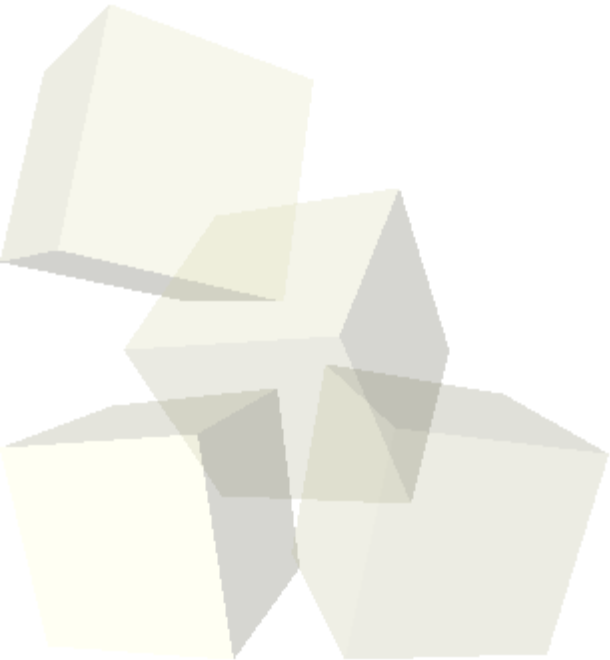




# Qt ile Programlama

İşbaran Akçayır  
<http://ish.kodzilla.org>



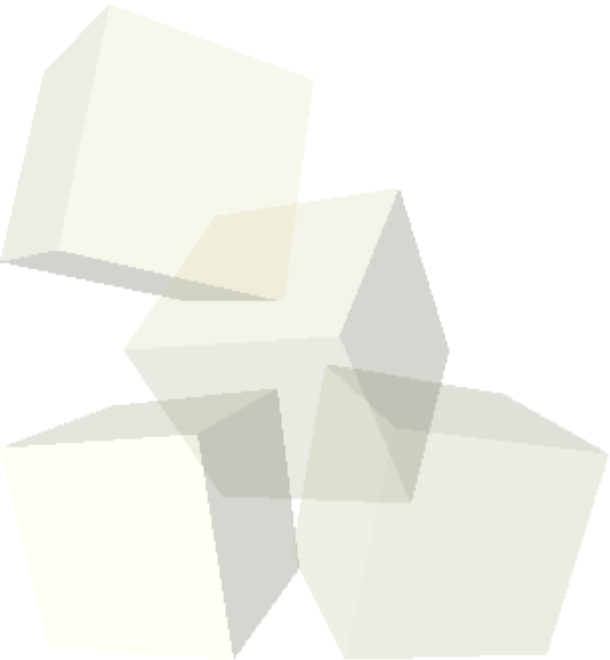
# Nedir

- Qt, platformlar arası bir uygulama geliştirme sistemidir. (ya da iskeleti diyebiliriz)
- Genellikle GUI içeren programlar geliştirmede kullanılır, Qt 4 sürümü ile konsol araçları ya da sunucular gibi grafik arayüz içermeyen programlar geliştirmede de kullanılır olmuştur.
- Qt kullanılan bir kaç proje: Opera, Google Earth, Skype, Qtopia (Qt Palmtop Environment), OPIE( Open Palmtop Integrated Environment ) ve KDE.
- Norveçli Trolltech ( Quasar Technologies ) tarafından geliştiriliyor.



# Nedir

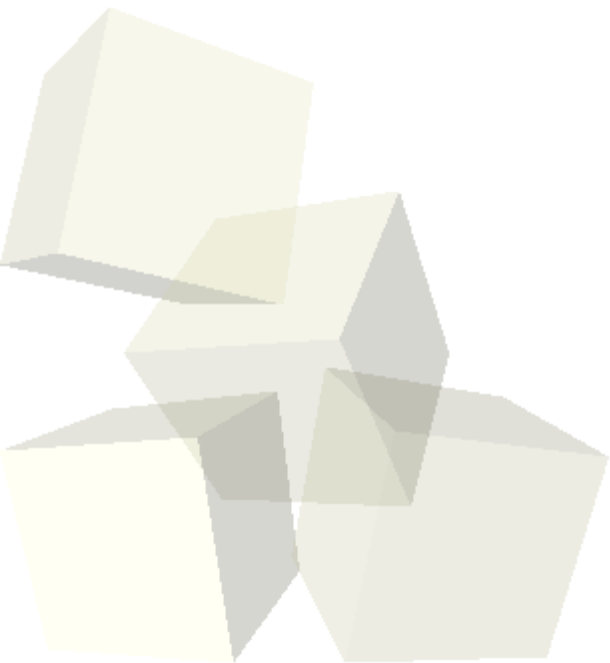
- Qt bir C++ sınıf kütüphanesi, ve geliştirme ve uluslararasılaştırma için araçlar içerir.
- Qt standart C++ kullanır. Ama diğer dillerde kod yazımına da olanak sağlayan bağlayıcıları içerir (binding) :
  - Python (PyQt), Ruby (QtRuby), PHP, C, Perl, Pascal, C#, Java (Jambi)





# Platformlar

- Qt/X11 — X Window System (Unix / Linux)
- Qt/Mac — Apple Mac OS X
- Qt/Windows — Microsoft Windows
- Qt/Embedded — Gömülü platformlar (PDA, Smartphone, ...)
- Qt/Jambi — Java platformu geliştirilmesi





# Dağıtımlar

- Qt Console — Konsol tabanlı programlama için
- Qt Desktop Light — Giriş seviyesinde, ağ ve veri tabanı desteği yok.
- Qt Desktop

Qt Open Source Edition

Qt Commercial Edition

◆ Çift Lisanslı




```
#include <QApplication>
#include <QPushButton>

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

    QPushButton hello("Hello world!", 0);
    hello.resize(100, 30);

    hello.show();
    return app.exec();
}
```

- 
- Qt kullanan her GUI uygulama bir QApplication nesnesi içermek zorundadır.
  - QApplication uygulama genelinde kaynakların yönetilmesinden sorumlu.  
( fontlar, fare imleci .. )
  - QPushButton ve Widget'ler ..
  - `app.exec();` ile `main()` kontrolü Qt'ye devreder.

derleme:

- ◆ `qmake -project`
- ◆ `qmake`
- ◆ `make`



# Örnek proje dosyası

```
// Automatically generated by qmake (2.0.1a) Sat Mar 15 00:50:10 2007  
#####  
#####
```

```
TEMPLATE = app  
TARGET =  
CONFIG += qt debug opengl thread  
DEPENDPATH += .  
LIBS += -lkdecore -lkdeui -L/usr/kde/3.5/lib/  
INCLUDEPATH += . /usr/kde/3.5/include  
  
# Input  
HEADERS += hello.h  
SOURCES += hello.cpp main.cpp  
RESOURCES += application.qrc  
QT += network xml ( veya QT -= gui )  
  
win32 {  
    SOURCES += hellowin.cpp  
}  
unix {  
    SOURCES += hellounix.cpp  
}  
!exists( main.cpp ) {  
    error( "main.cpp dosyasi bulunamadi" )  
}
```



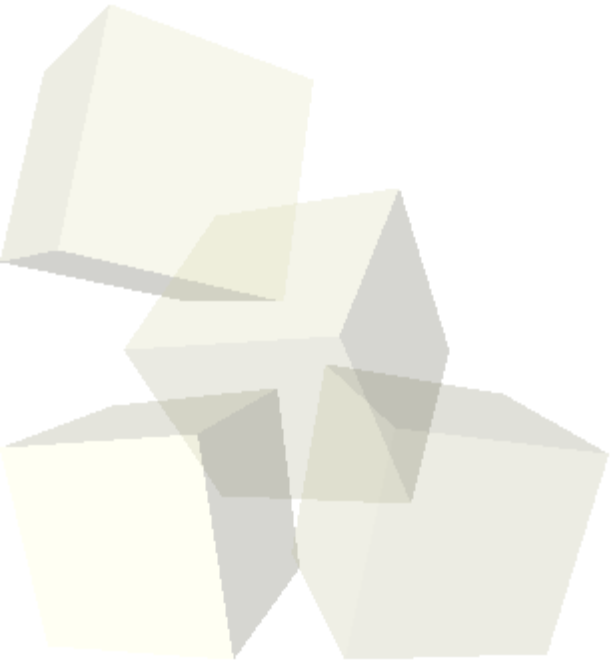
# Kaynak dosyası

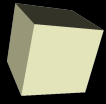
```
<!DOCTYPE RCC><RCC version="1.0">  
<qresource>  
  <file>images/copy.png</file>  
  <file>images/cut.png</file>  
  <file>images/new.png</file>  
  <file>images/open.png</file>  
  <file>images/paste.png</file>  
  <file>images/save.png</file>  
</qresource>  
</RCC>
```



# Sınıflar Hakkında

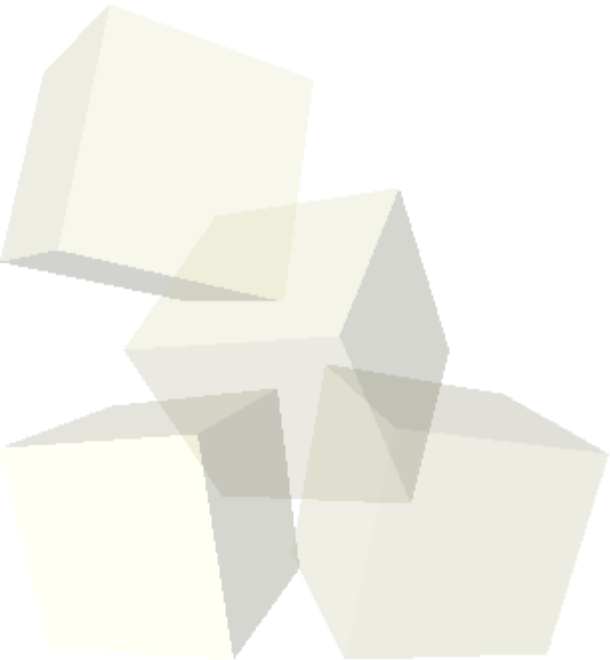
- Qt4 farklı kütüphaneler içinde olan bir çok modülün bir araya gelmesiyle oluşuyor.
- QtCore, QtGui, QtNetwork, QtOpenGL, QtSql ..
- QtCore ve QtGui ön tanımlı





# Qt Nesne Modeli

- Standart C++ nesne modeli çalışma anında nesne paradigması için çok etkin.
- Ama bazı problemlerin çözümünde doğası gereği statik olmasının dezavantajları var.
- GUI programlamada ise hem çalışma anında efektif olmalı, hem de yüksek esnekliği olmalı.





# Qt = C++ +

- Sinyal ve Slot mekanizması
- Dizayn edilebilir nesne özellikleri -
- Event ve event filter mekanizması
- Stringler üzerinde güçlü işlemler ( Ulusallaştırma )
- Timer ( zamanlayıcılar ) -
- Hiyerarşik nesne yapısı -
- Korumalı işaretçiler ( Qpointer )
- Dynamic cast
- ....
- ...
- ..



# Stiller

- Qt'nin widget stilleri programınızın masaüstü ortamına tam uyum sağlayabilmesini sağlar.

Name	Size
Makefile	7 KB File
forms	Fold
gallery...	6 KB qdc
gallery...	6 KB qdc
gallery...	6 KB qdc
gallery...	6 KB qdc
gallery...	6 KB qdc

Styles Margins

Heading

Paragraph

List

Footnote

Styles Margins

Heading

Paragraph

List

Footnote

Styles Margins

◇ Heading

◇ Paragraph

◇ List

◇ Footnote

Styles Margins

Heading

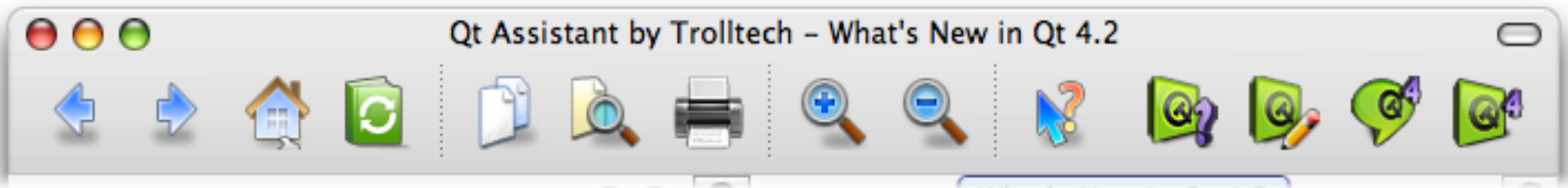
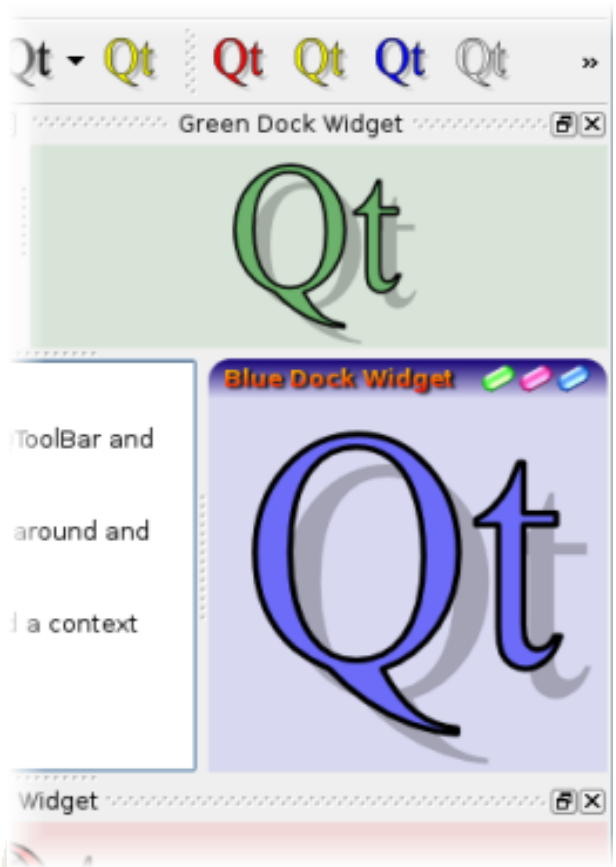
Paragraph

List

Footnote

Name	Size
Makefile	7 KB File
forms	Fo
galler...	6 KB qd
galler...	6 KB qd
galler...	6 KB qd
galler...	6 KB qd
galler...	6 KB qd

January	6
February	3
March	2
April	3
May	6





# Layout Sistemi ( Dağılım )

## ■ Horizontal, Vertical, Grid

- ◆ QHBoxLayout, QVBoxLayout, QGridLayout

```
QWidget *window = new QWidget;  
QPushButton *button1 = new QPushButton("One");  
QPushButton *button2 = new QPushButton("Two");
```

```
QHBoxLayout *layout = new QHBoxLayout;  
layout->addWidget(button1);  
layout->addWidget(button2);
```

```
window->setLayout(layout);  
window->show();
```





# Containers

- `QList<T>` ( Diziler, index ile hızlı erişim, `QList::append()`,  
`QList::prepend()` .. )
- `QLinkedList<T>` ( Bağlı Liste )
- `QVector<T>` ( Başa veya sona ekleme zor olabilir )
- `QStack<T>`
- `QQueue<T>`
- `QMap<Key, T>` ( Anahtar sırasına göre depolar )
- `QMultiMap<Key, T>` ( Multi value )
- `QHash<Key, T>` ( Çok daha hızlı arama, rasgele depolama )
- `QMultiHash<Key, T>`



# QPointer

- QPointer<T> ( T, QObject'ten türemiş )

```
QLabel *label = new QLabel;  
QPointer<QLabel> safeLabel = label;  
safeLabel->setText("Hello world!");  
delete label;
```

- Korumalı işaretçiler otomatik olarak T \* tipe dönüştürülür, yani korumasız olanlarla karıştırılması sorun yaratmaz.



# Uluslararasılaştırma

- Hazır metin motorunun gücü
- Destek kapsüllenmiştir.
- Unicode 4.0 kullanan QString kullanın
- Tüm metinlerde tr() kullanın

```
LoginWindow::LoginWindow()
```

```
{
```

```
    QLabel *label = new QLabel("Password:"); //
```

yanlış

```
    QLabel *label = new QLabel(tr("Password:"));
```

```
    ...
```

```
}
```

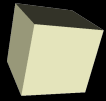
```
exitAct = new QAction(tr("E&xit"), this);
```

```
exitAct->setShortcut(tr("Ctrl+Q"));
```



# QT Linguist

- Bir Qt uygulamasının dil dönüşümü 3 aşamadır.
  - lupdate
    - C++ kaynak kodlarından çevirisi yapılacak metnin çıkarılması için .ts dosyalarının oluşturulması
  - Qt Linguist
    - .ts dosyalarının ( xml dosyaları ) Qt Linguist kullanarak hedef dile çevrilmesi
  - lrelease
    - .ts dosyasından bir mesaj dosyası ( .qm ) elde etmek için kullanılır. .ts dosyalarını “kaynak dosyalar”, .qm dosyalarını da “nesne dosyaları” olarak düşünebiliriz. Kullanıcılar sadece .qm dosyalarına ihtiyaç duyar. ( iki dosya da platform ve local bağımsızdır.
  - ve .pro dosyası:
    - TRANSLATIONS = uygulamaadi\_tr.ts \  
uygulamaadi\_fi.ts \  
uygulamaadi\_de.ts \  
uygulamaadi\_es.ts \  
uygulamaadi\_fr.ts \  
uygulamaadi\_it.ts \  
uygulamaadi\_ja.ts \  
uygulamaadi\_pt.ts \  
uygulamaadi\_ru.ts \  
uygulamaadi\_uk.ts \  
uygulamaadi\_zh\_CN.ts \  
uygulamaadi\_zh\_TW.ts



# uygulama içi kullanım

```
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    QTranslator myappTranslator;
    myappTranslator.load("myapp_" +
    QLocale::system().name());
    app.installTranslator(&myappTranslator);

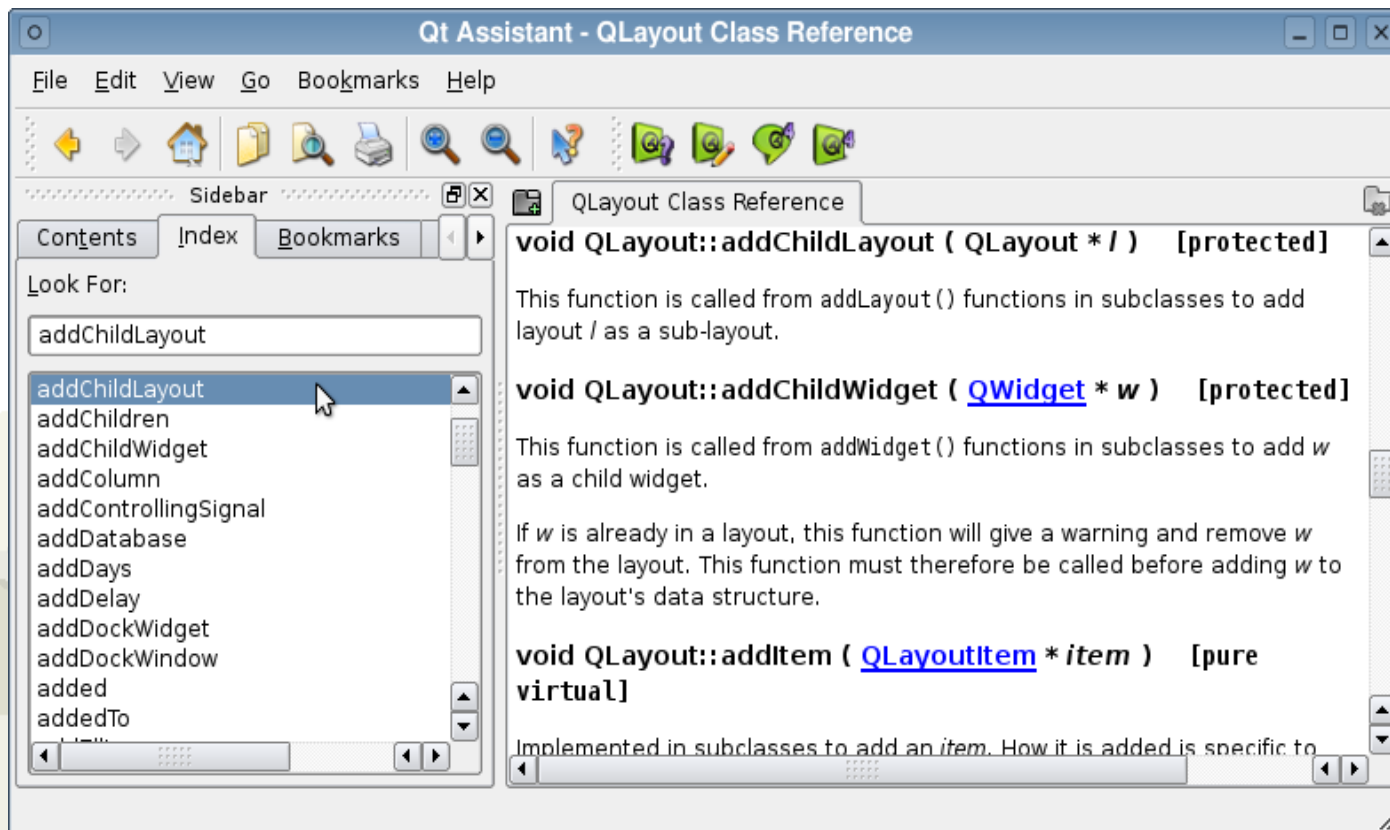
    ...
    return app.exec();
}
```

Ayrıca bkznz. ( QTextCodec .. I/O codec desteği )



# Qt Assistant

- Qt en iyi dokümantasyonu olan proje
- Açıklamalar hem kaynak kodları yazanlar tarafından .cpp dosyalarına yazılıyor, hem de dokümantasyon takımı web sitesi vs. ile sınıflar vb. alanların dokümantasyonunu hazırlıyor.





# Qt Designer

The screenshot displays the Qt Designer application window. The menu bar includes File, Edit, Form, Tools, Window, and Help. The toolbar contains various icons for editing and design. On the left, the 'Widget Box' is organized into three categories: Containers (Group Box, Tool Box, Tab Widget, Stacked Widget, Frame, Widget, Dock Widget), Input Widgets (Combo Box, Font Combo Box, Line Edit, Text Edit, Spin Box, Double Spin Box, Time Edit, Date Edit, Date/Time Edit, Dial, Horizontal Scroll Bar, Vertical Scroll Bar, Horizontal Slider, Vertical Slider), and Display Widgets (Label, Text Browser, Graphics View, Calendar, LCD Number, Progress Bar, Horizontal Line). The central design canvas, titled 'Form - designer.ui', shows a grid with a text label 'bu etiketin nesne ismi label'. A preview window, titled 'Form - [Preview]', shows the rendered appearance of the form with the same text and two buttons labeled 'Tamam' and 'Nayır'. On the right, the 'Property Editor' displays a table of properties for the selected object.

Property	Value
<b>QObject</b>	
objectName	Form
<b>QWidget</b>	
windowModality	Qt::NonModal
enabled	true
<b>geometry</b>	[0, 0, 400, 300]
sizePolicy	[Preferred, Pref...
minimumSize	[0, 0]
maximumSize	[16777215, 16...
sizeIncrement	[0, 0]
baseSize	[0, 0]
palette	
font	Ȧ [Sans Serif,...
cursor	Arrow
mouseTracking	false
focusPolicy	Qt::NoFocus
contextMenuPolicy	Qt::DefaultCont...
acceptDrops	false
<b>windowTitle</b>	Form
windowIcon	
tooltip	
statusTip	
whatsThis	
layoutDirection	Qt::LeftToRight
autoFillBackground	false
stylesheet	

Buttons at the bottom of the Property Editor: Reso..., Prop..., Signal/..., Objec...



# Designer Formlarının kullanımı

```
#include <QtGui>
#include "ui_designer.h"

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

    QWidget *widget = new QWidget;

    Ui::Form yeniPencere;
    yeniPencere.setupUi(widget);

    widget->show();

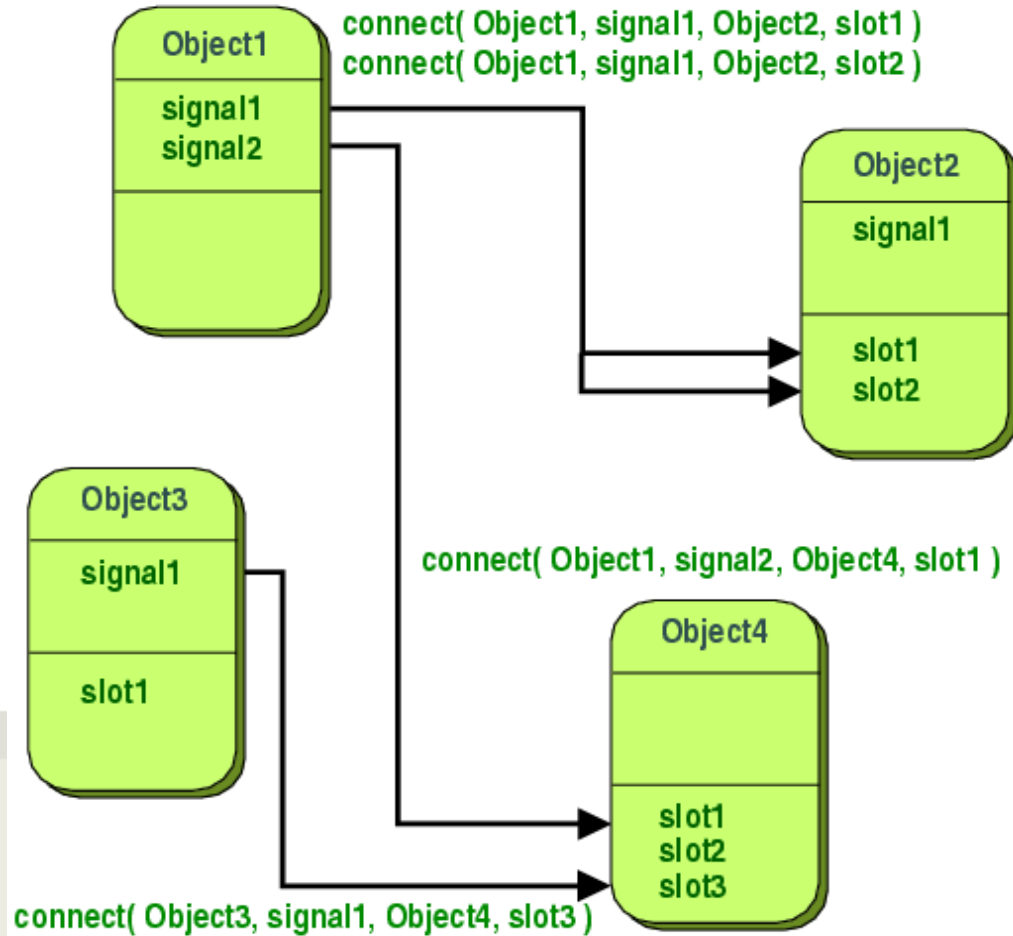
    return app.exec();
}

// FORMS += designer.ui
```



# Sinyal ve Slot Mekanizması

- Nesneler arası iletişim için kullanılıyor.
- Bir olay olduğunda, bir sinyal gönderilir
- Slotlar ise sinyallere cevap olarak çalıştırılan fonksiyonlardır.
- Qt widget lerinin sinyal ve slotlarının yanında kendi sinyal ve slotlarınızı da yazabilirsiniz.





# Sinyal ve Slotlar

- Sinyal gönderen bir sınıf sinyali alan olup olmadığını bilmez, eğer bir sinyali bir slotla bağlarsanız o sinyal verildiğinde o slot çalışır.
- Sinyal ve slotlar, parametre alabilir/gönderebilir.
- Sinyal ve slotlar GUI döngülerinden bağımsızdırlar, yani sinyal verildiği anda sinyalin bağlı olduğu slot çalıştırılır.
- Bu sayede en doğru şekliyle bilgi kapsülleme yapılmış olur ve nesnelerin bir yazılım bileşeni olarak kullanılması sağlanır.
- QObject (veya alt sınıflarından ) sınıfından türetilmiş her sınıf sinyal ve slot kullanabilir



# Sinyal Slot Örnek

```
#include <QObject>
```

```
class Counter : public QObject  
{  
    Q_OBJECT
```

```
public:
```

```
    Counter() { x = 0; }  
    int value() const { return x; }
```

```
public slots:
```

```
    void setValue(int value);
```

```
signals:
```

```
    void valueChanged(int newValue);
```

```
private:
```

```
    int x;  
};
```



# Sinyal Slot Kullanım

```
void Counter::setValue(int value)
{
    if (value != x) {
        x = value;
        emit valueChanged(value);
    }
}
```

// Örnek kullanım:

```
Counter a, b;
QObject::connect(&a, SIGNAL(valueChanged(int)),
                &b, SLOT(setValue(int)));

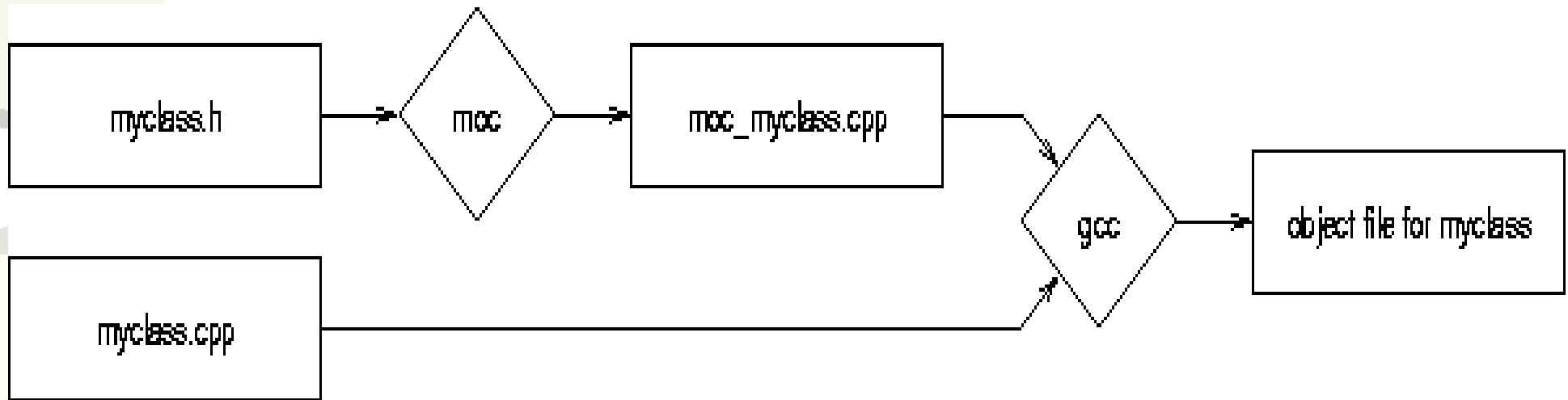
a.setValue(12);    // a.value() == 12, b.value() == 12
b.setValue(48);    // a.value() == 12, b.value() == 48
```

```
// QObject::disconnect()
```



# Meta Object Compiler

- C++ ön işlemcisi, sinyal, slot ve emit anahtar kelimelerini değiştirerek veya kaldırarak derleyiciye standart C++ sunar.
- moc'un sinyal ve slot içeren sınıf tanımlamalarına uygulanması ile, uygulama için gerekli diğer nesne dosyalarıyla derlenmesi ve bağlanması gereken C++ kaynak kodu üretilir.
- qmake kullanıyorsanız, projenizin Makefile dosyasına moc kuralları otomatik olarak eklenir.



# Olaylar ve Filtreler ( Events )

- Olaylar QEvent sınıfından türetilmiş nesnelere.
- Her QObject alt sınıfı olayları alabilir ve işleyebilir.
- Bir olay olduğunda Qt QEvent alt sınıflarından uygun olan bir nesne oluşturur ve bu nesneyi QObject'ın event fonksiyonunu çağırarak devreder.
- QResizeEvent, QPaintEvent, QMouseEvent, QKeyEvent, QCloseEvent ...
  - ◆ QEvent::Move
  - ◆ QEvent::MouseMove
  - ◆ QEvent::LanguageChange
  - ◆ QEvent::KeyRelease
  - ◆ QEvent::KeyPress ...
    - void QObject::mousePressEvent(QMouseEvent \*event)



# Event Handlers

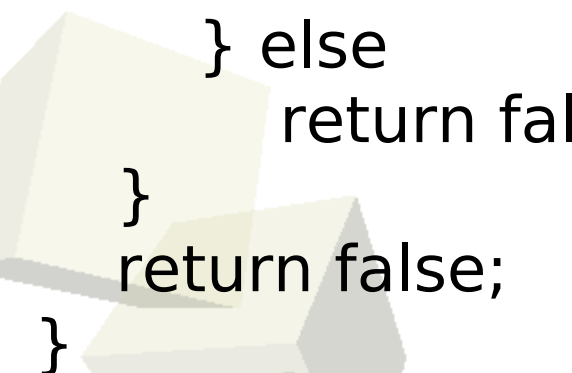
- Normalde bir olayı bir virtual fonksiyon çağırarak geçiririz. Mesela `QPaintEvent`, `QWidget::paintEvent()` çağrılarak geçirilir.
- Özel olarak ele almak için ise, bir `QCheckBox` için örneğin:

```
void BirCheckBoxSinifi::mousePressEvent(QMouseEvent
*event)
{
    if (event->button() == Qt::LeftButton)
        // sol tiklayınca bunu yap
    } else {
        // diğer butonlarda bunu yap
        QCheckBox::mousePressEvent(event); // Olayı yukarı
geçir
    }
}
```



# Filtre yapmak ve olay yollamak

```
bool FilterObject::eventFilter(QObject *object, QEvent *event)
{
    if (object == target && event->type() ==
        QEvent::KeyPress) {
        QKeyEvent *keyEvent = static_cast<QKeyEvent
*>(event);
        if (keyEvent->key() == Qt::Key_Tab) {
            return true;
        } else
            return false;
    }
    return false;
}
```





# Qt ile Programlama

İşbaran Akçayır  
<http://ish.kodzilla.org>

