HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES
INSTITUT INFORMATIK

18.09.2019

Klausur

MMI und GUI-Programmierung

Sommersemester 2019

Studiengänge: AI, MTI, WI

Prüfer: Prof. Dr. Malte Weiß

Persönliche Angaben

Nachname	Vorname	Matrikelnummer

Bewertung

Frage	Punkte	Erreicht
1	3	
2	4	
3	6	
4	12	
5	10	
6	6	
7	4	
8	8	

Frage	Punkte	Erreicht
9	3	
10	6	
11	25	
12	4	
13	3	
14	6	
15	0	
Gesamt:	100	

Ablauf der Prüfung

- **Zeit für die Lösung dieses Aufgabenblatts**: 120 Minuten.
- **Erlaubte Hilfsmittel**: Cheat Sheet am Ende der Klausur. Darüber hinaus sind keine Hilfsmittel erlaubt.
- **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.

MMI

1. Definieren Sie den Begriff “Usability” gemäß der in der Vorlesung vorgestellten DIN-Norm (DIN EN ISO 9241-11). (3)

2. Beschreiben Sie die Hauptziele von gutem UI/UX-Design. (4)

3. Beschreiben Sie die **drei** Gestalt Gesetze Ihre Wahl. Zeichnen Sie jeweils ein einfaches Beispiel. (6)

4. Nehmen wir an, Sie sollen eine Eingabemaske für den Bestellprozess in einem Web-Shop gestalten. Die Maske soll folgende Felder enthalten: (12)

   - Informationen über den Kunden:
     - Name
     - Vorname
     - Straße
     - Hausnummer
     - Postleitzahl
     - Stadt
   - Zahlungsinformationen:
     - IBAN (internationale Kontonummer)
     - BIC (Bankcode)
     - Name der Bank
   - Lieferoptionen
     - Soll Lieferung als Geschenk verpackt werden? (Ja/Nein)
     - Soll Lieferung per Express geschickt werden? (Ja/Nein)

   Außerdem gibt es drei Knöpfe:

   - Bankverbindung prüfen
   - Speichern
   - Bestellung abbrechen

   Zeichnen Sie einen Dialog, der es erlaubt, die obigen Felder ausfüllen, und die drei genannten Knöpfe enthält. Achten Sie dabei auf eine hohe Usability und setzen Sie dabei mindestens **drei Gestalt Gesetze** ein.

   Hinweis: Bitte achten Sie auf eine saubere Zeichnung. In dieser Aufgabe dürfen Sie auch mit Farben oder Schraffuren arbeiten.

   Bitte verwenden Sie die **nächste Seite**.

5. (a) Was zeichnet eine Anwendung aus, die eine gute *Responsiveness* bietet? (3)

(b) Lesen Sie das folgende Szenario: (7)

Tim hat Oma Adelheid gestern "Smurf OS" auf ihrem Rechner installiert. Heute versucht Oma Adelheid, ein Textverarbeitungsprogramm zu starten. Sie klickt doppelt auf das entsprechende Icon auf dem Desktop, aber nichts passiert. Sie klickt ein zweites Mal, ein drittes Mal ... plötzlich startet das Textverarbeitungsprogramm und wird drei Mal auf dem Bildschirm sichtbar.

Beantworten Sie nun die folgenden Fragen:

1. Was ist hier aus Sicht der Usability schief gegangen?

2. Wie könnte man dieses Design mit dem Wissen über die “Human Deadlines” verbessern?

C++

6. Betrachten Sie die folgende Klasse, die einen 3D-Vektor der Form $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ darstellt. (6)

```
class Vector3D
{
public:
    Vector3D(float x, float y, float z) : m_x(x), m_y(y), m_z(z) {}

    inline float x() { return m_x; }
    inline float y() { return m_y; }
    inline float z() { return m_z; }

private:
    float m_x, m_y, m_z;
};
```

Die **Länge** eines Vektors $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ ist definiert als $\sqrt{x^2 + y^2 + z^2}$.

Überladen Sie den **einstelligen** Operator `!` (Ausrufezeichen) zur Berechnung der Länge in der Klasse `Vector3D`, so dass folgender Code möglich ist:

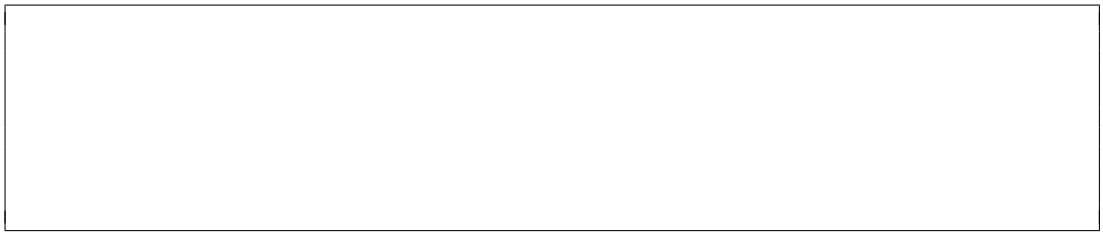
```
Vector3D a(1,0,0);
Vector3D b(1,2,3);
Vector3D c(4,3,3);

qDebug() << !a; // 1
qDebug() << !b; // 3.74166
qDebug() << !c; // 5.83095
```

Ergänzen Sie den Code auf der nächsten Seite.

Hinweis: Die C-Funktion `sqrt(x)` berechnet die Quadratwurzel.

Headerdatei vector3d.h

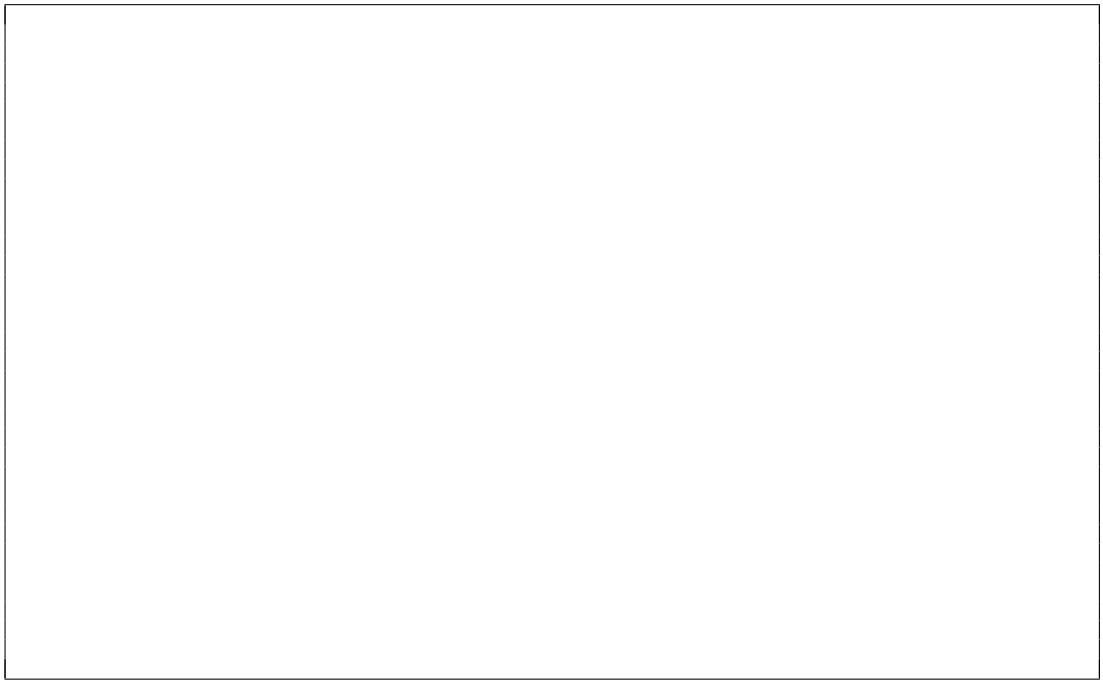
```
class Vector3D
{
public:
    Vector3D(float x, float y, float z) : m_x(x), m_y(y), m_z(z) {}

    inline float x() { return m_x; }
    inline float y() { return m_y; }
    inline float z() { return m_z; }

private:
    float m_x, m_y, m_z;
```



```
};
```

Sourcedatei vector3d.cpp

```
#include "vector3d.h"
```



7. Beschreiben Sie, wofür das Schlüsselwort `friend` eingesetzt wird. Nennen Sie ein Einsatzbeispiel. (4)

8. Die folgenden Codezeilen verwenden unsichere C-Casts. Wählen Sie jeweils einen sichereren C++-Cast und korrigieren Sie die betroffene Codezeile.

(a) (2)

```
int getAbsolute(int *v) { return *v < 0 ? -*v : *v; }

int main()
{
    const int x = 5;
    qDebug() << getAbsolute(&x);
    return 0;
}
```

Korrigierte Codezeile:

(b) (2)

```
void gehaltAusgeben(Mensch* mensch) {
  Mitarbeiter* mitarbeiter = (Mitarbeiter*) mensch;

  printf("Gehalt: %.2f", mitarbeiter->getGehalt());
}
```

Korrigierte Codezeile:

(c) (2)

```
double x = 3.7;
char i = (char) x;
```

Korrigierte Codezeile:

(d) (2)

```
// 64-Bit-Double-Kodierung als 64-Bit-Zahl ausgeben
double f = 1.25;
quint64* lp = (quint64*) &f;
qDebug() << *lp;
```

Korrigierte Codezeile:

Qt

9. Nennen Sie drei wesentliche Eigenschaften des Qt-Frameworks. (3)

   1.

   2.

   3.

10. Entscheiden Sie für die folgenden Aussagen, ob sie richtig oder falsch sind. Eine Begründung ist nicht erforderlich.

(a) Ein Objekt vom Typ QObject kann nicht kopiert werden. (1)

◯ Richtig ◯ Falsch

(b) QObject-Objekte sind einer Hierarchie organisiert. (1)

◯ Richtig ◯ Falsch

(c) Alle Objekte des Qt-Frameworks sind von QObject abgeleitet. (1)

◯ Richtig ◯ Falsch

(d) Ein modaler Dialog erlaubt keinen Einsatz von Signals und Slotsl (1)

◯ Richtig ◯ Falsch

(e) Werden in einem Projekt Signals und Slots verwendet, reicht ein reiner C++-Compiler nicht zum Kompilieren aus. (1)

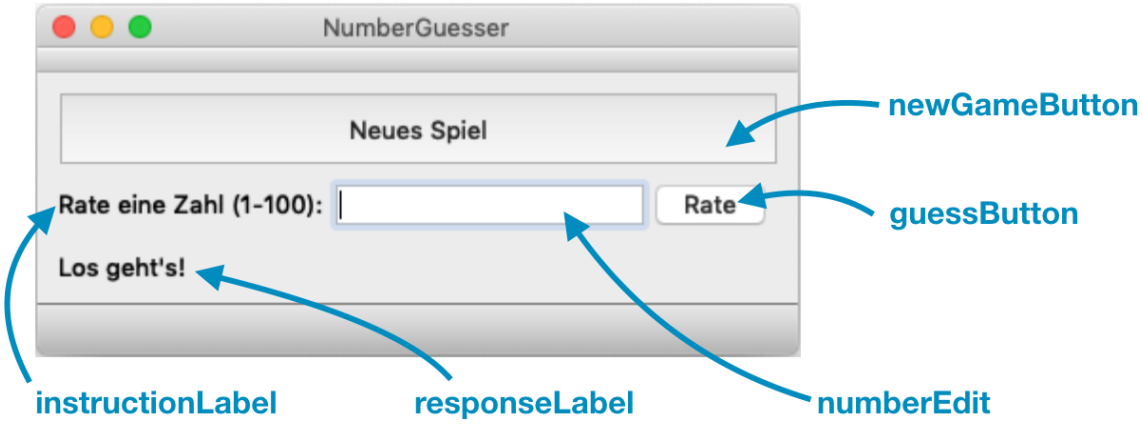
◯ Richtig ◯ Falsch

(f) Ein Objekt, das Signale (Signals) empfängt, muss zwingend von QObject abgeleitet sein. (1)

◯ Richtig ◯ Falsch

11. Sie sind der Chef-Entwickler der *Bankrupt Software AG* und sollen das Unternehmen mal wieder vor der Insolvenz retten, indem Sie eine bahnbrechende GUI-Anwendung schreiben. Nachdem PetAdder™ nicht so gut lief, sollen Sie jetzt das Spiel NumberGuesser™ entwickeln.

    NumberGuesser™ ist ein Spiel, bei dem ein Nutzer eine Zufallszahl raten soll. Die GUI sieht wie folgt aus. Objektnamen sind blau markiert.



    Das Hauptfenster (Klasse `MainWindow`) soll folgendes können:

    - Klickt ein Nutzer auf "Neues Spiel" (newGameButton), wird ein neues Spiel gestartet. Das Programm generiert dann eine neue Zufallszahl, das Ausgabe-Label (responseLabel) zeigt "Los geht's!" und das Eingabefeld für die Zahl (numberEdit) ist leer und aktiviert.
    - Klickt ein Nutzer auf "Rate" (guessButton), wird die Zahl aus dem Eingabefeld (numberEdit) ausgelesen und das Eingabefeld erstmal geleert. Dann wird die vom Nutzer eingegebene Zahl ausgewertet:
        1. Entspricht die Zahl der gesuchten Zufallszahl, erscheint die Meldung "Richtig!" und das Spiel wird beendet. Dann wird das Eingabefeld (numberEdit) deaktiviert, bis ein neues Spiel gestartet wird. Außerdem soll ein Signal gameFinished ausgelöst werden.
        2. Ist die geratene Zahl zu klein, erscheint die Meldung "Zu klein!". Der Nutzer kann anschließend weiterraten.
        3. Ist die geratene Zahl zu groß, erscheint die Meldung "Zu groß!". Der Nutzer kann anschließend weiterraten.
    - Der Knopf "Rate" (guessButton) ist nur drückbar, wenn sich mindestens ein Zeichen im Eingabefeld numberEdit befindet.
    - Beim Start des Programms soll direkt ein neues Spiel gestartet werden.

    Hinweise:

    - Sie können davon ausgehen, dass ein Nutzer immer korrekte Zahlen eingibt.
    - Das Qt Cheat Sheet auf der letzten Seite enthält hilfreiche Informationen.

(a) Überlegen Sie, welche Signals und Slots Sie benötigen und ergänzen Sie die folgende Header-Datei von `MainWindow` entsprechend. (6)

```
#include <QMainWindow>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private:
    Ui::MainWindow *ui;
    int m_secretNumber; // speichert die Zahl, die zu raten ist

public slots:
    void startGame();   // Slot, um neues Spiel zu starten
    void endGame();     // Slot, um Spiel zu beenden
```

```

```

```
signals:
```

```

```

```
};
```

(b) Ergänzen Sie nun die Implementierung der Slots `startGame()` und `endGame()`, gemäß des oben beschriebenen Verhaltens. (6)

```
void MainWindow::startGame()
{
    // Neue Zufallszahl von 1 bis 100 generieren
    m_secretNumber =
        (QRandomGenerator::global()->generate() % 100) + 1;

    // GUI für neues Spiel einrichten
```

```
}
```

```
void MainWindow::endGame()
{
    // GUI für Spielende einrichten
```

```
}
```

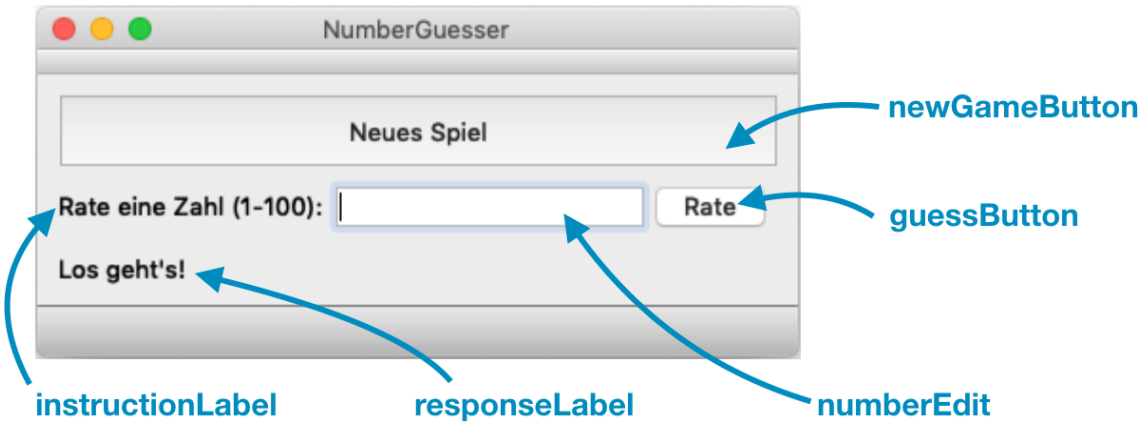
(c) Implementieren Sie nun den Konstruktor. Hier sollen die Signale und Slots korrekt verbunden werden. Außerdem sollte die GUI sinnvoll initialisiert werden. (5)

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
```

```
}
```

(d) Implementieren Sie nun die verbleibende GUI-Logik Ihrer Applikation in der Source-Datei, d.h. das Raten einer Zahl und das Verhalten der Rate-Schaltfläche, wenn das Eingabefeld leer oder gefällt ist. (8)

12. Betrachten wir nochmal die GUI aus Aufgabe 11. (4)



Erklären Sie an diesem Beispiel, wie das Layout einer solchen GUI in Qt erstellt wird.
Tipp: In der Referenz finden Sie die relevante Qt-Klassen.

Nebenläufigkeit

13. Erklären Sie, warum es in GUI-Anwendungen wichtig ist, bei länger andauernden Berechnungen auf Nebenläufigkeit zu setzen. (3)

14. Ergänzen Sie das folgende Programm so, dass die Berechnung von `x` in der Funktion `doSomeComputation` von höchstens drei Threads gleichzeitig ausgeführt werden kann. (6)

    Tipp: Referenz.

```
#include <math.h>

void doSomeComputation() {

    double x = (double) rand() / RAND_MAX;
    x *= 10 + pow(x, 2);

    return x;

}
```

Bonusfrage (+2 Punkte)

15. Es gilt “Responsiveness ≠ Leistung”. Was ist damit gemeint? (2 (bonus))

Qt Cheat Sheet

Signale

QPushButton / QAbstractButton

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

QLineEdit

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

Methoden

QLineEdit

Text auslesen
QString text() const

Text setzen
void setText(const QString &)

QString

Länge ermitteln
int size() const

String in int umwandeln:
int toInt() const

String aus int erzeugen:
static QString number(int n)

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