\chapter{Alex} \label{chp:alex}

Klasse io_data

Die Klasse „io_data“ dient der strukturierten Konvertierung, Ein- und Ausgabe der Patientendaten. Im Konstruktor werden Membervariablen definiert, die patientenbezogene Informationen, wie ID, Name, Geburtsdatum, Geschlecht, Kontaktdaten und medizinische Details umfassen.

Die Implementierung der Klasse enthält eine Methode zur Altersberechnung anhand des Geburtsdatums und Funktionen zur bidirektionalen Konvertierung des Datums zwischen den Formaten QDate und QString.

Des Weiteren verwaltet Sie die Datenflüsse für der Datenein- und -ausgabe. Dies erfolgt mithilfe der Klasse „database“, die die Datenbankanbindung realisiert und die Patientendaten in der Datenbank abspeichert, bzw. liest. Zudem sind die Funktionalitäten zum Einlesen von Patientendaten aus CSV-Dateien und Exportierung der gespeicherten Patientendaten in einer CSV-Datei enthalten.

Diese Klasse **„io_data“** ist zentraler Bestandteil des Projektes und enthält folgende Methoden:

- **int returnAge()**
  Berechnet das Alter basierend auf dem Geburtsdatum des Patienten

- **static QDate convertQStringToQDate(const QString datumString)**
  Konvertiert ein Datum vom QString-Format in ein QDate-Objekt

- **static QString convertQDateToQString(const QDate datum)**
  Wandelt ein QDate-Objekt in eine QString um.

- **static void CSVeinlesen(QString pfad, Database &database)**
  Liest Patientendaten aus einer CSV-Datei ein und speichert sie in die Datenbank

- **static void CSVerstellen(QString pfad, Database &database)**
  Erstellt eine CSV-Datei aus den in der Datenbank gespeicherten Patientendaten

int io_data::returnAge()

Die Methode returnAge() berechnet das Alter eines Patienten basierend auf dem Geburtsdatum und dem aktuellen Datum und gibt dieses als int zurück.

Sie wandelt den QString der Membervariable „geburt“ über einen Standardstring in einen stream um und zerlegt diesen an den enthaltenen Punkten in drei Variablen „int tag“, „int monat“ und „int jahr“.

Anschließend wird die Systemzeit geladen und durch Subtraktion des Jahres das Alter berechnet. Zudem wird über eine if-Abfrage das Alter korrigiert, falls das aktuelle Datum vor dem Geburtstag liegt um das korrekte tagesaktuelle Alter zurückzugeben.

QDate io_data::convertQStringToQDate(const QString datumString)

convertQStringToQDate() prüft den übergebenen String auf Gültigkeit (Format „dd.MM.yyyy“) und gibt bei Erfolg das Datum als QDate zurück. Ansonsten wird ein ungültiges QDate zusammen mit einer qWarning zurückgegeben.

QString io_data::convertQDateToQString(const QDate datum)

convertQDateToQString() ist die inverse Funktion zu convertQStringToQDate() und wandelt das QDate in einen QString im Format „dd.MM.yyyy“ um. Dieses wird zur z. B. zur Speicherung in der Datenbank nach dem Bearbeiten eines Datensatzes benötigt.

void io_data::CSVeinlesen(QString pfad,Database &database)

Die Methode CSVeinlesen() ist mit Try-Catch-Blöcken realisiert, um die Programmstabilität sicherzustellen. Sie öffnet die ausgewählte CSV-Datei, liest dessen Patientendaten zeilenweise ein und validiert diese mittels Regex. Anschließend werden die validierten Daten in der Datenbank abgespeichert. Dabei wird die erste Zeile, die die Spaltenüberschrift enthält übersprungen.

Damit nur vollständige und korrekte Datensätze in der Datenbank abgelegt werden, wird nach dem erfolgreichen Einlesen jeder Zeile der Patientendatensatz mit regex validiert, und die Anzahl an Werten überprüft, bevor er über die Klasse „database“ mit der Methode „insertPatient“ der Datenbank hinzugefügt wird. Schlägt hierbei regex fehl oder stimmt die Anzahl der eingelesenen Werte nicht, wird eine Catch ausgelöst und ein qDebug ausgegeben.

Nach dem vollständigen Einlesen der Datei, wird diese wieder geschlossen, um Speicherzugriffsproblemen, Programmstabilitätseinbrüche und Bugs vorzubeugen.

void io_data::CSVerstellen(QString pfad, Database &database)

Die Methode CSVerstellen() exportiert die Patientendaten aus der Datenbank in eine CSV-Datei. Sie erstellt (bzw. überschreibt) die CSV-Datei und fügt in dessen erste Zeile die Spaltenüberschrift der Patientendaten ein. Anschließend werden die Patientendatensätze der Datenbank nacheinander durchgegangen und in die CSV-Datei geschrieben, bevor die Datei wieder geschlossen wird.

Um die Stabilität des Programms zu gewährleisten, ist diese Methode ebenfalls in einen Try-Catch-Block eingebettet. Dies stellt sicher, dass mögliche Fehler durch Dateioperationen auf Systemebene, die durch Systeminteraktionen oder das Verhalten anderer im Hintergrund laufender Anwendungen entstehen können, abgefangen und behandelt werden.

mainwindow.cpp

connect(ui->data_table, &QTableWidget::itemClicked, this, &MainWindow::on_data_table_itemClicked) und connect(ui->data_table->selectionModel(), &QItemSelectionModel::currentRowChanged, this, &MainWindow::on_data_table_rowSelected) verbinden und aktualisieren das TextEdit Feld im Hauptfenster rechts unten mit den gesammelten Daten des ausgewählten Patientendatensatzes.

In der Methode on_speicher_btn_clicked() wird über den QFileDialog der Dateipfad für die speichern Funktionalität eingelesen und an die oben erwähnte Methode CSVerstellen() der Klasse „io_data“ übergeben. Hierbei wird dem Benutzer Rückmeldung gegeben, ob die Aktion erfolgreich war. Dasselbe gilt auch für die on_open_btn_clicked() Methode, die CSVeinlesen() ausführt.

on_data_table_itemClicked() wählt einen angezeigten Datensatz aus und on_suche_btn_clicked() aktualisiert die angezeigten Patienten. Die Methode on_suche_btn_clicked() wurde als public deklariert, damit auch nach dem Bearbeiten eines Datensatzes in einem separaten Fenster die Anzeige im Hauptfenster automatisch in der Tabelle und im TextEdit Feld aktualisiert wird.