



Klausur
MMI und GUI-Programmierung
Sommersemester 2024

Studiengänge: AI, MTI, WI

Prüfer: Prof. Dr. Malte Weiß

Persönliche Angaben

Nachname	Vorname	Matrikelnummer
----------	---------	----------------

Bewertung

Frage	Punkte	Erreicht
1	3	
2	9	
3	9	
4	6	
5	18	

Frage	Punkte	Erreicht
6	15	
7	6	
8	26	
9	8	
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 – 23 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.

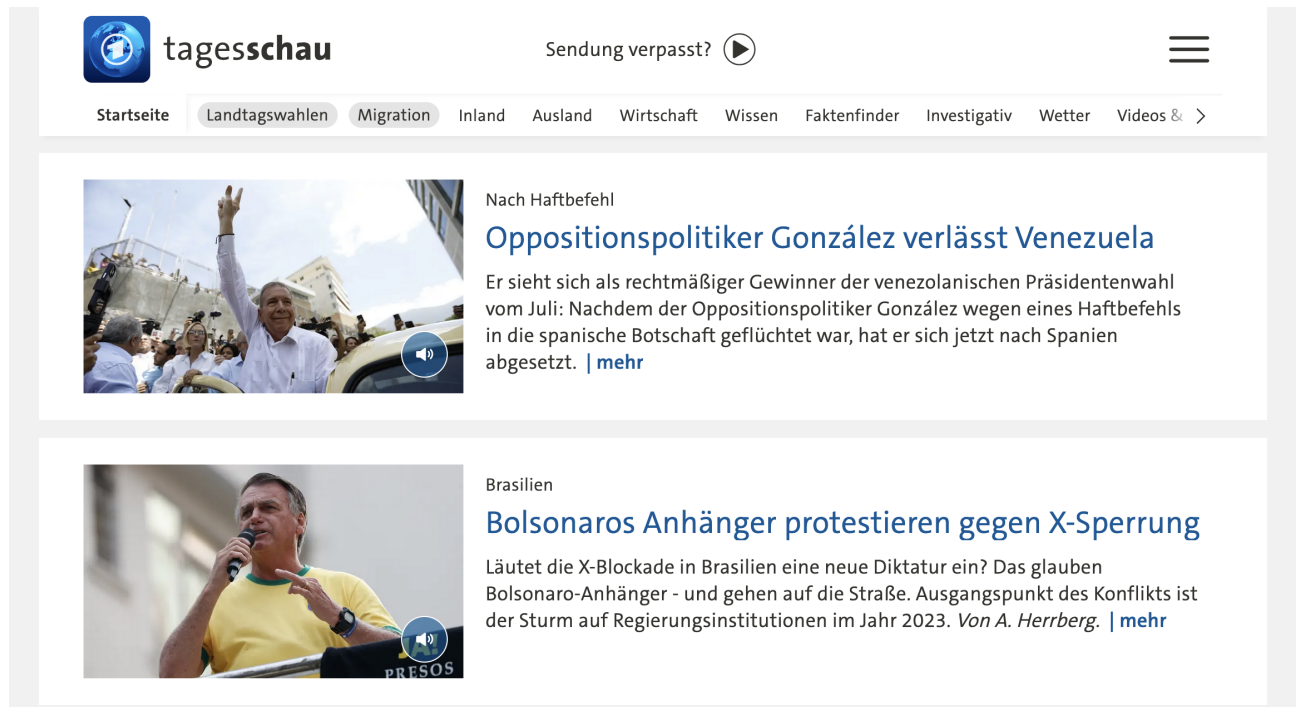
Matrikelnummer:

1. Eine goldene Regel der Usability lautet “Minimiere die kognitive Belastung”. (3)
Erklären Sie die Regel anhand eines Beispiels, das **nicht** aus der Vorlesung stammt.

MMI

2. Betrachten Sie den folgenden Auszug der Website tagesschau.de.

(9)



Nennen Sie **drei** Gestaltungsetze, die hier eingesetzt werden, und beschreiben Sie, welchen Effekt sie bei der Wahrnehmung der Website haben.

(Bitte lösen Sie die Aufgabe auf der nächsten Seite.)

Matrikelnummer:

C++-Grundlagen

3. (a) Nennen Sie für die folgenden Situationen das jeweils passende C++-Schlüsselwort. (3)

- Ein Attribut soll nach der Initialisierung unveränderlich sein:

- Ein Funktions-Member soll in einer Unterklasse überschreibbar sein (= polymorph):

- Ein `private` Attribut einer Klasse soll für einen anderen Typ sichtbar gemacht werden:

- (b) Welchen Vorteil besitzt ein *Shared Pointer* gegenüber der Verwendung der Operatoren `new` und `delete`? (2)

- (c) In C++ gibt es die Möglichkeit, Variablen im Konstruktor über sog. **Initialisierungslisten** zu setzen. Beispiel: (4)

Setzen im Konstruktor:

```
MyClass::MyClass(int x) {  
    m_x = x;  
}
```

Setzen über Initialisierungslisten:

```
MyClass::MyClass(int x) : m_x(x) {  
}
```

Nennen Sie **zwei** Fälle, in denen für die Initialisierung von Attributen eine Initialisierungsliste verwendet werden **muss**.

4. Schauen Sie sich den folgenden Code genau an. Was wird in der letzten Zeile jeweils ausgegeben?

(a) Code:

(2)

```
int x = 5;

int &r = x;
r = 6;

cout << x << endl;
```

Ausgabe:

(b) Code:

(2)

```
int x = 5;
int y = 10;

int &r = x;
r = y;
r = 6;

cout << y << endl;
```

Ausgabe:

(c) Code:

(2)

```
int array[] = {1, 2, 3};
int *p = &array[0];
*(p + 1) = 5;

cout << array[0] << endl;
```

Ausgabe:

Polymorphie

5. In dieser Aufgabe sollen Sie eine Klassenhierarchie implementieren, die Positionen in einem Warenkorb darstellt.

(a) Die Klasse `Position` stellt eine Position in einem Warenkorb da:

(6)

- Ein Attribut für den Namen eines Produkts und einen entsprechende Getter.
- Einen Konstruktor, der den Namen setzt.
- Eine rein virtuelle, nicht implementierte Methode `getPreis()`, die einen Preis zurückgibt.
- Eine polymorphe Methode `umtauschen()`. Sie erhält keine Parameter und gibt keinen Wert zurück. Stattdessen gibt sie nur den Text "Produkt wird umgetauscht" aus.

- (b) Die Klasse `StueckgutPosition` ist von `Position` abgeleitet und stellt kaufbares Stückgut dar (z.B. Handys, Brötchen, Wassermelonen etc.). Die Klasse besitzt zusätzlich zur Oberklasse: (6)
- Ein Attribut für den Stückpreis und ein Attribut für die Anzahl.
 - Einen Konstruktor, der alle Attribute setzt (auch das der Oberklasse).
 - Getter für alle Attribute.
 - Eine überschriebene Variante von `getPreis()`, die den Preis als $\text{Anzahl} * \text{Stückpreis}$ zurückgibt.

- (c) Die Klasse `EssbaresStueckgutPosition` ist von `StueckgutPosition` abgeleitet. Sie fügt keine neuen Attribute hinzu. (6)

Aber: Ein Umtausch einer solchen Position ist nicht möglich. Die Methode `umtauschen()` sollte daher überschrieben werden und beim Aufruf eine Exception werfen (z.B. `std::runtime_error`). Tipp: Konstruktor nicht vergessen.

C++ – Operatoren

6. (a) Betrachten Sie die folgende Struktur eines 4D-Vektors:

(5)

```
struct Vector4d {  
    int x, y, z, w;  
};
```

Überladen Sie den Operator << so, dass man ein Vector4d-Objekt über cout ausgeben kann, z.B. so:

```
Vector4d v = { 1, 2, 3, 4 };  
cout << v << endl; // Ausgabe: (1, 2, 3, 4)
```

Ihre Lösung:

(b) Betrachten Sie den folgenden Typ, der eine Person darstellt:

(5)

```
struct Person {  
    string name;  
    int age;  
};
```

Überladen Sie den Operator ! so, dass er eine Person volljährig macht, sofern sie es nicht bereits ist. Die Anwendung sieht so aus:

```
Person p = {"Alice", 17};  
Person q = {"Bob", 23};  
!p;  
!q;  
cout << p.age << endl; // 18 (neu)  
cout << q.age << endl; // 23 (unverändert)
```

Ihre Lösung:

- (c) Betrachten Sie die folgenden Struktur, die eine Gruppe von Personen darstellt. (5)
Der Name jeder Person ist als String in einem Vektor gespeichert:

```
struct Group {  
    vector<string> persons;  
};
```

Überladen Sie eine Additions-Operator +, so dass sich zwei Gruppen addieren lassen. Die Summe der Gruppen ist eine neue Gruppe, die alle Personen der beiden Gruppe enthält.

```
Group group1, group2;  
group1.persons.push_back("Alice");    // group1 = Alice, Bob  
group1.persons.push_back("Bob");  
group2.persons.push_back("Charles");  // group2 = Charles  
  
// group3 = Alice, Bob und Charles  
Group group3 = group1 + group2;
```

Ihre Lösung:

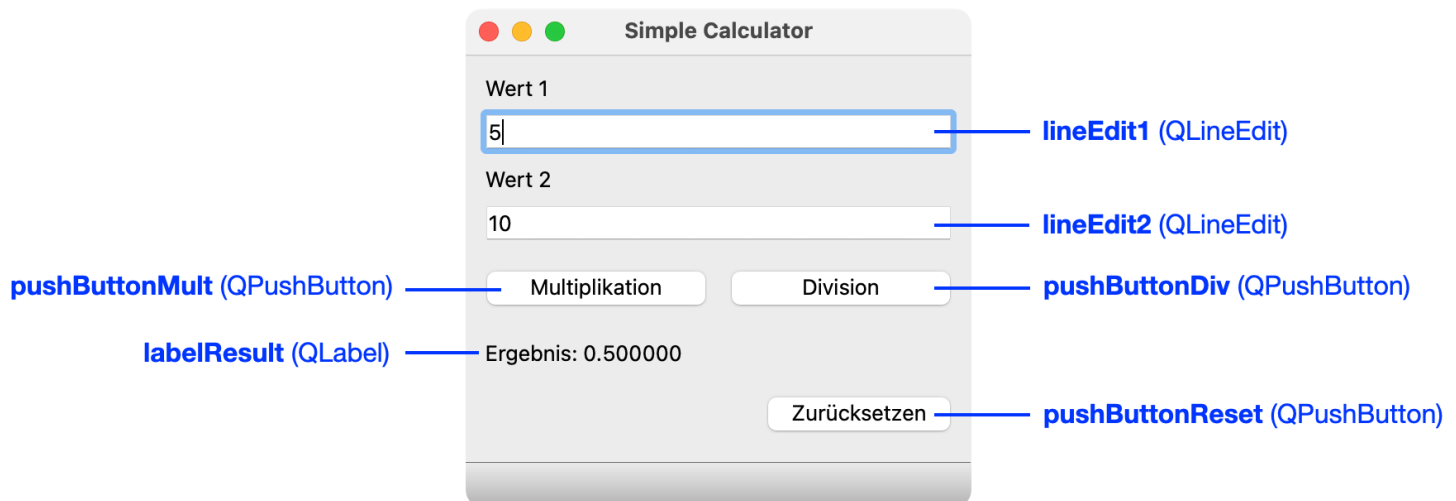
Qt – Grundlagen

7. (a) Nennen Sie **zwei** Gründe, die dafür sprechen, eine Klasse von QObject abzuleiten. (3)

- (b) Wann sollte man eine Klasse **nicht** von QObject ableiten? (3)

Qt – GUI

8. Betrachten Sie das folgende Wireframe einer GUI, mit der sich einfache Berechnungen durchführen lassen.



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

- In die beiden Werte-Felder (`lineEdit1` und `lineEdit2`) kann ein Nutzer Zahlen eintragen.
- Klickt ein Nutzer auf “Multiplikation” wird das Ergebnis der Multiplikation der beiden Werte als “Ergebnis: <Wert>” im Ergebnisfeld `resultLabel` angezeigt.
- Klickt ein Nutzer auf “Division” wird das Ergebnis der Division der beiden Werte als “Ergebnis: <Wert>” im Ergebnisfeld `resultLabel` angezeigt. Ist allerdings der zweite Wert 0, erscheint stattdessen “Kann nicht durch 0 teilen.” im Ergebnisfeld.
- Beim Klick auf “Zurücksetzen” werden die Werte-Felder geleert und im Ergebnisfeld wird “Noch kein Ergebnis.” eingetragen.
- Die Schaltflächen “Multiplikation” und “Division” sind nur aktiviert, wenn in beiden Werte-Feldern etwas eingetragen ist.
- Die Schaltfläche “Zurücksetzen” ist nur aktiviert, wenn in mindestens einem Werte-Feld etwas eingetragen ist.
- Beim Starten der Anwendung sind alle Felder leer.

Tipps:

- Es bietet sich an, eine zentrale Methode schreiben, die die GUI-Dynamik umsetzt, d.h. das Aktivieren oder Deaktivieren von Schaltflächen.
- Die Referenz im Anhang enthält eine Übersicht über sinnvolle Funktionen.

- (a) Beschreiben Sie, welche Layout-Manager und Spacer eingesetzt werden, um die gezeigte GUI umzusetzen. Beachten Sie dabei, dass die Fenstergröße geändert werden kann. Sie dürfen die Aufgabe über eine Grafik, einen Text oder beides lösen.

(6)

- (b) Vervollständigen Sie den Code auf den folgenden Seiten, um die beschriebene Logik zu implementieren. Trennen Sie den Quelltext in Header- und Source-Datei sinnvoll auf. (20)

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:
    Ui::MainWindow *ui;
```

```
};
```

```
#endif // MAINWINDOW_H
```

Sourcdatei mainwindow.cpp (Konstruktor)

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

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

```
}
```

```
MainWindow::~MainWindow()  
{  
    delete ui;  
}
```

Sourcedatei mainwindow.cpp (weitere Methoden)

Matrikelnummer:

Qt – Nebenläufigkeit

9. (a) Man sollte Threads einsetzen, damit der GUI-Thread bei lang laufenden Aufgaben nicht blockiert. Was ist die technische Ursache für eine Blockierung der GUI? (4)

- (b) Erweitern Sie den folgenden Code mit Mitteln von Qt so, dass die `for`-Schleife nur von drei Threads gleichzeitig durchlaufen werden kann. (4)

```
void myFunction() {  
  
    for(int i = 0; i < 100; i++) {  
  
        doSomeStuff();  
  
    }  
  
}
```

Qt Cheat Sheet

Signale

QPushButton / QAbstractButton

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

QSlider

```
void valueChanged(int value)
void sliderPressed()
void sliderMoved(int value)
void sliderReleased()
```

Methoden

QLabel

Text auslesen

```
QString text() const
```

Wert setzen

```
void setText(const QString &)
void setNum(int)
```

QSlider

Wert auslesen

```
int value() const
```

Wert setzen

```
void setValue(int)
```

QString

String mit Zahl erstellen

```
QString::number(int)
```

String in Zahl umwandeln

```
toInt()
```

```
toFloat()
```

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()
```

Layout-Manager

QHBoxLayout

QVBoxLayout

QFormLayout

QBoxLayout

QGridLayout

QStackedLayout

Thread-Synchronisation

QMutex

QReadWriteLock

QSemaphore