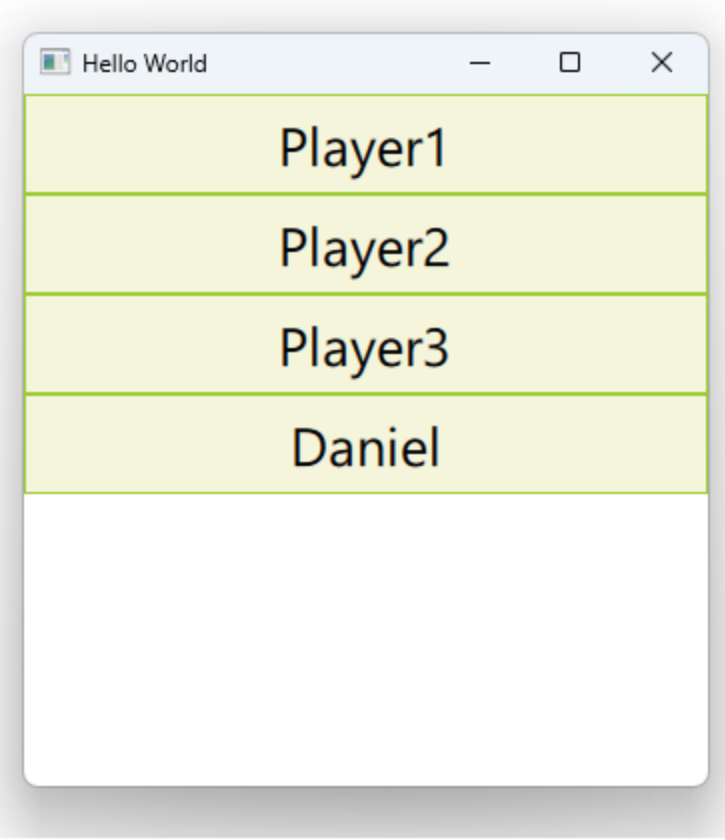
Notes to self

```
. Exploring object and list properties
    . Taking reference from the example :
      https://doc.qt.io/qt-6/qtqml-referenceexamples-properties-example.html
    . The difference from the Qt 5 course is that they changed the
      parameter types form functions taking a size from int to qsizetype

          . In Qt 6 they take qsizetype :
              .     static qsizetype playerCount(QQmlListProperty<Player>*);
                      static Player* player(QQmlListProperty<Player>*, qsizetype);

          . While they took int in Qt 5 :
              .    static int playerCount(QQmlListProperty<Player>*);
                      static Player* player(QQmlListProperty<Player>*, int);

    . The rest is the same as in Qt 5

    . Take explanations from the docs :
      https://doc.qt.io/qt-6/qtqml-cppintegration-exposecppattributes.html#properties-with-object-types

    . This explanation must not be left out :
      Properties with Object-List Types
          Properties containing lists of QObject-derived types can also be exposed to QML.
          For this purpose, however, one should use QQmlListProperty rather than QList<T>
          as the property type. This is because QList is not a QObject-derived type,
          and so cannot provide the necessary QML property characteristics through
          the Qt meta object system, such as signal notifications when a list is modified.

    . Combine it with the Qt 5 course material
```

Object and List Properties



FootBallTeam

```
FootBallTeam {
    id : team1
    title: "Rayon Sports"
    coatch: "Coatch Name"
    captain: Player{
        name: "Captain"
        position: "Middle Field"
        playing: true
    }
    players: [
        Player{
            name: "Player1"
            position: "Middle Field"
            playing: true
        },
        Player{...}
        //...
    ]
}
```

Using the data

```
ListView {
    anchors.fill: parent
    model : team1.players
    delegate: Rectangle{
        width: parent.width
        height: 50
        border.width: 1
        border.color: "yellowgreen"
        color: "beige"

        Text {
            anchors.centerIn: parent
            text : name
            font.pointSize: 20
        }
    }
}
```

Player

```
class Player : public QObject
{
    Q_OBJECT
    Q_PROPERTY(QString name READ name WRITE setName NOTIFY nameChanged)
    Q_PROPERTY(bool playing READ playing WRITE setPlaying NOTIFY playingChanged)
    Q_PROPERTY(QString position READ position WRITE setPosition NOTIFY positionChanged)
public:
    explicit Player(QObject *parent = nullptr);
    QString name() const;
    bool playing() const;
    QString position() const;
    void setName(QString name);
    void setPlaying(bool playing);
    void setPosition(QString position);
signals:
    void nameChanged(QString name);
    void playingChanged(bool playing);
    void positionChanged(QString position);
private :
    QString m_name;
    bool m_playing;
    QString m_position;
};
```

FootBallTeam

```
class FootBallTeam : public QObject
{
    Q_OBJECT

    Q_PROPERTY(QString title READ title WRITE setTitle NOTIFY titleChanged)
    Q_PROPERTY(QString coatch READ coatch WRITE setCoatch NOTIFY coatchChanged)
    Q_PROPERTY(Player * captain READ captain WRITE setCaptain NOTIFY captainChanged) // Object property
    Q_PROPERTY(QQmlListProperty<Player> players READ players  NOTIFY playersChanged) // List property

public:
    explicit FootBallTeam(QObject *parent = nullptr);
    /* ... */
private:
    //Callback Methods
    static void appendPlayer(QQmlListProperty<Player>*, Player*);
    static qsizetype playerCount(QQmlListProperty<Player>*);
    static Player* player(QQmlListProperty<Player>*, qsizetype);
    static void clearPlayers(QQmlListProperty<Player>*);

    QString m_title;
    QString m_coatch;
    Player * m_captain;
    QVector<Player*> m_players;
};
```

The methods

```
void FootBallTeam::appendPlayer(QQmlListProperty<Player> * list, Player * player)
{
    reinterpret_cast<FootBallTeam*>(list->data)->appendPlayerCustom(player);
}

qsizetype FootBallTeam::playerCount(QQmlListProperty<Player> * list)
{
    return reinterpret_cast<FootBallTeam*>(list->data)->playerCountCustom();
}

Player *FootBallTeam::player(QQmlListProperty<Player> * list, qsizetype index)
{
    return reinterpret_cast<FootBallTeam*>(list->data)->playerCustom(index);
}

void FootBallTeam::clearPlayers(QQmlListProperty<Player> * list)
{
    reinterpret_cast<FootBallTeam*>(list->data)->clearPlayersCustom();
}
```

players()

```
QQmlListProperty<Player> FootBallTeam::players()
{
    return QQmlListProperty<Player>(this,this,&FootBallTeam::appendPlayer,
                                    &FootBallTeam::playerCount,
                                    &FootBallTeam::player,
                                    &FootBallTeam::clearPlayers);
}
```

```
//Register Types
qmlRegisterType<Player>("com.blikoon.Football", 1,0, "Player");
qmlRegisterType<FootBallTeam>("com.blikoon.Football", 1,0, "FootBallTeam");
```

Use in qml

```
import com.blikoon.Football 1.0
FootBallTeam {
    id : team1
    title: "Rayon Sports"
    coatch: "Coatch Name"
    captain: Player{
        name: "Captain"
        position: "Middle Field"
        playing: true
    }

    players: [

        Player{
            name: "Player1"
            position: "Middle Field"
            playing: true
        },
        Player{...}
    ]
}
```