HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES
INSTITUT INFORMATIK

28.03.2022

Klausur

MMI und GUI-Programmierung

Wintersemester 2022/2023

Studiengänge: AI, MTI, WI

Prüfer: Prof. Dr. Malte Weiß

Persönliche Angaben

Nachname	Vorname	Matrikelnummer

Bewertung

Frage	Punkte	Erreicht
1	12	
2	12	
3	6	
4	17	

Frage	Punkte	Erreicht
5	21	
6	20	
7	12	
Gesamt:	100	

Ablauf der Prüfung

- **Zeit für die Lösung dieses Aufgabenblatts**: 120 Minuten.
- **Erlaubte Hilfsmittel**: Cheat Sheet am Ende der Klausur. Handgeschriebener, ein- oder beidseitig beschriebener Notizzettel im DIN-A4-Format.
- **Nicht erlaubte Hilfsmittel**: Mobiltelefon, Kommunikation (E-Mail, Web, SMS, etc.). Die Verwendung eines nicht erlaubten Hilfsmittels wird als Täuschungsversuch gewertet.
- **Stift und Papier**: Schreiben Sie ausschließlich auf dem Papier der Aufgabenblätter. Sollten die Zwischenräume nicht ausreichen, können Sie zusätzliches Papier von der Aufsicht erhalten. Beschriften Sie dieses sofort mit ihrem Namen und Ihrer Matrikelnummer. Eigenes Papier ist nicht zulässig. Schreiben Sie ausschließlich mit dokumentenechten schwarzen oder blauen Stiften.
- **Vollständigkeit des Aufgabenblatts**: Die Klausur besteht aus den Aufgabenseiten 5 – 22 und einer Funktionsreferenz auf Seite R1. Die Aufgabenblätter sind doppelseitig bedruckt. Prüfen Sie, ob alle Blätter vorhanden sind.
- **Fragen**: Melden Sie sich, wenn Sie eine Frage haben. Die Aufsicht kommt zu Ihnen, verlassen Sie nicht unaufgefordert Ihren Platz.
- **Zwischenschritte**: Geben Sie Zwischenschritte an. So können Sie selbst bei falschem Endergebnis einen Teil der Punkte erreichen.

Usability

1. (a) Wieso wird ein Wireframe nur skizzenhaft gezeichnet und nicht detailliert ausgearbeitet? (4)

(b) Wie berücksichtigt ein:e gute UX-Designer:in, dass Nutzer:innen Fehler machen? Nennen Sie zwei Aspekte. (4)

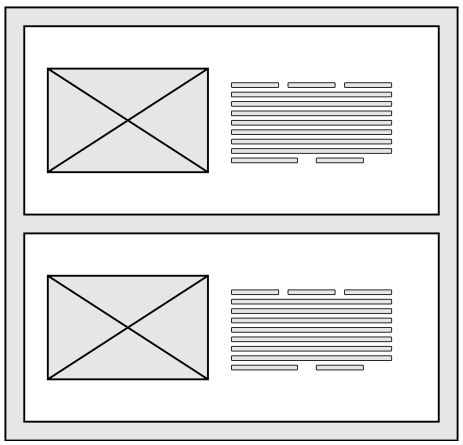
(c) Erläutern Sie, was mit dem Ausdruck “Responsiveness ≠ Leistung” gemeint ist. (4)

Gestaltgesetze

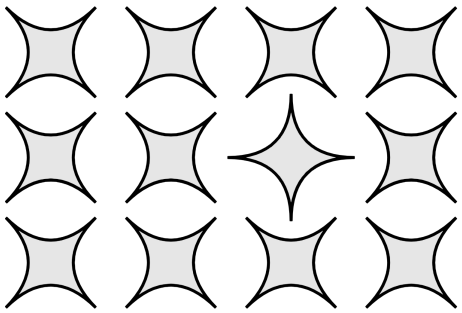
2. Betrachten Sie die folgenden Abbildungen. Welches Gestaltgesetz wird hier gezielt umgesetzt oder verletzt? Nennen Sie das umgesetzte oder verletzte Gestaltgesetz *und* begründen Sie kurz ihre Entscheidung.

(a) (4)

(b) (4)



(c) (4)



SmartPointer

3. Welches Problem wurde durch die Einführung von Smart-Pointern in C++ gelöst? Erläutern Sie dies anhand eines konkreten Beispiels. (6)

C++ – Operatoren

4. Betrachten Sie die C++-Klasse `AverageMaker`. Sie speichert ein Liste von int-Werten und soll die folgenden (überladenen) Operatoren bieten. (17)

   - Der Operator `<<` fügt eine Zahl der Liste hinzu. (5 P)
   - Der Operator `[]` liefert eine Zahl aus der Liste mit dem gegebenem Index, so dass lesend **und** schreibend darauf zugegriffen werden kann. (5 P)
   - Der Operator $\sim$ liefert den Durchschnitt aller Elemente als `double`-Wert. Gibt es keine Elemente, wird stattdessen eine Exception geworfen (7 P)

Headerdatei `averagemaker.h`

```
class AverageMaker {
public:
    size_t size() { return m_values.size(); }

private:
    std::vector<int> m_values;
```

```
};
```

Der folgende Code soll möglich sein:

```
int main(int argc, const char * argv[]) {
    AverageMaker av;
    av << 2 << 3 << 4;
    av[1] = 6;

    std::cout << "Durchschnitt: " << ~av; // Durchschnitt: 3
    return 0;
}
```

Implementieren Sie im Folgenden die drei genannten Operatoren. Ergänzen Sie die zuvor genannte Headerdatei, sofern erforderlich.

Sourcedatei `averagemaker.cpp`

C++ – Templates

5. Schreiben Sie eine Template-Funktion, die folgende Parameter besitzt. (12)

   - einen Zeiger auf ein Array von Zahlen
   - die Anzahl der Elemente in dem Array
   - eine Untergrenze
   - eine Obergrenze

   Die Funktion gibt zurück, wieviele Werte des Arrays sich innerhalb der angegebenen Unter- und Obergrenze befinden.

   Beispiel:

   - Array: $[1, 2, 3, 2, 6, 5]$
   - Untergrenze: 2
   - Obergrenze: 5
   - Ergebnis: 4 (die Elemente 2, 3, 2 und 5 befinden sich zwischen den Grenzen)

   Realisieren Sie die beschriebene Funktion als Template-Funktion in C++. Der Datentyp der Zahlen soll dabei über einen Template-Parameter `T` realisiert werden.

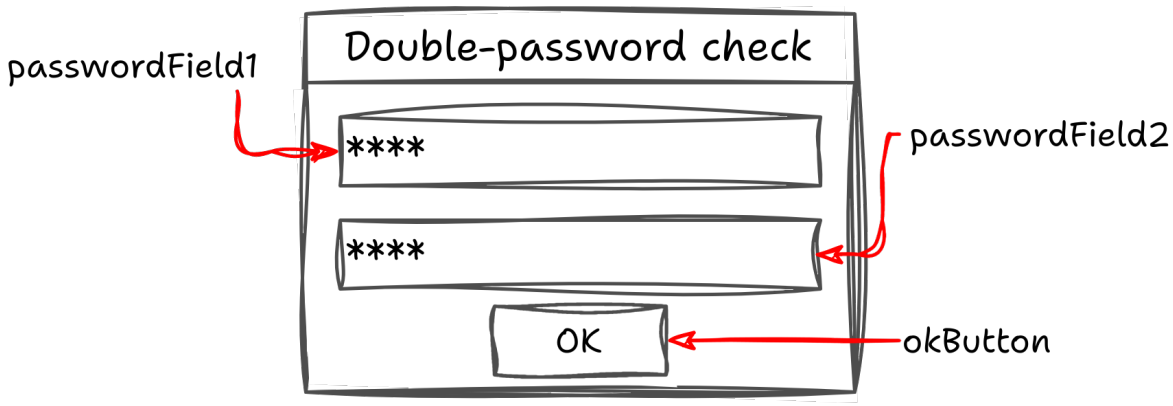
Qt-Grundlagen

(a) Erläutern Sie den Unterschied zwischen einem **modalen** und einem **nicht-modalen** Dialog in Qt. (4)

(b) Was sind die Besonderheiten der Qt-Klasse **QObject** und aller davon abgeleiteten Subklassen in Qt? (5)

Qt – GUI

6. Betrachten Sie das folgende Wireframe einer GUI für die Eingabe von Passwörtern. (20)



Die GUI wurde bereits im Qt-Designer umgesetzt. Implementieren Sie in der Hauptfensterklasse `MainWindow` (erbt von QMainWindow) die folgende Anwendungslogik:

- Legen Sie ein QString-Attribut `correctPassword` für ein korrektes Passwort sowie weitere ggfs. notwendige Attribute an.
- Der OK-Button ist nur **drückbar**, wenn beide Passwortfelder mindestens drei Zeichen enthalten und die Eingaben der beiden Passwortfelder übereinstimmen. Schreiben Sie für diese Überprüfung eine Methode `checkPasswords`.
- Bei Klick auf den OK-Button wird der QString der Passwortfelder mit dem hinterlegten korrekten Passwort QString-Attribut verglichen. Stimmt die Eingabe mit dem hinterlegten Passwort überein, erscheint eine entsprechende Nachricht in einer QMessageBox. Bei Abweichung vom korrekten Passwort erscheint eine Fehlermeldung. Diese Logik wird in einer Methode `onOkButtonClicked` implementiert.
- Stimmt das eingegebene Passwort drei Mal in Folge nicht mit dem hinterlegten Passwort überein, werden beide Passwortfelder und der OK-Button ausgegraut bzw. deaktiviert.

Hinweis: Sie können davon ausgehen, dass Eingaben in beiden Passwortfeldern bereits im Qt-Designer durch Stern-Symbole unkenntlich gemacht werden. Trennen Sie den Quelltext in Header- und Source-Datei sinnvoll auf.

Headerdatei mainwindow.h

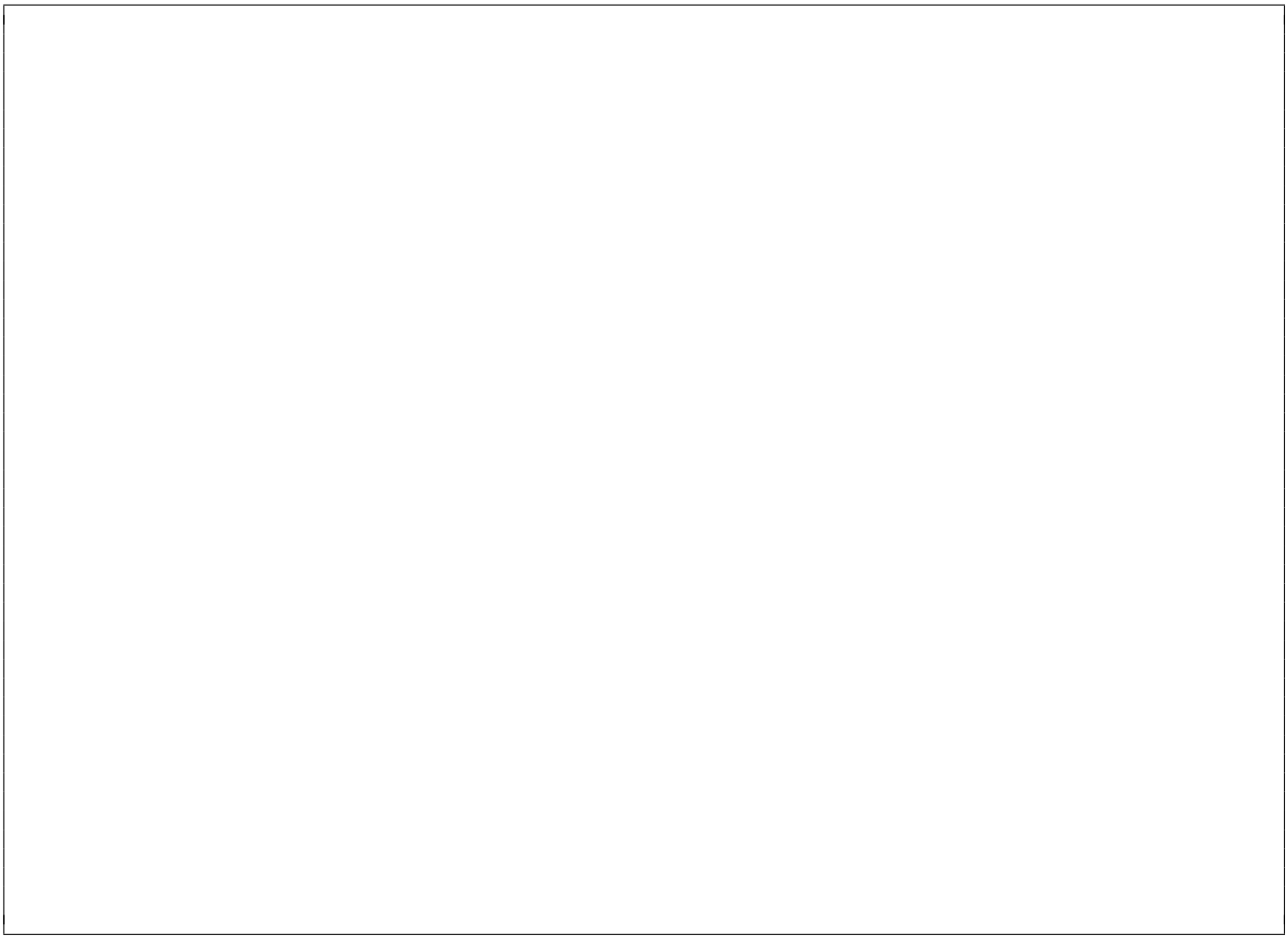
```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QLineEdit>
#include <QPushButton>

class MainWindow : public QMainWindow {
  Q_OBJECT

public:
  MainWindow(QWidget *parent = nullptr);

private:
```



```
};

#endif // MAINWINDOW_H
```

Sourcedatei mainwindow.cpp

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

MainWindow::MainWindow(QWidget * parent): QMainWindow(parent) {
```

```
}
```

Nebenläufigkeit

7. Betrachten Sie den folgenden Thread-Code der Klasse `MagicThread`, die von `QThread` abgeleitet wurde:

```
void MagicThread::run() {

    for(int i = 1; i <= m_value; i++) {

        prepareMagicStuff();

        doExpensiveMagicStuff();

        finishMagicStuff();

    }

}
```

(a) Will man den Thread auf die klassische Art beenden, d.h. über (6)

```
thread->quit();
thread->wait();
delete thread;
```

muss man warten, bis der Thread komplett durch die Schleife gelaufen ist, bevor er tatsächlich beendet wird.

Wie lässt sich das Beenden sauber beschleunigen? Geben Sie den konkreten Code an, der dafür erforderlich ist.

(b) Es soll sichergestellt werden, dass der Aufruf `doExpensiveMagicStuff();` nur von maximal drei Threads gleichzeitig erfolgen kann. Wie wird das umgesetzt? Geben Sie den konkreten Code an, der dafür erforderlich ist. (6)

Qt Cheat Sheet

Signale

QPushButton / QAbstractButton

```
void    clicked(bool checked = false)
void    toggled(bool checked)
void    pressed()
void    released()
```

QLineEdit

```
void    textChanged(const QString &text)
void    editingFinished()
void    inputRejected()
void    returnPressed()
void    selectionChanged()
```

Methoden

QLineEdit

Text auslesen
QString text() const

Text setzen
void setText(const QString &)

QAbstractButton

Auslesen: Ist Checkbox angehakt?
bool isChecked()

QString

Länge ermitteln
int size() const

Slots

QWidget

```
bool    close()
void    hide()
void    setDisabled(bool disable)
void    setEnabled(bool)
void    setFocus()
void    setHidden(bool hidden)
void    setVisible(bool visible)
void    setWindowModified(bool)
void    setWindowTitle(const QString &)
void    show()
void    update()
```

QAbstractButton

```
void    animateClick(int msec = 100)
void    click()
void    setChecked(bool)
void    setIconSize(const QSize &size)
void    toggle()
```

QPushButton *(erbt von QAbstractButton)*

```
void    showMenu()
```

QLineEdit

```
void    clear()
void    copy() const
void    cut()
void    paste()
void    redo()
void    selectAll()
void    setText(const QString &)
void    undo()
```

Layout-Manager

QHBoxLayout
QVBoxLayout
QFormLayout
QBoxLayout
QGridLayout
QStackedLayout

Thread-Synchronisation

QMutex
QReadWriteLock
QSemaphore

Message-Boxen

```
QMessageBox::critical(QWidget *parent, const QString &title, const QString &text,
    StandardButtons buttons = Ok, StandardButton defaultButton = NoButton);
```