Code less.
Create more.
Deploy everywhere.

Creación de nuestros componentes

- Debemos crear nuestra clase que herede de algún componente, por ejemplo si queremos hacer una ventana podemos heredar de QWidget.
- Debemos agregar la palabra clave Q_OBJECT con ella le indicamos a Qt que es un componente Qt.
- Con las palabras claves slots y signal, indico los slots y señales del componente.

Creación de nuestros componentes

```
class FindDialog : public QDialog
{
Q_OBJECT
public:
…
signals:
void findNext(const QString &str, Qt::CaseSensitivity cs);
void findPrevious(const QString &str, Qt::CaseSensitivity cs);
private slots:
void findClicked();
void enableFindButton(const QString &text);
...
```

Qt: Signals y Slots

El mecanismo de signals y slots es fundamental para qt. Los Slots son muy similares a las funciones comunes, pueden ser virtuales, sobrecargados, privados, protegidos y públicos. Además pueden ser invocados como cualquier otra función de c++ y sus parámetros pueden ser de cualquier tipo. La diferencia con las funciones es que los Slots pueden ser invocados automáticamente desde una señal.

Qt: Signals y Slots

La conexión entre signal y slot es así:
connect(sender, SIGNAL(signal), receiver, SLOT(slot));

Donde sender y receiver son objetos de tipo QObjects, además signal y slot se pasan como parámetro sin parámetros, solo el nombre de la función.

Relación entre Signals y Slots

- Una signal puede llamar a diferentes slots:
  - connect(slider, SIGNAL(valueChanged(int)), spinBox, SLOT(setValue(int)));
  - connect(slider, SIGNAL(valueChanged(int)), this, SLOT(updateStatusBarIndicator(int)));
- Un slot es llamado por diferentes signal:
  - connect(lcd, SIGNAL(overflow()), this, SLOT(handleMathError()));
  - connect(calculator, SIGNAL(divisionByZero()), this, SLOT(handleMathError()));
- Una signal puede conectar otra signal:
  - connect(lineEdit, SIGNAL(textChanged(const QString &)), this, SIGNAL(updateRecord(const QString &)));
- Una conexión puede ser removida:
  - disconnect(lcd, SIGNAL(overflow()), this, SLOT(handleMathError()));

Qt: Signals y Slots

- Algo **muy importante** es que la signal y el slot o signal a la que es conectada debe tener la misma cantidad de parámetros y del mismo tipo. Si el slot o signal a la que es conectada tiene más parámetros, simplemente sera ignorados. **Si los parámetros son de diferente tipo qt lanzara una advertencia cuando se compila**

Qt Designer

- Qt designer es una herramienta pensada para el diseño de interfaces de forma cómoda y fácil.
- Una de las ventajas de la utilización de Qt Designer es que permite a los programadores una gran libertad para modificar sus diseños sin ser forzados a cambiar su código fuente. Al desarrollar escribiendo solo código C++, los cambios en el diseño pueden llevar mucho tiempo. Con Qt Designer, no se pierde tiempo, simplemente regenera el código fuente que ha cambiado.
- Las Interfaz de usuario (ventanas) se guarda en un archivo .ui (un formato de archivo basado en XML), mientras que la funcionalidad personalizada se implementa mediante subclases.

Qt Designer

- Supongamos que debamos acceder a cualquier componente que hemos diseñado desde el diseñar. Debemos llamarlos desde un objeto generado, que se construye en el constructor y es de igual tipo que la ventana:

```
MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}
MainWindow::~MainWindow()
{
    delete ui;
}
void MainWindow::on_pushButton_clicked()
{
    ui->pushButton->setText("Chau!!");
}
```

- En el ejemplo para acceder a pushButton tuvimos que utilizar el objeto ui que es la ventana.