Qt Design Patterns

Sources
https://sourcemaking.com/design_patterns

Simple solutions to common problems
Can be C++ or Qt specific
Just because a pattern exists does not mean you NEED to use it
design patterns become code reuse

- Classes – blue print for an object
  - SOLID - https://en.wikipedia.org/wiki/SOLID
    - S = Single Responsibility Principle
    - O = Open-Closed Principle
    - L = Liskov Substitution Principle
    - I = Interface Segregation Principle
    - D = Dependency Inversion/Injection
  - Interfaces – contract between objects
  - Inheritance – re-susable code
  - Interface vs Inheritance
- Memory – where these objects are stored
  - Stack
  - Heap
  - Memory leaks
  - Smart Pointers
  - Shared Pointer
  - Scoped pointer
  - Weak Pointer
  - D-Pointers https://wiki.qt.io/D-Pointer
  - Creating a custom pointer type, maybe a pointer to a file stream
- Signals and slots – communicating between objects
  - connecting
  - disconnecting
  - dynamically creating
  - sender function
  - Connection types (auto, direct, qued)
  - Thread example – why a direct connection fails, used qued
  - Between app and lib
  - Remote Objects - https://doc.qt.io/qt-5/qtremoteobjects-index.html
- Threads – where the objects execute
  - QThread
  - QThreadPool
  - QtConCurrent
  - Example – Folder size on thread, who a Qt Widgets
  - GUI Thread locking, make an example, use QtconCurrent to get around it
- Generics - polymorphism
  - Understanding templates
  - Function templates
  - Class templates

  - Qhash, Qlist
  - Lists of objects
  - Lists of pointers
  - Lists of smart pointers
- Traditional Patterns - Creational – creating objects
  - Abstract Factory
  - Builder Factory
  - Object Pool
  - Prototype
  - Singleton
- Traditional Patterns - Structural – why the objects exist
  - Adapter
  - Bridge
  - Composite
  - Decorator
  - Facade
  - Flyweight
  - Private Class Data
  - Proxy
- Traditional Patterns - Behavioral – how the objects behave
  - Chain of responsibility
  - Command
  - Interpreter
  - Iterator
  - Mediator
  - Memento
  - Null Object
  - Observer
  - State
  - Strategy
  - Template Method
  - Visitor
- Qmake
  - Basics
  - Sub Dirs
  - Ordered Sub Dirs
  - Adding libs
  - OS Specific
  - Running commands (bat / sh /exe)
- Cmake
  - Basics
  - Sub Dirs
  - Ordered Sub Dirs
  - Adding libs
  - OS Specific
  - Running commands (bat / sh /exe)
- Libraries
  - SubDirs Ordered Project type
  - Creating a lib

    - Re-using libs
    - 3rd Party libs
- Plugins
    - Creating a plugin
    - netwotk in plugin
    - gui in plugin
    - extending app functionality – command line app – add commands
- File System
    - Text Encoding
    - Streams - versions
    - Directory Recursion
    - Disk Info
    - File Formats
    - File Format Versioning
    - Object Serialization
    - Tar files
    - Temp files
    - Settings
    - Storing passwords and sensitive info
    - Saving windows settings
- Sockets
    - Client models
    - Server models
    - Protocols
    - Versioning
    - Session State
    - Session Client
    - Session Server (ThreadPool Server with Signals and Slots)
    - Creating a simple protocol
    - Creating an advanced protocol
    - Stress testing with seige
    - Building a stress tester
- Hash
    - What is a hash
    - Why use a hash
    - Collisions and birthday attacks
    - Hash types
    - Creating a hash
    - Comparing hashes
- Compression
    - qCompress
    - Gzip
    - Zip using zLib
- Encryption – use OpenSSL
    - Asymetric
        - public keys
        - private keys
    - Symetric
        - passwords, passphrases and keys

    - Block Cyphers
    - AES 256 using OpenSSL
    - Storing passwords
    - Encrypting a file with AES
    - Decrypting a file with AES
    - Encrypting a file with RSA – public vs private keys
    - Decrypting a file with RSA – public vs private keys
- Databases
    - MySQL – the damn plugin and client
    - SQLite
    - XML
    - JSON
- Models and Views
    - QStringListModel
    - QStandardItemModel
    - QfileSystemModel
    - QsqlQueryModel
    - QsqlTableModel
    - QSqlRelationalTableModel
    - Creating custom models
        - QabstractItemModel
        - QabstractListModel
        - QAbstractTableModel
    - Loading custom models
    - Editing from views
    - Paging - https://github.com/voidrealms/PagedModel
- Cross Platform
    - Determining the platform
    - Packaging
    - Common issues
- Background services
    - Windows service
    - Linux service
    - Mac service
- Scalability
    - Screen resolutions
    - screen orientation
    - Resizing windows
    - Min and Max sizes
- OS Specific
    - Linux Desktop integration
    - Windows registry
    - Windows UAC
    - Linux – root
    - Mac – admins
    - Android – permissions
    - processes – bash / cmd - qprocess
        - pinging a server
        - ns lookup

- list processes