QPushButton *button1 = new QPushButton("1");
    QPushButton *button2 = new QPushButton("2");
    QPushButton *button3 = new QPushButton("3");
   // QHBoxLayout *hlayout = new QHBoxLayout; //layout orizontal
    QVBoxLayout *vlayout = new QVBoxLayout; //layout vertical

    //hlayout->addWidget(button1);
    //hlayout->addWidget(button2);
    //hlayout->addWidget(button3);
    // window->setLayout(hlayout);
    vlayout->addWidget(button1);
    vlayout->addWidget(button2);
    vlayout->addWidget(button3);
    window->setLayout(vlayout);
    window->show();
    return app.exec();
```

T9 - QGridLayout - minimal

```
QGridLayout *layout = new QGridLayout();
    QLabel *label1 = new QLabel("Name");
    QLineEdit *txtName = new QLineEdit;
    layout->addWidget(label1,0,0); //(widget, row, column) //param functiei
    layout->addWidget(txtName,0,1);

    QLabel *label2 = new QLabel("Age");
    QLineEdit *txtAge = new QLineEdit;
    layout->addWidget(label2,1,0); //(widget, row, column) //param functiei
    layout->addWidget(txtAge,1,1);
```

```
    QLabel *label3 = new QLabel("Height");
    QLineEdit *txtHeight = new QLineEdit;
    layout->addWidget(label3,2,0); //(widget, row, column) //param functiei
    layout->addWidget(txtHeight,2,1);

    QPushButton *button = new QPushButton("Ok");
    layout->addWidget(button,3,0,1,2);// (widget, row, column, how many rows we want it to span, how many collumns we want it to span)

    window->setLayout(layout);
```

T10 - t10

splitters

Se selecteaza obiectele ui dorite si se alege din ribbon horizontal sau vertical splitter

ca sa revenim la pozitiile initiale dam click dr pe form - lay out - break

T11 - Dirs

QDir - directoare

```
#include <QDebug>
#include <QDir>
```

cand folosim un path hardcoded schimbam orientarea /-urilor

in mod normal (in windows, linux): C:\Users\Acasa\Desktop\Uni\Programare\TAP\Proiecte\Qt

in qt: "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt"

qFileInfo tine informatia despre fisiere

```
#include <QCoreApplication>
#include <QDebug>
#include <QDir>
```

```
#include <QFileInfo>
#include <QString>
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    //QDir mDir("C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt");

    //qDebug() << mDir.exists(); //returneaza true daca exista si false daca nu

    /*foreach(QFileInfo mItm , mDir.drives())// ne v a selecta toate drive-urile calculatorului
    {
        qDebug() << mItm.absoluteFilePath();
    }*/


    // QDir mDir; // va lua dir curent
    //QString mPath="C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/Dirs/exemplu";
   /* if(!mDir.exists(mPath)) // folosing mPath se va creea un director daca nu exista
    {
        mDir.mkpath(mPath); //creaza un director
        qDebug()<<"Created!";
    }
    else
    {
        qDebug()<<"Already Exists!";
    }
*/
    QDir mDir("C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt");
    foreach(QFileInfo mItm , mDir.entryInfoList()){ // ne va afisa toate path-urile fisierelor din mDir
       // qDebug() << mItm.absoluteFilePath(); // afiseaza path-ul lui mItm
        if(mItm.isDir()) qDebug() << "Dir:" << mItm.absoluteFilePath(); //isDir verifica daca e diector
```

```
    if(mItm.isFile()) qDebug() << "File:" << mItm.absoluteFilePath(); //isFile verifica daca e file
  }
  return a.exec();



}
```

```
Dir: "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/build-T3-Desktop
Dir: "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/build-T3-Gui-Des
Dir: "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/build-T6-Desktop
Dir: "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/Dirs"
Dir: "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/hello-T2"
Dir: "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/minimal"
Dir: "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/Splitter01"
Dir: "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/T3"
Dir: "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/T6"
File: "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/txtQt.txt"
```

T12 – File

Qfile – interfata de citire si scriere in fisiere

```
#include <QCoreApplication>
#include <QFile>
#include <QString>
#include <QDebug>
#include <QTextStream> //interfata de scriere citire text


//functie scriere
void Write (QString Filename){
  QFile mFile(Filename);
  if(!mFile.open(QFile::WriteOnly | QFile::Text)){//verifica daca e writeonly si text si deschis
    qDebug() << "Couldnt open";
    return;
  }
  QTextStream out(&mFile); // referinta catre fisierul deschis
  out<<"Hello World";
```

```
    mFile.flush(); //

    mFile.close();
}

void Read (QString Filename){
    QFile mFile(Filename);
    if(!mFile.open(QFile::ReadOnly | QFile::Text)){ //verifica daca e readonly si text si deschis
        qDebug() << "Couldnt open";
        return;
    }
    QTextStream in(&mFile); // referinta catre fisierul deschis // mai multe functii putem gasi in help
    QString mText = in.readAll(); //citeste tot din fisierul text
    qDebug() << mText;

    mFile.close();
}

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    QString  mFilename = "C:/Users/Acasa/Desktop/Uni/Programare/TAP/Proiecte/Qt/myfile.txt";
    Write(mFilename);
    Read(mFilename);

    return a.exec();
}
```

T13 – File

(Proiectul este qmake nu cmake la crearea acestuia)

Resource files – collection of data that can be put into exec when its compiled and u can access that data at runtime

Se da Click dr in solution explorer -> add new -> qt -> resource file (asa se creeaza un nou resource file)

In fereastra de resurse, avem nevoie intai de un prefix, caruia ii putem adauga fisiere de resurse(alte .pro-uri , imagini , etc)

```
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    //QString  mFilename = "C:/Users/Acasa/Desktop
    // Write(mFilename);
    Read(":/MyFiles/File.pro"); //MyFiles=prefix f
    /*QT = core
```

```
04:19:58: Starting C:\Users\Acasa\Desktop\Uni\Programare\TAP\Proiecte\Qt\build-File-Desktop_Qt_6_6_0_MinGW_64_bit-Debug\debug\File.exe...
Couldnt open
04:21:09: C:\Users\Acasa\Desktop\Uni\Programare\TAP\Proiecte\Qt\build-File-Desktop_Qt_6_6_0_MinGW_64_bit-Debug\debug\File.exe crashed.

04:21:14: Starting C:\Users\Acasa\Desktop\Uni\Programare\TAP\Proiecte\Qt\build-File-Desktop_Qt_6_6_0_MinGW_64_bit-Debug\debug\File.exe...
"QT = core\n\nCONFIG += c++17 cmdline\n\n# You can make your code fail to compile if it uses deprecated APIs.\n# In order to do so, uncomment the following line.\n#DEFINES +=
QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0\n\nSOURCES += \\\n        main.cpp\n\n# Default rules for deployment.\nqnx: target.path = /
tmp/$${TARGET}/bin\nelse: unix:!android: target.path = /opt/$${TARGET}/bin\n!isEmpty(target.path): INSTALLS += target\n\nRESOURCES += \\\n    MyResources.qrc\n"
```

T14 – GUI

Qlabel si Qt designer

La fel ca si java , avem tabela de proprietati;

Pentru Label-uri am putea ,

din proprietati : sa schimbam textul , sa deschidem un editor text tip word sau html

din cod ui->label->setText("eyyy"); si modificarile de tip bold etc se fac in sintaxe html

daca facem ambele , codul v a da override

T15 – GUI

Pushbutton

Pt a avea parte de o actiune la folosirea butonului dam click dr pe el -> go to slot -> clicked()

Acesta va genera o functie in care scrie

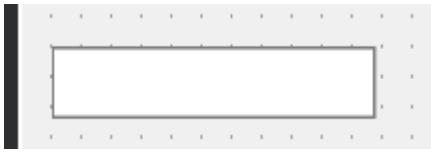
void Dialog::on_pushButton_clicked()

```
{
    QMessageBox::information(this, "Title here", "Text here");

}
```

T16 – Gui

QLineEdit -> textbox din java



Echo mode (in proprietati) => vizualizarea a timpul scrierii

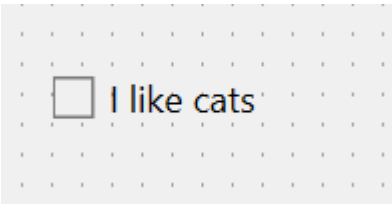
Putem avea normal (se vad literele in timp ce scrii)

Sau password – se vad stelute

Sau alternative

T17 – gui

Qcheckbox

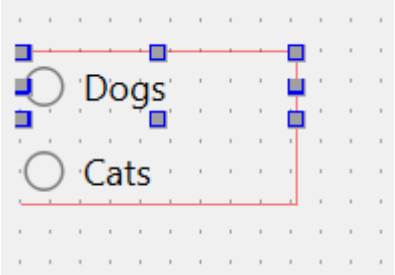


Setarea starii checkbox-ului : ui->checkBox->setChecked(true);

Verificarea starii checkboxului : if(ui->checkBox-> isChecked())

T18 – Gui

Qradiobutton

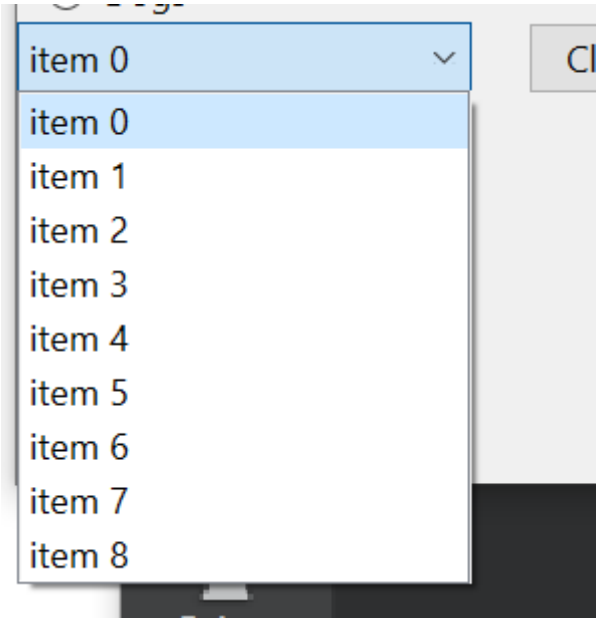


Diferenta dintre radio button si checkbox este ca cu radio buttons poti allege o singura varianta pe cand cu checkbox-urile poti bifa mai multe

```
void Dialog::on_pushButton_3_clicked()
{
   if(ui->radioButton1->isChecked()){
      QMessageBox::information(this,ui->radioButton1->text(),"you like cats");
   }
   if(ui->radioButton2->isChecked()){
      QMessageBox::information(this,ui->radioButton2->text(),"you like Dogs");
   }
}
```

T19 – Gui

ComboBox



T20 – Gui

Listwidget

Diferit de ListView

Folosit ca o lista de diferite variante

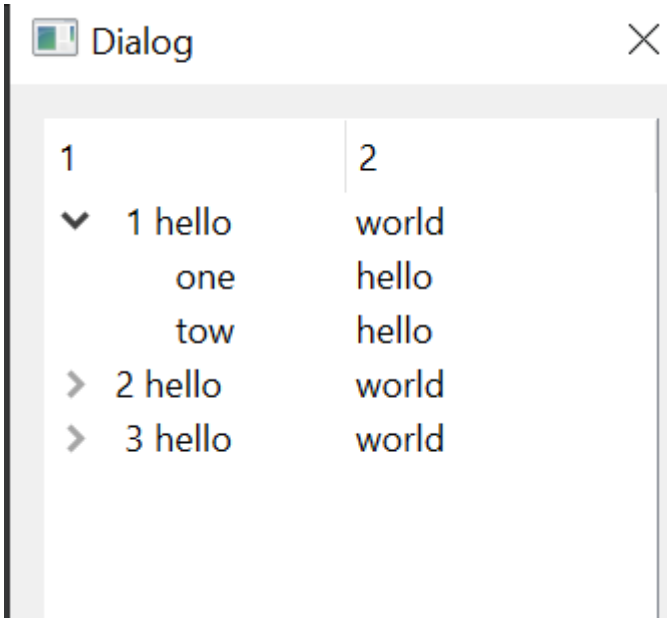
Fiecare varianta este un qlistwidgetobject cu numele item

QListWidgetItem *itm =ui->listWidget->currentItem();

Acesta are diferite proprietati cum ar fi culoarea, alignment, foreground, background etc

//T21 Gui2

QTreeWidget



Vizualizare in tree, cum ar arata un set de fisiere

Se adauga in .h functii de adaugare root si copil

```
void AddRoot(QString name, QString Description);
void AddChild(QTreeWidgetItem *parent , QString name, QString Description);
```

Apoi se implementeaza in .cpp ambele functii

```
void Dialog::AddRoot(QString name, QString Description){
    QTreeWidgetItem *itm = new QTreeWidgetItem(ui->treeWidget);
    itm->setText(0,name); //0 - coloana
    itm->setText(1,Description);
    ui->treeWidget->addTopLevelItem(itm);
```

```
void Dialog::AddChild(QTreeWidgetItem *parent , QString name, QString Description

    QTreeWidgetItem *itm = new QTreeWidgetItem();
    itm->setText(0,name); //0 - coloana
    itm->setText(1,Description);
    parent->addChild(itm);

}
```

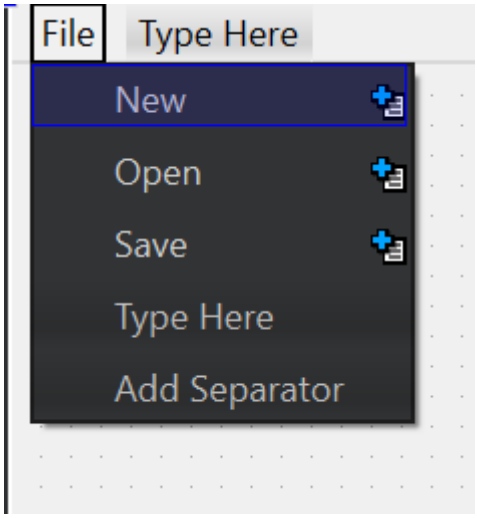
Pentru Child I se da un nod root/parinte ca pointer

//T22 – continuare T21

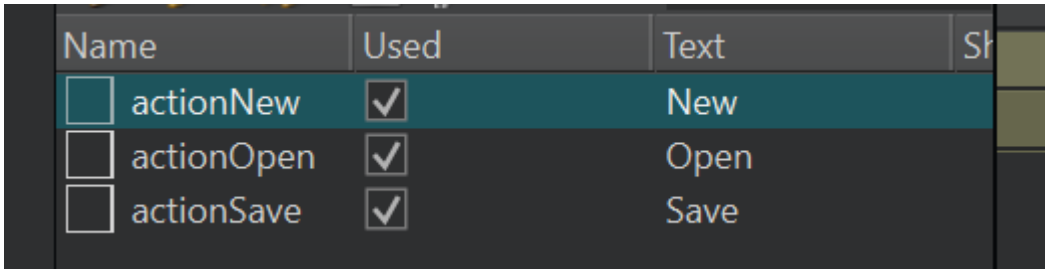
//T23 MyActions

Actions

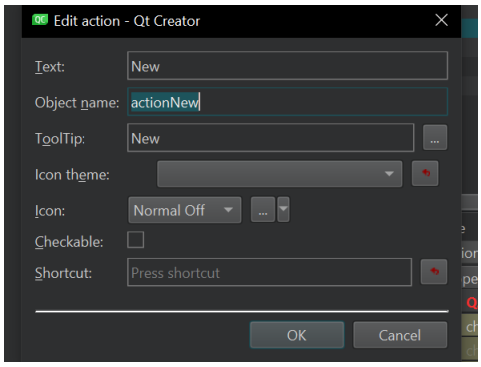
Clasa ce interactioneaza cu toolbar-ul si menubar-ul

Pot fi scrise : 

Daca dam dublu click pe una din actiuni de aici :



apare



Adaugarea de imagini se face din resurse (vezi tutorial 13)

Ca sa schimbam imaginea unei actiuni , la icon -> … -> selectam din imagine

Actiunile pot fi utilizate din toolbar precum si din file-ul nostrum

T24 – Gui3

QSlider si QprogressBar

Conectam manual semnalul slider la slotul progress bar

in cazul nostru avem valueChanged(int) din horizontal slider si setValue(int)

T25 – StatusBar

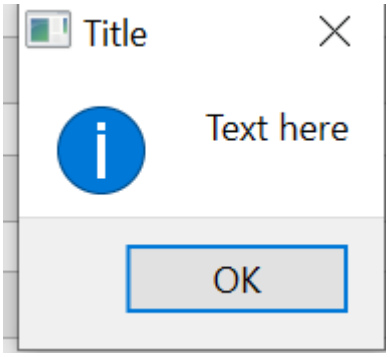
Se afla implicit in partea de sud a form-ului mainwindow

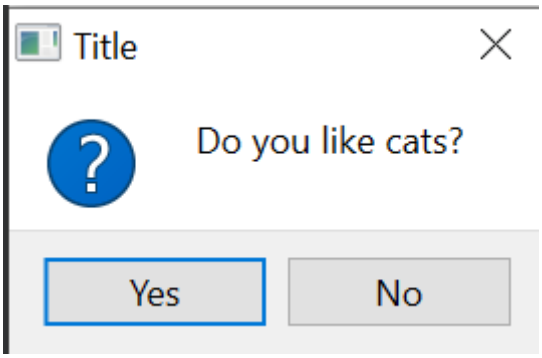
Unde putem pune widget-uri

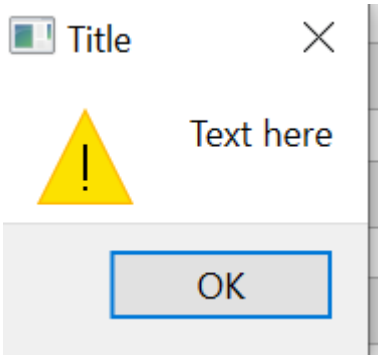
T25- Message

QMessageBox

Exista 3 tipuri de message box

Information 

Question 

Warning 

T27 –Timer

QTimer – clasa cu care putem declara un pointer

```
  QTimer *timer;
```

Si a il folosi

L-am folosit pt a afisa “Timer executed” dupa o durata de timp

```
timer = new QTimer(this);
   connect(timer,SIGNAL(timeout()),this,SLOT(MySlot()));
   timer->start(1000);
```

T28 - MyThread

Creating a thread

Se creeaza o clasa ce mosteneste qThread si se face override la run();

```
#include "mythread.h"
#include <QDebug>

MyThread::MyThread()
{

}
void MyThread::run()
{
   qDebug() << "Running";
}
#include <QCoreApplication>
#include "mythread.h"

int main(int argc, char *argv[])
{
   QCoreApplication a(argc, argv);
   MyThread mThread;
   mThread.start(); // putem avea prioritatea ca si argument
```

```
    return a.exec();
}
```

Mai multe thread-uri mergand simultan , fara ca acestea sa aibe o prioritate anume:

```
"mThread1" Running
"mThread2" Running
"mThread3" Running
"mThread1" 0
"mThread1" 1
"mThread1" 2
"mThread1" 3
"mThread1" 4
"mThread1" 5
"mThread3" 0
"mThread2" 0
"mThread1" 6
```

T29 - MyThread

nu este suportat pe pc-ul personal dar codul ar trebui sa functioneze

Thread cu priority

Thread-uri atunci cand thread 1 si thread 3 au highest priority

"mThread3" 0

"mThread1" 0

"mThread1" 1

"mThread3" 1

"mThread1" 2

"mThread3" 2

"mThread2" 0

"mThread2" 1

"mThread2" 2

T30 – MyThread

QMutex – semafor

Deadlock – cand doua thread-uri incearca sa foloseasca aceeasi resursa in acelasi timp

Folosim un bool Stop pt a tine cont de ocuparea resurselor

Se declara : QMutex mutex;

2 functii importantel : mutex.lock(); .unlock();

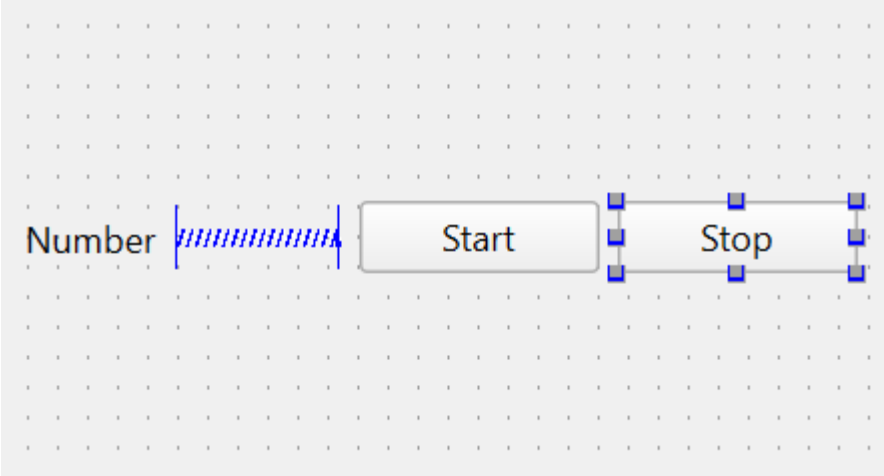
T31 MyGuiThread;

Threads with GUI

Avem un thread care schimba textul : number

Threadul incepe la apasarea start

Si se opreste la apasarea stop



T32 – community feedback ☺

T33 Waiter

QThread – Waiting

mThread.wait(); comanda ce asteapta thread-ul sa se completeze

deci nicio alta linie de cod nu se executa pana la finisarea wait-ului

T34 – installing questions ☺

T35 -Threads done right

Threading done correctly – forumurile spun una , help din qt creator spune alta :/

Ideea este ca nu vrei sa folosesti threaduri prin mostenirea QThread

Deci se mosteneste Qobject

Apoi

```
QThread cThread;
   MyObject cObject;
   cObject.DoSetup(cThread);
   cObject.moveToThread(&cThread);
   cThread.start();
```

T36 - MyList

Qlist = container

Se declara : QList<int>List;

Functie adaugare element : List.append(i);

Functie stergere element : List.removeOne(5);

T37-MyList

ListIterator ,Qlist

O alta metoda de a itera prin Qlist

```
QListIterator<int>Iter(List);
while(Iter.hasNext()) //hasPrevious in caz ca nu aveam toBack
  {
     qDebug() << Iter.next();
     if(Iter.hasNext()){
        qDebug() << "next..." <<Iter.peekNext();
     }

  }
  return a.exec();
```

38 -MyList

QMutableListIterator O alta metoda de a itera prin Qlist

Care, spre deosebire de list iterator, poate sterge elem.

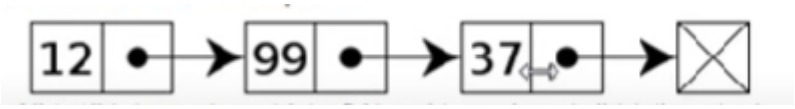
```
QMutableListIterator<int> Iter(List);
  while(Iter.hasNext())
  {
     int i=Iter.next();
     if(i==5){
        Iter.remove();
```

```
    }
  }
  Iter.toFront();
  while(Iter.hasNext())
  {
    qDebug() << Iter.next();
  }

  return a.exec();
```

T39 -Linked

QLinkedList – lista simplu inlantuita



//nu mai exista in versiuni noi de qt

Dar asa s-ar fi declarat una :

```
QLinkedList<int> List;
  List <<1 <<3 <<5;
```

T40 – MapsTest

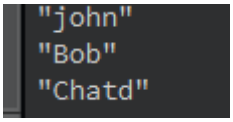
QMap= un container asociativ sortat, în care datele sunt stocate sub formă de perechi cheie-valoare

Functia insert – introduce o pereche de key(cheie) si values, daca exista deja , perechea initiala va fi suprascrisa

Deci

Pentru

```
Employees.insert(1,"john");
  Employees.insert(2,"Bob");
  Employees.insert(3,"Chatd");
```

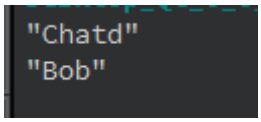
Avem : 

Si pentru

Employees.insert(1,"john");

Employees.insert(2,"Bob");

Employees.insert(1,"Chatd");

Avem: 

Cheia si valoarea pot fi luate individual cu un iterator

```
QMapIterator<int,QString>Iter(Employees);
   while(Iter.hasNext()){
      Iter.next();
      qDebug() << Iter.key() << "  " << Iter.value();
   }
```

T41 – HashTest

QHash

Putem observa ca implementarea pt qHash si Qmap sunt practic identice

Dar

QHash este mai rapid decat Qmap din cauza eficientei cautarilor

Qmap sorteaza mereu dupa cheie pe cand Qhash sorteaza arbitrar

42 -String List

QStringList – adauga functii mai multe decat Qlist<QString>

Cum ar fi :

```
List = Line.split(","); //functie din qstring care taie fiecare element cu un caracter dat
   List.replaceInStrings("b","BATMAN"); //cand un elem este "b" va deveni batman; replacing
   QString After = List.join(" ... "); //converteste lista intr un qstring
```

T43 –44-45-46 Indisponibil

In proiectul Algoritm am introdus alternative din std::

T47- ModViewTest

Model View Programming

Model – data

View – ui

Orice widget ce are view in nume (list view, treeview) urmareste arhitectura model view

T48 –DirMod

QdirModel nu mai exista!

Poate fi inlocuit cu QFileSystemModel

Aceeasi idee ca si T49 doar ca pentru Directoare si fisiere

T49 -FileSystem

QFileSystemModel

-folosit in loc de QDirModel pentru ca nu blocheaza ui-ul si este mai eficient

T50 - MyDelegate

Delegates – de ajuta sa vizualizezi si edit un item intr-un model