HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES
INSTITUT INFORMATIK

19.03.2024

Klausur

MMI und GUI-Programmierung

Wintersemester 2023/24

Studiengänge: AI, MTI, WI

Prüfer: Prof. Dr. Malte Weiß

Persönliche Angaben

Nachname	Vorname	Matrikelnummer

Bewertung

Frage	Punkte	Erreicht
1	9	
2	9	
3	6	
4	14	
5	8	

Frage	Punkte	Erreicht
6	12	
7	8	
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 – 24 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:

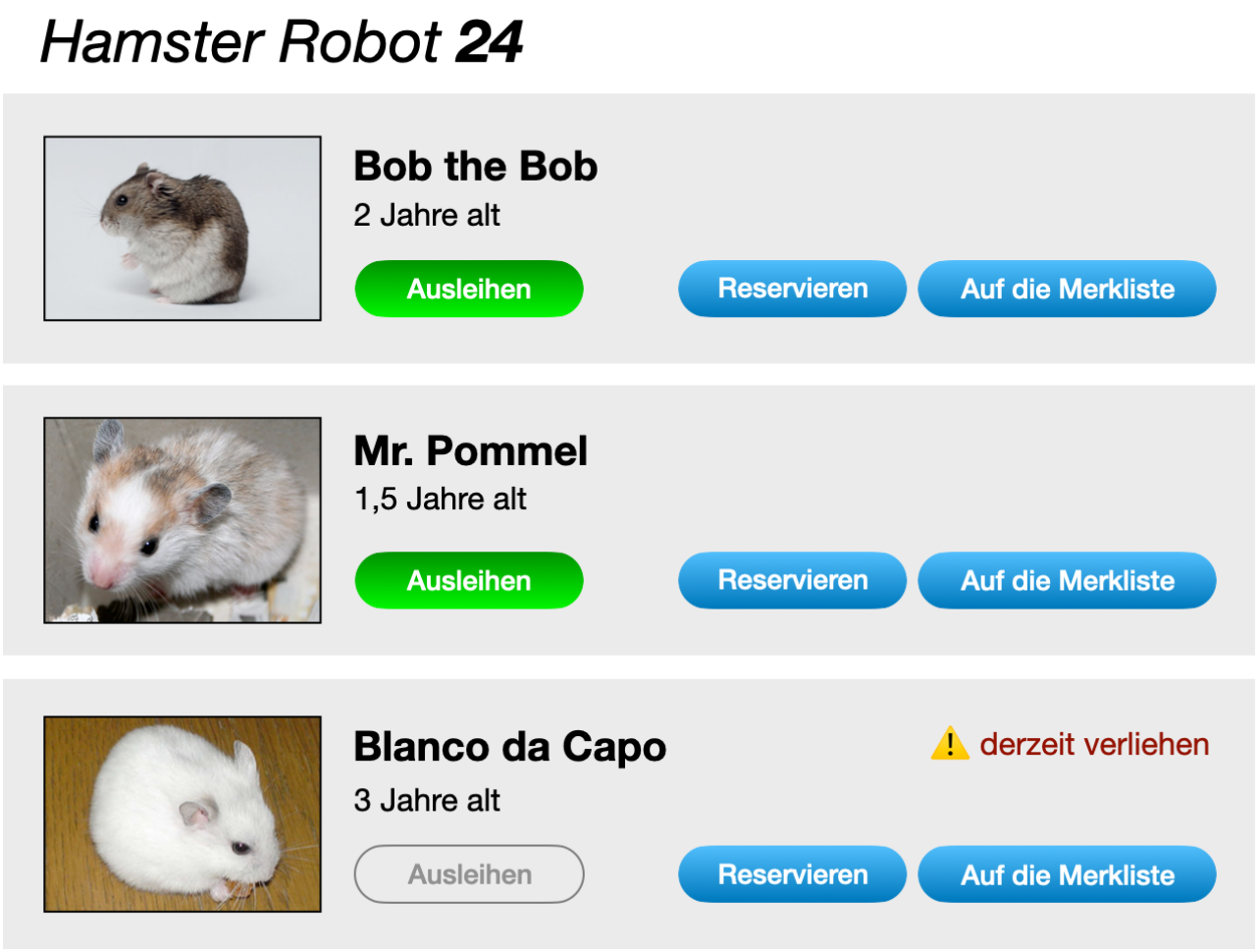
---

1. Stellen Sie sich einen Geldautomaten vor, an dem ein(e) Nutzer(in) Bargeld abheben kann. Wählen Sie **drei** der "10 goldenen Regeln der Usability" aus der Vorlesung und erklären Sie, wie die jeweilige Regel die Nutzererfahrung am Geldautomaten verbessern kann. (9)

Usability

2. Die Website Hamster Robot 24™ ist das Resultat jahrelanger KI-Forschung. Die Seite erlaubt das Ausleihen von KI-gesteuerten Roboterhamstern. (9)

   Betrachten Sie das folgende Interface der Website und identifizieren Sie **drei** unterschiedliche Gestalt-Gesetze, die hier zum Einsatz kommen, und inwieweit Sie das Verständnis der Oberfläche vereinfachen.



Benutzen Sie für Ihre Antwort bitte die nächste Seite.

Zeiger und Referenzen

3. Betrachten Sie die folgenden Codeausschnitte und geben Sie an, wie die genannte Variable nach der Ausführung belegt ist.

   (a) Code: (2)

```
int x = 5;
int& a = x;
a++;
```

   Variablenbelegung:

   x =

   (b) Code: (2)

```
int x = 5;
int y = 10;
int& a = x;
a = y;
a++;
```

   Variablenbelegung:

   y =

   (c) Code: (2)

```
char str[] = "Hallo Welt";
char* p = str;
char*& q = p;
q++;
*p = 'X';
```

   Variablenbelegung:

   str =

Objektorientierte Programmierung

4. Implementieren Sie die folgende Klassenhierarchie.

   (a) Implementierten Sie eine Klasse `Cake`, die einen Kuchen beschreibt. Ein Kuchen hat einen **Namen** und eine Variable, die bestimmt, ob der Kuchen eine **Glasur** (frosting) hat oder nicht. Implementieren Sie folgendes: (7)

   - Einen Konstruktor, der den Namen und die Glasur-Eigenschaft setzt. Alle Attribute sollen nach der Konstruktion **unveränderlich** sein.
   - Getter für alle Attribute
   - Eine polymorphe Methode `getCalories`, die die Anzahl der Kalorien zurück gibt. Ein Kuchen hat standardmäßig 2200 Kalorien. Eine Glausur bedeutet weitere 200 Kalorien.

   Berücksichtigen Sie die Regeln der objektorientierten Programmierung aus der Vorlesung.

(b) Implementierten Sie die Klasse `SprinkleCake` als Unterklasse von `Cake`. Sie beschreibt einen Streuselkuchen, wobei als zusätzliches Attribut das Gewicht der Streusel in Gramm definiert werden kann. Implementieren Sie auch hier: (7)

- Einen Konstruktor, der alle Attribute setzt. Alle Attribute sollen nach der Konstruktion **unveränderlich** sein.
- Entsprechende Getter.
- Eine angepasste Methode `getCalories`. Zusätzlich zur von der Oberklasse berechneten Kalorienzahl kommen hier 4 Kalorien pro Gramm Streusel hinzu.

Berücksichtigen Sie die Regeln der objektorientierten Programmierung aus der Vorlesung.

Generics

5. Schreiben Sie eine generische Hilfsmethode `getAverageOfPositiveElements`. Sie erhält ein Array von Zahlen, die einen generischen Typ `T` besitzen und gibt den Durchschnitt der **positiven** Zahlen (> 0) zurück (negative Zahlen werden nicht berücksichtigt). Gibt es keine positiven Zahlen, gibt die Methode -1 zurück. (8)

Operatoren in C++

6. Betrachten Sie die folgende Klasse `LimitedList`. Die Klasse modelliert eine Liste von Ganzzahlen, die eine begrenzte Kapazität besitzt. In der aktuellen Implementierung erlaubt sie das Setzen der Kapazität über den Konstruktor und die Rückgabe der Kapazität über einen Getter. Außerdem wird intern eine Liste für die Elemente angelegt: (12)

```
class LimitedList {
public:
    LimitedList(size_t capacity) : m_capacity(capacity) {}

    size_t getSize() { return m_elements.size(); }

private:
    std::vector<int> m_elements;
    size_t m_capacity;
};
```

Implementieren Sie die folgenden Operatoren:

- Operator `+=` erlaubt, einen Wert der Liste hinzuzufügen. Würde durch das Hinzufügen eines Elements die Kapazität überschritten, soll stattdessen eine Exception geworfen werden (z.B. ein `std::runtime_error`).
- Operator `[]` erlaubt den Zugriff auf ein Element der Liste mit einem bestimmten Index. Die Gültigkeit des Indizes müssen Sie *nicht* prüfen.
- Operator `<<` erlaubt die Ausgabe der Liste in einen Stream, z.B. über `cout`.

Das folgende Programm demonstriert einen mögliche Einsatz der Operatoren:

```
LimitedList list(3);    // Liste mit Kapazität 3 anlegen
list += 2;              // Element 2 hinzufügen
list += 5;              // Element 5 hinzufügen
list += 3;              // Element 3 hinzufügen
list[1] = 6;            // zweites Element auf 6 setzen

std::cout << list;      // gibt 2, 6 und 3 aus

try {
    list += 5;          // überschreitet Kapazität
} catch(...) {
    std::cerr << "Exception aufgetreten." << std::endl;
}
```

Implementieren Sie nun die Operatoren in den freien Feldern:

```
class LimitedList {
public:
    LimitedList(size_t capacity) : m_capacity(capacity) {}

    size_t getSize() { return m_elements.size(); }
```

```

```

```
private:
    std::vector<int> m_elements;
    size_t m_capacity;
};
```

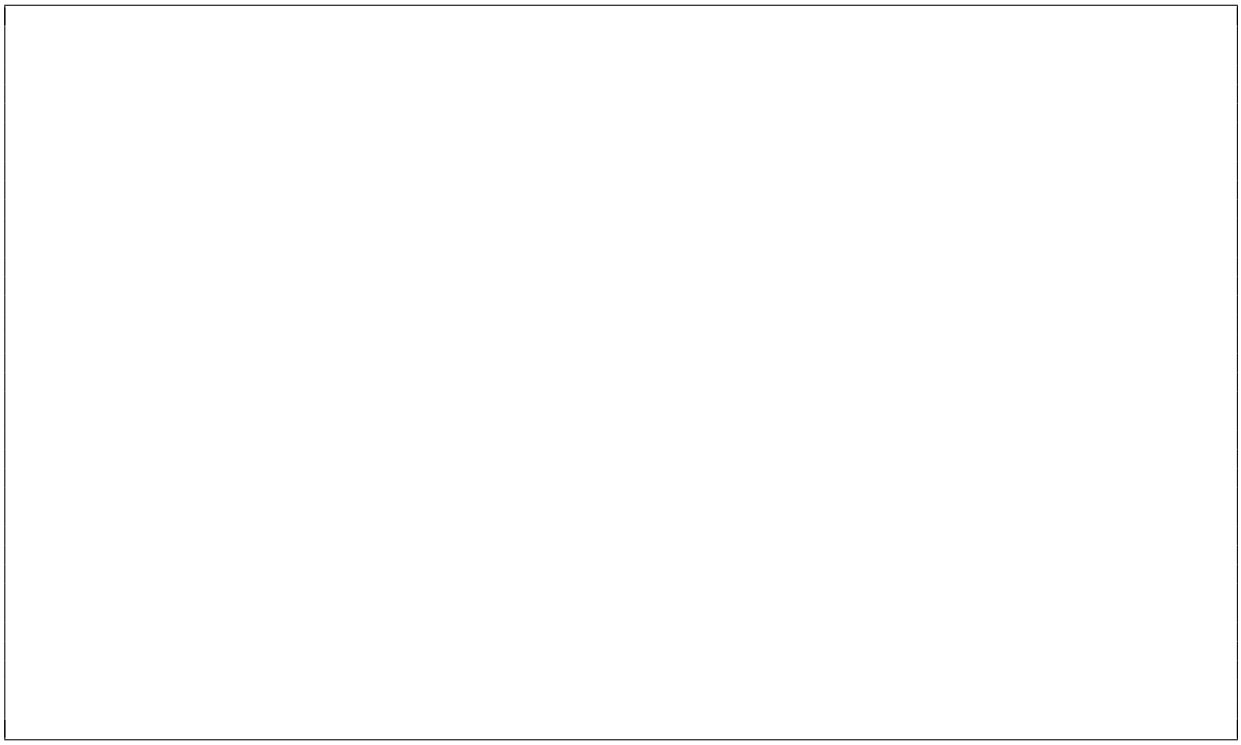
```

```

Analyse

7. Erläutern Sie die Fehler der folgenden beiden Code-Auszüge und korrigieren Sie den Fehler:

   (a) (4)

```
1  class Person {
2  public:
3      Person(string name) : m_name(name) {}
4
5      string getName() { return m_name; }
6
7  private:
8      string m_name;
9  };
10
11 int main() {
12     const Person p("Bob");
13     cout << p.getName();
14
15     return 0;
16 }
```

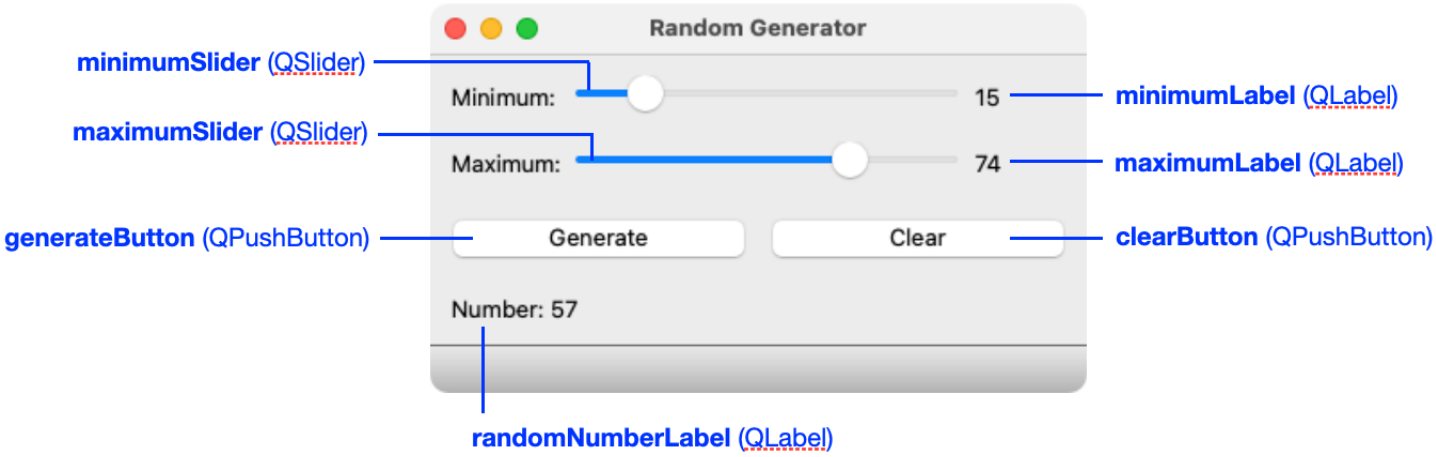


(b) (4)

```
class Employee {
public:
    Employee(string name, int age) {
        m_name = name;
        m_age = age;
    }

private:
    const string m_name;
    const int m_age;
};
```

Qt – GUI

8. Betrachten Sie das folgende Wireframe einer GUI, mit der sich eine Zufallszahl generieren lässt.



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

- Mit dem Minimum-Slider wird der Minimalwert (min) eingestellt. Der entsprechende Wert wird rechts von dem Slider angezeigt.
- Mit dem Maximum-Slider wird der Maximalwert (max) eingestellt. Der entsprechende Wert wird rechts von dem Slider angezeigt.
- Beim Starten der Anwendung steht der Minimum-Slider auf 0 und der Maximum-Slider auf 100. Es wird beim Starten keine generierte Zufallszahl gezeigt.
- Die Generate-Schaltfläche ist klickbar, wenn der Minimalwert kleiner oder gleich dem Maximalwert ist. Ansonsten ist er deaktiviert.
- Die Clear-Schaltfläche ist klickbar, wenn die Standardwerte aktuell nicht gesetzt sind (min = 0, max = 100).

Klickt ein Nutzer auf **Generate**, wird eine Zufallszahl zwischen dem Minimal- und dem Maximalwert generiert und angezeigt.

Tipps:

- Ein Objekt der Klasse `QRandomGenerator` stellt einen Zufallsgenerator dar. Der Generator sollte nur einmal erzeugt werden (also nicht als lokale Variable). Der Aufruf `bounded(min, max) + 1` erzeugt die gewünsche Zufallszahl.
- 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)

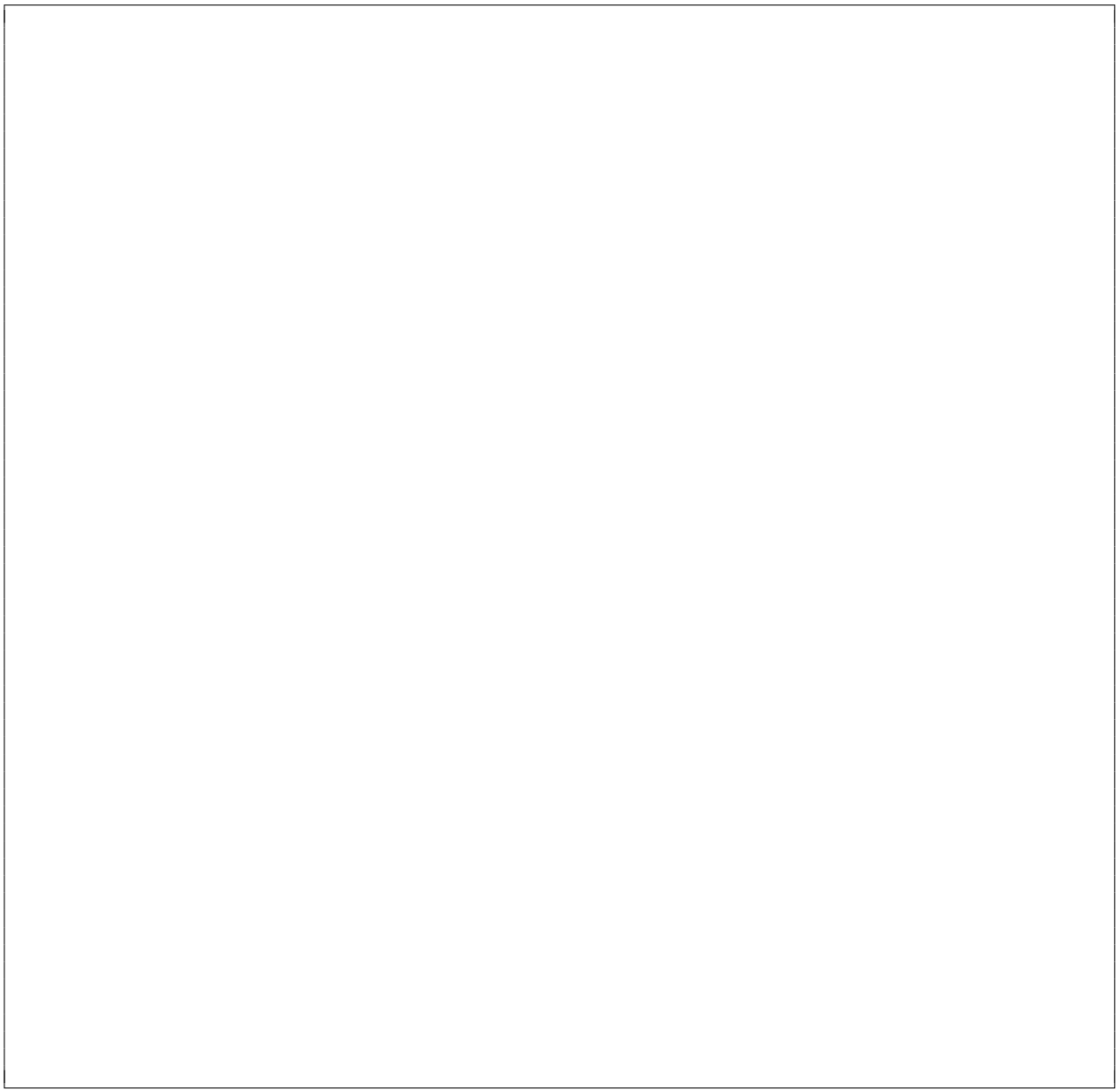
Headerdatei mainwindow.h

```
#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
```

Sourcedatei mainwindow.cpp (Konstruktor)

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

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



```
}

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

Sourcedatei mainwindow.cpp (weitere Methoden)

Qt – Nebenläufigkeit

9. (a) Erläutern Sie, warum man bei lang laufenden Nebenberechnungen in Qt einen separaten Thread benötigt, um weiterhin mit der GUI arbeiten zu können. (4)

(b) Erweitern Sie die folgende Funktion so, dass der enthaltene Code höchstens von **drei** Threads gleichzeitig ausgeführt werden kann: (4)

```
void myMagicFunction(int numEmployees) {

    for(int i = 0; i < numEmployees; i++) {

        auto employee = getEmployee(i);

        precomputeEmployeeSalary(employee);

        transferEmployeeSalary(employee);

    }

}
```

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)

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