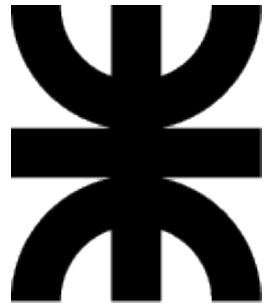
Informática II

Programación gráfica con Qt QtCreator

Gonzalo F. Perez Paina

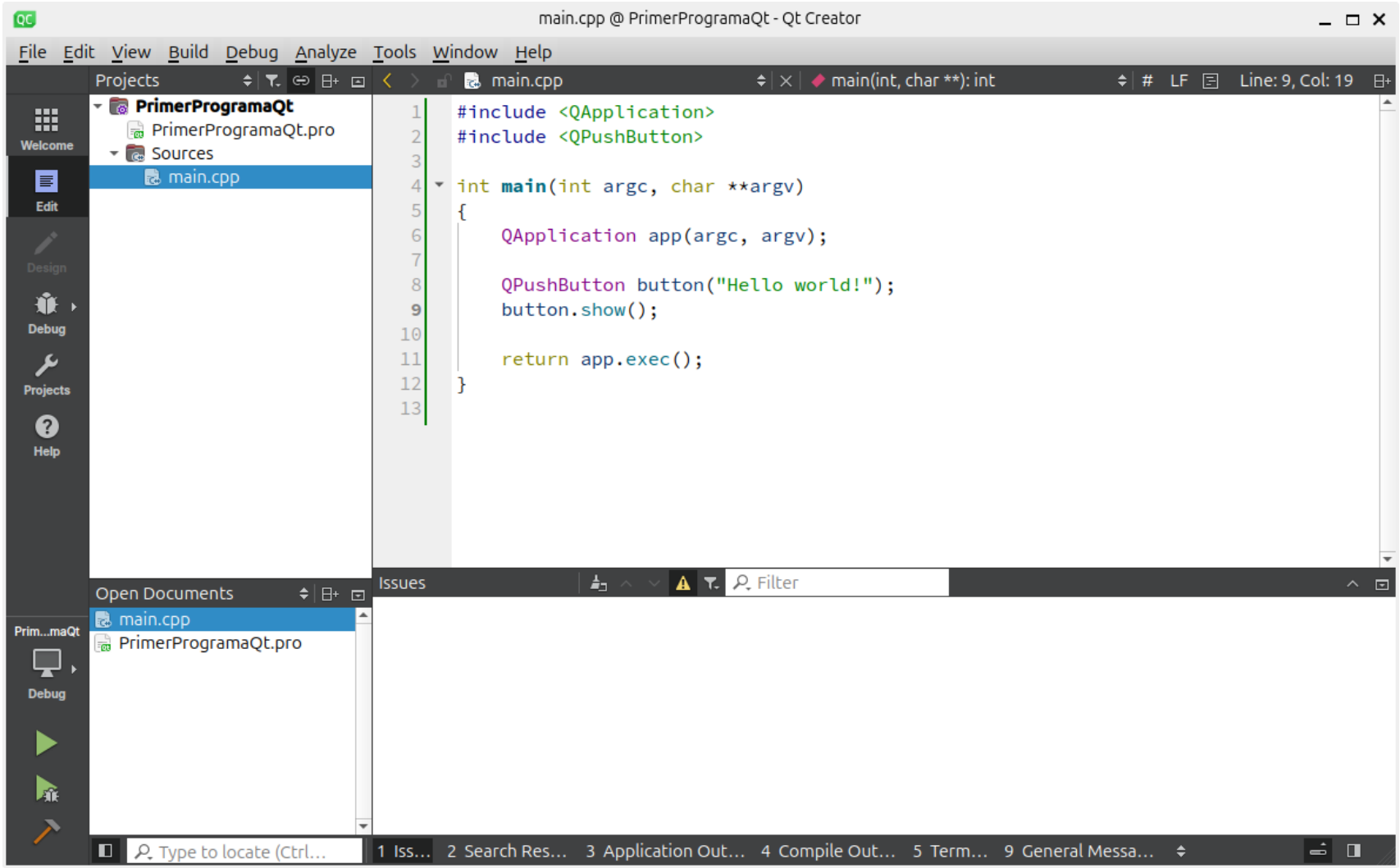


Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2024 –

Programando con QtCreator

- IDE para C++ que resulta adecuado para desarrollar aplicaciones Qt.
- Proporciona un navegador de documentos y el Designer ("diseñador") que facilita la creación de ventanas.



Programando con QtCreator

1. Menú "File → New file or project
   - Seleccionar "Aplication (Qt)" y "Qt Widget Application".

Programando con QtCreator

1. Menú "File → New file or project
   - Seleccionar "Aplication (Qt)" y "Qt Widget Application".
2. Elegir "CMake" como "Build system"

Programando con QtCreator

1. Menú "File → New file or project
   - Seleccionar "Aplication (Qt)" y "Qt Widget Application".
2. Elegir "CMake" como "Build system"
3. Menú "File → New file or project → C++ → C++ Source file"

```
#include "mainwindow.h"
#include <QApplication>

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

    MainWindow window;
    window.show();

    return app.exec();
}
```

Programando con QtCreator

1. Menú "File → New file or project
   - Seleccionar "Aplication (Qt)" y "Qt Widget Application".
2. Elegir "CMake" como "Build system"
3. Menú "File → New file or project → C++ → C++ Source file"

```
#include "mainwindow.h"
#include <QApplication>

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

    MainWindow window;
    window.show();

    return app.exec();
}
```

4. Construir y ejecutar con el botón verde (`Ctrl+R`)

Programando con QtCreator

- Los objetos `Qt` tienen muchos atributos que se pueden modificar utilizando funciones miembros getters y setters.
- Si un atributo tiene el nombre `foo` las funciones getter y setter asociadas tendrán el siguiente prototipo:

```
T foo() const;
void setFoo(const T);
```

Programando con QtCreator

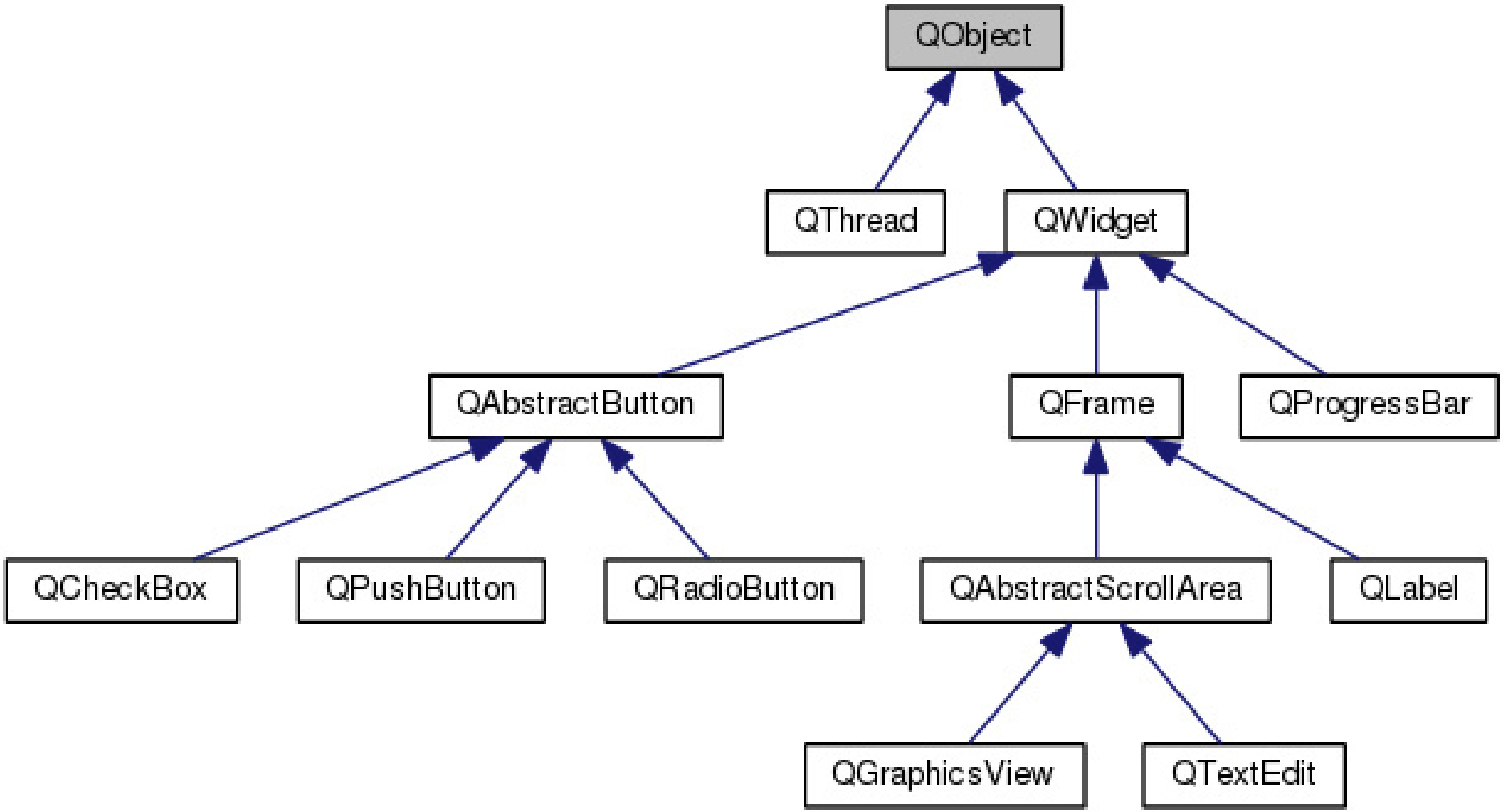
- Los objetos `Qt` tienen muchos atributos que se pueden modificar utilizando funciones miembros getters y setters.
- Si un atributo tiene el nombre `foo` las funciones getter y setter asociadas tendrán el siguiente prototipo:

```
T foo() const;
void setFoo(const T);
```

- Qt extiende este sistema de atributos y funciones setters y getters a algo llamado *propiedad*.
- Se utilizará “atributo” o “propiedad” de forma indistinta.

Jerarquía de clases

Qt utiliza ampliamente la herencia, especialmente en el módulo Widgets.



Jerarquía de clases

- `QObject` es la clase más básica de Qt.

Jerarquía de clases

- `QObject` es la clase más básica de Qt.
- La mayoría de las clases de Qt heredan de esta clase.

Jerarquía de clases

- `QObject` es la clase más básica de Qt.
- La mayoría de las clases de Qt heredan de esta clase.
- `QObject` proporciona algunas capacidades como:
  - **object name:** puede establecer un nombre, como una cadena, para un objeto y buscar objetos por nombres.
  - **parenting system**: (más adelante)
  - **signal and slots**: (más adelante)

Jerarquía de clases

- `QObject` es la clase más básica de Qt.
- La mayoría de las clases de Qt heredan de esta clase.
- `QObject` proporciona algunas capacidades como:
  - **object name:** puede establecer un nombre, como una cadena, para un objeto y buscar objetos por nombres.
  - **parenting system**: (más adelante)
  - **signal and slots**: (más adelante)
- Los widgets pueden responder a eventos y utilizar el sistema de parentesco y señales y slots.

Jerarquía de clases

- `QObject` es la clase más básica de Qt.
- La mayoría de las clases de Qt heredan de esta clase.
- `QObject` proporciona algunas capacidades como:
  - **object name:** puede establecer un nombre, como una cadena, para un objeto y buscar objetos por nombres.
  - **parenting system**: (más adelante)
  - **signal and slots**: (más adelante)
- Los widgets pueden responder a eventos y utilizar el sistema de parentesco y señales y slots.
- El widget más básico es el `QWidget`.

Jerarquía de clases

- `QObject` es la clase más básica de Qt.
- La mayoría de las clases de Qt heredan de esta clase.
- `QObject` proporciona algunas capacidades como:
  - **object name:** puede establecer un nombre, como una cadena, para un objeto y buscar objetos por nombres.
  - **parenting system**: (más adelante)
  - **signal and slots**: (más adelante)
- Los widgets pueden responder a eventos y utilizar el sistema de parentesco y señales y slots.
- El widget más básico es el `QWidget`.
- `QWidget` contiene la mayoría de las propiedades que se utilizan para describir una ventana o un widget, como la posición y el tamaño, el cursor del mouse, información sobre herramientas, etc.

Sistema de parentesco

Ver ejemplos: `ButtonInWindget` y `TwoButtons`

Sistema de parentesco

Ver ejemplos: `ButtonInWindget` y `TwoButtons`

- El sistema de parentesco es una forma conveniente de tratar con los objetos en Qt, especialmente los widgets.

Sistema de parentesco

Ver ejemplos: `ButtonInWindget` y `TwoButtons`

- El sistema de parentesco es una forma conveniente de tratar con los objetos en Qt, especialmente los widgets.
- Cualquier objeto que herede de `QObject` puede tener padre e hijos. Este árbol de jerarquía facilita muchas cosas:

Sistema de parentesco

Ver ejemplos: `ButtonInWindget` y `TwoButtons`

- El sistema de parentesco es una forma conveniente de tratar con los objetos en Qt, especialmente los widgets.
- Cualquier objeto que herede de `QObject` puede tener padre e hijos. Este árbol de jerarquía facilita muchas cosas:
  - Cuando se destruye un objeto, también se destruyen todos sus hijos.

Sistema de parentesco

Ver ejemplos: `ButtonInWindget` y `TwoButtons`

- El sistema de parentesco es una forma conveniente de tratar con los objetos en Qt, especialmente los widgets.
- Cualquier objeto que herede de `QObject` puede tener padre e hijos. Este árbol de jerarquía facilita muchas cosas:
  - Cuando se destruye un objeto, también se destruyen todos sus hijos.
  - Todos los `QObjects` tienen métodos `findChild` y `findChildren` que se pueden usar para buscar elementos secundarios de un objeto determinado.

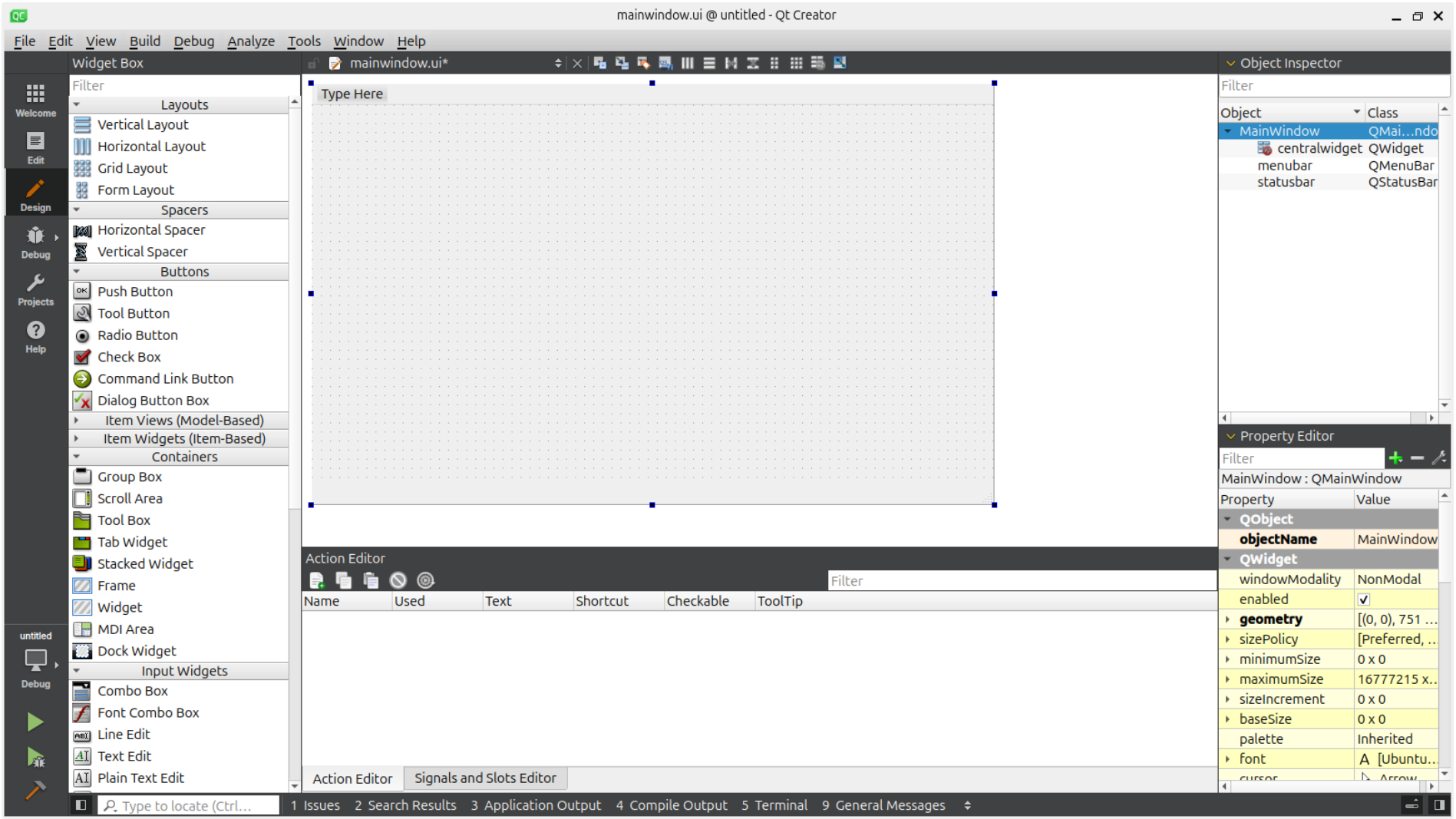
Sistema de parentesco

Ver ejemplos: `ButtonInWindget` y `TwoButtons`

- El sistema de parentesco es una forma conveniente de tratar con los objetos en Qt, especialmente los widgets.
- Cualquier objeto que herede de `QObject` puede tener padre e hijos. Este árbol de jerarquía facilita muchas cosas:
  - Cuando se destruye un objeto, también se destruyen todos sus hijos.
  - Todos los `QObjects` tienen métodos `findChild` y `findChildren` que se pueden usar para buscar elementos secundarios de un objeto determinado.
  - Los widgets secundarios en un `QWidget` aparecen automáticamente dentro del widget principal.

Qt Designer

- Herramienta para diseñar GUI de usuario basadas en widgets



Qt Designer

- El editor de UI se llama Qt Designer.

Qt Designer

- El editor de UI se llama Qt Designer.
- Qt Designer es la herramienta para diseñar la interfaz de usuario de su programa sin escribir ningún código.

Qt Designer

- El editor de UI se llama Qt Designer.
- Qt Designer es la herramienta para diseñar la interfaz de usuario de su programa sin escribir ningún código.
- Qt Designer utiliza un enfoque WYSIWYG (lo que ve es lo que obtiene): lo que diseñe con Qt Designer resultará exactamente igual cuando el programa se compile y ejecute.

Qt Designer

- El editor de UI se llama Qt Designer.
- Qt Designer es la herramienta para diseñar la interfaz de usuario de su programa sin escribir ningún código.
- Qt Designer utiliza un enfoque WYSIWYG (lo que ve es lo que obtiene): lo que diseñe con Qt Designer resultará exactamente igual cuando el programa se compile y ejecute.

Widgets (algunos ejemplos):

Layouts: Vertical, Horizontal, Grid, Form

Spaces: Horizontal Spacer, Vertical Spacer

Buttons: Push Button, Tool Button, Radio Button, Check Box, etc.

Containers: Group Box, Scroll Area, Widget, etc.

Input Widgets: Combo Box, Line Edit, Spin Box, etc.

Display Widgets: Label, Text Box, Progress Bar, etc.

Qt Designer – Ejemplo paso a paso

1. Colocar (arrastrar y soltar) un **PushButton** en **Form** (ventana en blanco).

Qt Designer – Ejemplo paso a paso

1. Colocar (arrastrar y soltar) un **PushButton** en **Form** (ventana en blanco).

2. Seleccionar el botón recién agregado y ver sus propiedades en el **Property Editor** (Panel de edición de propiedades).

Qt Designer – Ejemplo paso a paso

1. Colocar (arrastrar y soltar) un **PushButton** en **Form** (ventana en blanco).

2. Seleccionar el botón recién agregado y ver sus propiedades en el **Property Editor** (Panel de edición de propiedades).

3. Estas propiedades se pueden modificar desde el código fuente. Algunas propiedades se pueden modificar desde el **Form Editor**.
(Hacer doble-click al botón para modificar su texto y redimencionarlo arrastrando alguno de sus bordes)

Qt Designer – Ejemplo paso a paso

1. Colocar (arrastrar y soltar) un **PushButton** en **Form** (ventana en blanco).

2. Seleccionar el botón recién agregado y ver sus propiedades en el **Property Editor** (Panel de edición de propiedades).

3. Estas propiedades se pueden modificar desde el código fuente. Algunas propiedades se pueden modificar desde el **Form Editor**.
(Hacer doble-click al botón para modificar su texto y redimencionarlo arrastrando alguno de sus bordes)

4. Colocar un **Horizontal Layout** y luego arrastar el botón dentro del mismo.

Qt Designer – Ejemplo paso a paso

1. Colocar (arrastrar y soltar) un **PushButton** en **Form** (ventana en blanco).

2. Seleccionar el botón recién agregado y ver sus propiedades en el **Property Editor** (Panel de edición de propiedades).

3. Estas propiedades se pueden modificar desde el código fuente. Algunas propiedades se pueden modificar desde el **Form Editor**.
   (Hacer doble-click al botón para modificar su texto y redimencionarlo arrastrando alguno de sus bordes)

4. Colocar un **Horizontal Layout** y luego arrastar el botón dentro del mismo.

(Se puede probar el comportamiento de la ventana si tener que compilar el programa. Ir al menú Tool→Form Editor→Preview, o Alt+Shift+R)

Qt Designer – Ejemplo paso a paso

5. Por defecto, la ventana principal no maneja los efectos de layouts, por lo que los widgets permaneceran en los lugares colocados originalmente, aún cuando se redimensione la ventana, lo cual no resulta adecuado. Para activar los efectos del layout, hacer click derecho al **Form Editor** y elegir **Lay Out Vertically**. Ahora, el **Horizontal Layout** colocado anteriormente ocupará toda la ventana.

Qt Designer – Ejemplo paso a paso

5. Por defecto, la ventana principal no maneja los efectos de layouts, por lo que los widgets permaneceran en los lugares colocados originalmente, aún cuando se redimensione la ventana, lo cual no resulta adecuado. Para activar los efectos del layout, hacer click derecho al **Form Editor** y elegir **Lay Out Vertically**. Ahora, el **Horizontal Layout** colocado anteriormente ocupará toda la ventana.

6. Colocar un **Vertical Spacer** en la parte superior del layout que tiene el botón y luego colocar dos **Horizontal Spacers** a ambos lados del botón. Cambiar la propiedad del botón `minimumSize` a 120x40.

Qt Designer – Ejemplo paso a paso

5. Por defecto, la ventana principal no maneja los efectos de layouts, por lo que los widgets permaneceran en los lugares colocados originalmente, aún cuando se redimensione la ventana, lo cual no resulta adecuado. Para activar los efectos del layout, hacer click derecho al **Form Editor** y elegir **Lay Out Vertically**. Ahora, el **Horizontal Layout** colocado anteriormente ocupará toda la ventana.

6. Colocar un **Vertical Spacer** en la parte superior del layout que tiene el botón y luego colocar dos **Horizontal Spacers** a ambos lados del botón. Cambiar la propiedad del botón `minimumSize` a 120x40.

7. Agregar un **Form Layout** entre el botón y el espaciador vertical que está arriba y un espaciador vertical abajo del botón. El mismo tendrá un espesor muy pequeño debido al espaciador vertical, lo que puede complicar colocar algo dentro del **Form Layout**. Para resolver esto, temporalmente fijar la propiedad `layoutTopMargin` al valor 20 o uno mayor.

Qt Designer – Ejemplo paso a paso

8. Luego, arrastrar y soltar dos **Labels** en el lado izquierdo del **Form Layout** y dos **Line Edits** en su parte derecha. Hacer doble-click a los labels y cambiar el texto a `Usuario:` y `Clave:` respectivamente. Volver al valor cero la propiedad `layoutTopMargin` del **Form Layout**.

Qt Designer – Ejemplo paso a paso

8. Luego, arrastrar y soltar dos **Labels** en el lado izquierdo del **Form Layout** y dos **Line Edits** en su parte derecha. Hacer doble-click a los labels y cambiar el texto a `Usuario:` y `Clave:` respectivamente. Volver al valor cero la propiedad `layoutTopMargin` del **Form Layout**.

9. Poner un **Horizontal Layout** arriba del **Form Layout** y fijar `layoutTopMargin` y `layoutBottomMargin` a 20. Luego, arrastrar el **Form Layout** con todo su contenido dentro del **Horizontal Layout**. Luego, poner dos **Horizontal Spacers** a ambos lados del **Horizontal Layout** para centrarlo.

Qt Designer – Ejemplo paso a paso

8. Luego, arrastrar y soltar dos **Labels** en el lado izquierdo del **Form Layout** y dos **Line Edits** en su parte derecha. Hacer doble-click a los labels y cambiar el texto a `Usuario:` y `Clave:` respectivamente. Volver al valor cero la propiedad `layoutTopMargin` del **Form Layout**.

9. Poner un **Horizontal Layout** arriba del **Form Layout** y fijar `layoutTopMargin` y `layoutBottomMargin` a 20. Luego, arrastrar el **Form Layout** con todo su contenido dentro del **Horizontal Layout**. Luego, poner dos **Horizontal Spacers** a ambos lados del **Horizontal Layout** para centrarlo.

10. Modificar la propiedad `minimumSize` de los dos **Line Edit** a 150x25 y fijar las propiedades `layoutLeftMargin`, `layoutRightMargin`, `layoutTopMargin` y `layoutBottomMargin` del **Form Layout** a 25.

Qt Designer – Ejemplo paso a paso

8. Luego, arrastrar y soltar dos **Labels** en el lado izquierdo del **Form Layout** y dos **Line Edits** en su parte derecha. Hacer doble-click a los labels y cambiar el texto a `Usuario:` y `Clave:` respectivamente. Volver al valor cero la propiedad `layoutTopMargin` del **Form Layout**.

9. Poner un **Horizontal Layout** arriba del **Form Layout** y fijar `layoutTopMargin` y `layoutBottomMargin` a 20. Luego, arrastrar el **Form Layout** con todo su contenido dentro del **Horizontal Layout**. Luego, poner dos **Horizontal Spacers** a ambos lados del **Horizontal Layout** para centrarlo.

10. Modificar la propiedad `minimumSize` de los dos **Line Edit** a 150x25 y fijar las propiedades `layoutLeftMargin`, `layoutRightMargin`, `layoutTopMargin` y `layoutBottomMargin` del **Form Layout** a 25.

11. Dado que el **Push Button** está ahora demasiado alejado del **Form Layout**, fijar la propiedad `layoutBottomMargin` del **Horizontal Layout** a 0. Ajustar el tamaño del **Push Button** para alinearlo con el **Form Layout**. Fijar la propiedad `minimumSize` del **Push Button** a 260x35.