Notes lecture 11

*Model:*

* *QAbstractItemModel* = interface used by views and delegates to access data
  + All item models are based on this abstract class.
  + can be used with views that represent data in the form of tables, lists, and trees
* *QAbstractListModel* and *QAbstractTableModel* more appropriate for table-like data structures
* Predefined models:

|  |  |
| --- | --- |
| QStringListModel | stores a list of strings |
| QStandardItemModel | stores arbitrary hierarchical data |
| QFileSystemModel | encapsulates the local file system |
| QSqlQueryModel | encapsulates an SQL result set |
| QSqlTableModel | encapsulates an SQL table |
| QSortFilterProxyModel | sorts and/or filters another model |

* Example: QAbstractTableModel
  + Provide implementation for at least:
    - *rowCount()*
    - *columnCount()*
    - *data()*
  + *QModelIndex* - used to locate data in a model

A screenshot of a computer code

Description automatically generated

* Controlling the text appearance with roles

A white sheet with black text

Description automatically generated

* *headerData()* function must be implemented in the model if using a table/tree

QVariant headerData(

int section,

Qt::Orientation orientation,

int role = Qt::DisplayRole

) const override;

*View:*

* *QAbstractItemView* = abstract class for views
* Predefined views:

|  |  |
| --- | --- |
| QListView | Display a list of items |
| QTableView | Display in a table. |
| QTreeView | Display in a hierarchical list |

*Filtering and sorting:*

* *QSortFilterProxyModel* class 🡪 support for sorting and filtering data
* Usage:
  + Create a QSortFilterProxyModel object
  + use the *setSourceModel()* and set the QSortFilterProxyModel on the view.

*Delegates:*

* standard interface: *QAbstractItemDelegate*
* *QStyledItemDelegate* should be used as base class when implementing custom delegates