HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES

16.03.2022

Klausur

MMI und GUI-Programmierung

Prüfer: Prof. Dr. Malte Weiß

Vorname	Nachname	Matrikelnummer

Bewertung:

1	2	3	**Gesamt**	**Note**
39	41	20 (+ 5)	100 (+5)	

Zum Ablauf der Prüfung

- **Zeit für die Lösung:** 4 Stunden
- **Beginn:** 9:00 Uhr, **Ende:** 13:00 Uhr am 16.03.2022
- **Lösung *wo* hochladen:** in E-Learning im https://onlinepruefung.hs-ruhrwest.de/course/view.php?id=3764
- **Lösung *wie* hochladen (Form):** Zip-Datei im Format *mmi-exam-VORNAME-NACHNAME-MATRIKELNUMMER.zip*.
  Beispiel: Student *Mathias Bauer* mit Matrikelnummer *10001234* gibt eine Zip-Datei mmi-exam-Mathias-Bauer-10001234.zip ab.
- **Ordnerstruktur:** Jede Aufgabe liegt in einem entsprechenden Unterordner in der Zip-Datei. Siehe aufgabenspezifische Konventionen.
- **Nicht erlaubte Hilfsmittel/Hilfe:** Kommunikation digital oder persönlich, direkt oder indirekt mit anderen Menschen (**keine** SMS, WWW-Kommunikation, Telefon, E-Mail, Chat, etc.)
- **Kompilierbarkeit:** Lösungen *müssen* kompilieren, ansonsten werden 50 % der Aufgabenpunkte abgezogen.
- **Fragen:** Für Fragen während der Klausurzeit schreiben Sie bitte eine E-Mail an malte.weiss@hs-ruhrwest.de
- Die Klausur gilt als **nicht angetreten**, wenn Sie die eidesstattliche Versicherung nicht unterschrieben abgeben (siehe Aufgabe 0).

Aufgabe 0: Eidesstattliche Erklärung

Voraussetzung für die Teilnahme an dieser Prüfung ist das Ablegen einer Eidesstattlichen Erklärung:

1. Lesen Sie die Datei **Formular eidesstattliche Versicherung MMI.pdf** im Ordner **Eidesstattliche Versicherung** der Vorlage.

2. Unterschreiben Sie die Eidesstattliche Erklärung, indem Sie alle Felder des Formulars ausfüllen, d.h. Name, Matrikelnummer, Semester, Ort und Datum, Unterschrift (als Unterschrift reicht es aus, wenn Sie Ihren Namen eintragen)

3. Speichern Sie die ausgefüllte PDF-Datei in Ihrem abzugebenden ZIP-Archiv im Ordner **Eidesstattliche Versicherung**.

Hinweis: Fehlt die Erklärung, wird Sie verändert, unvollständig oder falsch ausgefüllt, gilt die Prüfung als **nicht angetreten**.

Aufgabe 1: Modellierung & Polymorphie (39 Punkte)

Die Lösung der Aufgabe muss im ZIP-Archiv im Ordner A1 gespeichert werden.

Ein *AssetHamster* ist ein Nagetier, das über Geldmittel und Aktienportfolios verfügt. Es besitzt die folgenden Eigenschaften:

- Name (std::string)
- Menge an Bargeld in Euro (double)
- Liste von Assets

Ein Asset wiederum besitzt zwei Eigenschaften:

- ID (std::string)
- Wert in Euro (double)

Achten Sie bei der folgenden Programmierung auf die Konventionen der Vorlesung (Information Hiding etc.).

Setzen Sie Folgendes um:

1. Implementieren Sie eine Klasse **Asset**, die ein Asset darstellt. Dazu gehören:

    a. Ein Standardkonstruktor und ein Konstruktor, der alle Werte setzt.
    b. Getter und Setter

    (4 Punkte)

2. Implementieren Sie eine Klasse **AssetHamster**, die einen AssetHamster darstellt. Dazu gehören:

    a. Ein Standardkonstruktor und ein Konstruktor, der den Namen und den Wert für die Bargeldmenge setzt.
    Hinweis: Die Assets werden nicht über den Konstruktor gesetzt.
    b. Getter und Setter
    c. Eine Methode, die ein Asset zum AssetHamster hinzufügt.

    (4 Punkte)

3. Implementieren Sie die Methode **assetAmount()** für die Klasse **AssetHamster**. Sie gibt den Gesamtwert aller Assets zurück.

    Implementieren Sie außerdem die Methode **totalWealth()** für die Klasse **AssetHamster**. Sie gibt das gesamte Vermögen des Hamsters zurück, d.h. die Summe aus der Bargeldmenge und den Gesamtwert aller Assets.

    (4 Punkte)

4. Implementieren Sie den **<<-Operator für die Klasse Asset**, so dass man ein Asset z.B. über cout menschenlesbar ausgeben kann. Alle Attribute des Assets sollen ausgegeben werden.

   Demonstrieren Sie die Funktion in einem **Hauptprogramm**.

   (4 Punkte)

5. Implementieren Sie den **<<-Operator für die Klasse AssetHamster**, so dass man ein AssetHamster z.B. über cout menschenlesbar ausgeben kann. Dabei soll folgendes ausgegeben werden:

   a. Name des Hamsters.
   b. Menge des Bargelds.
   c. Gesamtwert aller Assets.
   d. Gesamtes Vermögen des Hamsters.
   e. Liste aller einzelnen Assets (mit ID + Wert). Dabei <<-Operator von Asset wiederverwenden.

   Demonstrieren Sie die Funktion in einem **Hauptprogramm**.

   (6 Punkte)

6. Implementieren Sie für die **Klasse AssetHamster** den **Operator !** (Ausrufezeichen). Der Operator gibt einen neuen Hamster zurück. Dieser besitzt den gleichen Namen, allerdings keine Assets. Das Gesamtvermögen des aufgerufenen Hamsters liegt bei dem zurückgegebenen Hamster als Bargeld vor.

   Beispiel: Hamster **Alice** besitzt Bargeld im Wert von 10.000 € und Assets im Wert von 5.000 €. Durch Aufruf des !-Operators wird ein neuer Hamster zurückgegeben. Dieser besitzt Bargeld im Wert von 15.000 € und keine Assets.

   Demonstrieren Sie die Funktion in einem **Hauptprogramm**.

   (5 Punkte)

7. Implementieren Sie für die **Klasse AssetHamster** den **Operator <<=** (binäre Verschiebung mit Zuweisung). Dieser Operator überträgt das gesamte Vermögen eines Hamsters auf einen anderen Hamster.

   Beispiel: Hamster **Alice** besitzt Bargeld im Wert von 10.000 € und Assets im Wert von 5.000 €. Hamster **Bob** besitzt Bargeld im Wert von 12.000 € und Assets im Wert von 2.000 €.
   Durch den Aufruf `Alice <<= Bob` besitzt Alice Bargeld im Wert von 22.000 €

und zusätzlich alle Assets von Bob, also Assets im Wert von insgesamt 7.000 €. Bob hingegen besitzt nach der Operation weder Bargeld noch Assets.

Demonstrieren Sie die Funktion in einem **Hauptprogramm**.

(7 Punkte)

8. Ein **CheatingAssetHamster** ist von **AssetHamster** abgeleitet. Er besitzt dieselben Konstruktoren wie AssetHamster. Er ist bei der Rückgabe des verfügbaren Bargelds allerdings nicht ehrlich und gibt den doppelten Wert seiner Oberklasse zurück.

   Hinweis: Hier muss Polymorphie angewendet werden. Achten Sie auch darauf, dass bei der Ausgabe über cout der doppelte Wert ausgegeben wird.

   Demonstrieren Sie die Funktion in einem **Hauptprogramm**.

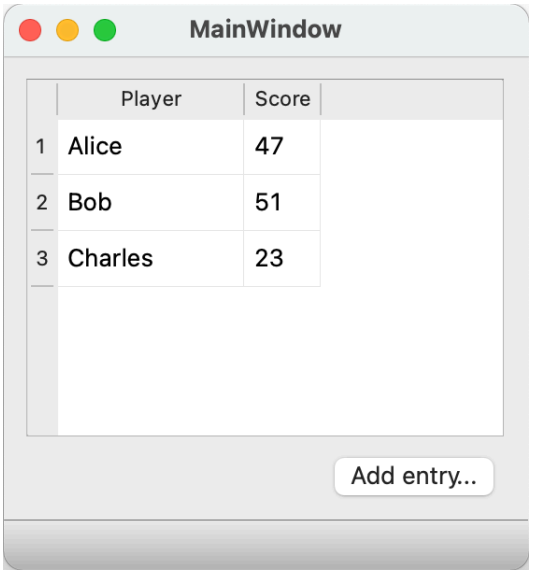
   (5 Punkte)

Aufgabe 2: Qt-Programmierung (41 Punkte)

Die Lösung der Aufgabe muss im ZIP-Archiv im Ordner A2 gespeichert werden.

Bob soll das Programm **Highscore** entwickeln. Das Programm dient dazu, die Punktzahlen von Spieler:innen zu erfassen. Bob hat auch schon angefangen. Leider ist er in ein spontan entstandenes Wurmloch gesogen worden und in einem Paralleluniversum gelandet. Daher ist es nun Ihre Aufgabe, das Programm zu vervollständigen.

Bisher sieht das Programm so aus:



Klickt ein:e Nutzer:in auf „Add entry“ wird ein Beispieleintrag hinzugefügt.

Sie finden das Programm **Highscore** in der Vorlage. Setzen Sie nun folgende Anforderungen um.

Hinweis: Die folgenden Teilaufgaben lassen sich weitestgehend unabhängig voneinander bearbeiten. Kommen Sie bei einer Teilaufgabe nicht weiter, können Sie trotzdem andere Teilaufgaben bearbeiten.

1. Geben Sie dem Hauptfenster einen **sinnvollen Titel** und stellen Sie sicher, dass die GUI sinnvoll **skaliert**: Die Tabelle soll bei Größenänderung wachsen oder schrumpfen; die Schaltflächen sollen immer unten rechts angeordnet sein.

   (3 Punkte)

2. Fügen Sie ein Hauptmenü hinzu. Es soll mindestens zwei Funktionen enthalten:

    a. Die Funktion **File > Exit** beendet das Programm.
    b. Die Funktion **Entry > Add…** hat den gleichen Effekt wie ein Klick auf „Add entry“.

    (2 Punkte)

3. Implementieren Sie die Funktion „Add entry…“ aus. Bei Aufruf der Funktion soll ein eigens entwickelter modaler Unterdialog erscheinen, der die Angabe von Name und Punktzahl erlaubt. Bei Bestätigung wird der Eintrag dann der Liste hinzugefügt.

    Hier darf kein Standard-Dialog verwendet werden.

    (8 Punkte)

4. Fügen Sie eine Funktion „Edit entry…“ hinzu, um einen ausgewählten Eintrag zu bearbeiten. Für die Funktion soll es eine Schaltfläche und einen Menüeintrag geben. Sie ist nur verfügbar, wenn tatsächlich ein Eintrag ausgewählt ist.

    Bei Aufruf der Funktion soll ein modaler Unterdialog erscheinen, der die Angabe von Name und Punktzahl erlaubt. Bei Bestätigung wird der Eintrag dann in der Liste aktualisiert (Tipp: Dialog aus vorheriger Aufgabe wiederverwenden).

    Hier darf *kein* Standard-Dialog verwendet werden.

    (8 Punkte)

5. Fügen Sie eine Funktion „Delete entry…“ hinzu, um einen ausgewählten Eintrag zu löschen. Für die Funktion soll es eine Schaltfläche und einen Menüeintrag geben. Sie ist nur verfügbar, wenn tatsächlich ein Eintrag ausgewählt ist.

    Bei Aufruf der Funktion soll eine Sicherheitsabfrage erscheinen. Wird diese bestätigt, wird der Eintrag aus der Liste gelöscht.

    Für die Sicherheitsabfrage darf ein Standard-Dialog verwendet werden.

    (10 Punkte)

6. Fügen Sie eine Funktion „Export“ hinzu, um die Liste in das CSV-Format zu exportieren. Das CSV-Format ist ein Textformat, bei dem Tabellenspalten durch Kommas getrennt sind. Das obigen Beispiel kann z.B. wie folgt aussehen:

```
Player,Score
Alice,47
Bob,51
Charles,23
```

   Für die Funktion soll es einen Menüeintrag geben.

   Bei Aufruf der Funktion darf der/die Nutzer:in eine Datei auswählen (Standard-Dialog verwenden). Wird eine Datei ausgewählt, werden alle Datensätze im Programm in die Datei exportiert. Tritt ein Fehler auf, erscheint eine sinnvolle Fehlermeldung.

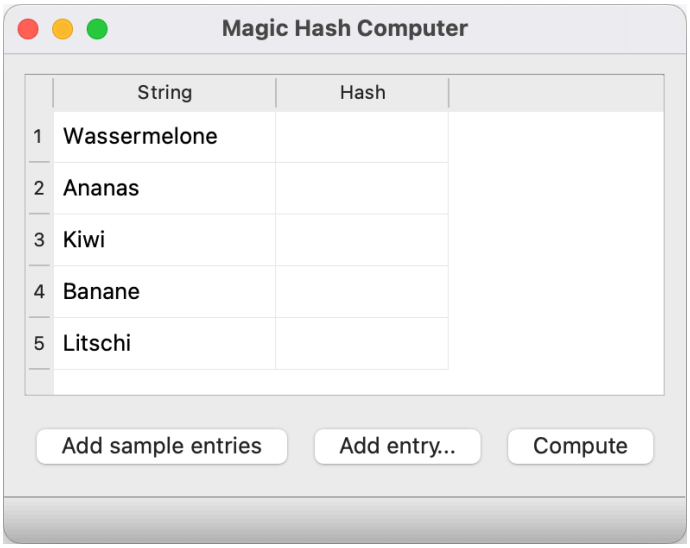
   (10 Punkte)

Aufgabe 3: Nebenläufigkeit (20 Punkte, +5 Bonuspunkte)

Die Lösung der Aufgabe muss im ZIP-Archiv im Ordner A3 gespeichert werden.

Die *SlowAF Software AG* hat kürzlich mit seinem Programm *Magic Hash Computer* den Negativpreis für das hakeligste Programm der Welt bekommen. Sie wurden beauftragt, das Programm responsiver zu gestalten.

Es sieht wie folgt aus:



Das Programm erlaubt die Erfassung von Strings und die Berechnung spezieller Hashwerte für diese Strings. Die Schaltflächen am unteren Rand besitzen die folgenden Funktionen:

- **Add sample entries** fügt Beispieleinträge zum Testen hinzu.
- **Add entry** erlaubt es Nutzer:innen, einen selbst definierten String hinzuzufügen.
- **Compute** löst die Berechnung der Hash-Werte aus. Nach der Berechnung wird für jeden String in der Liste der Hash-Wert angegeben.

Aktuell blockiert das Programm, sobald man auf **Compute** klickt.

Der Quellcode des Programms ist dieser Klausur beigefügt. Lösen Sie die folgenden Aufgaben:

a) Schreiben Sie das Programm so um, dass der GUI-Thread während der Berechnung der Dateigrößen nicht mehr blockiert. Dabei soll **Threading** über eine separate **Arbeitsklasse** eingesetzt werden. Ein Ableiten von QThread ist nicht zulässig.

Hinweis: Die Funktion **computeHash** darf nicht verändert werden. Eine Änderung der Funktion – auch in geringem Umfang – führt zu 0 Punkten für die gesamte Aufgabe.

(10 Punkte)

b) Erweitern Sie das Programm so, dass während der Berechnung der aktuelle Fortschritt in Form einer Progress Bar angezeigt wird. Die Progress Bar sollte sich jedes Mal ändern, wenn der Hash einer Datei berechnet wurde. (10 Punkte)

c) **Bonusaufgabe**: Fügen Sie eine Möglichkeit hinzu, eine laufende Berechnung abzubrechen, und zwar so, **dass danach wieder eine erneute Berechnung gestartet werden kann**.
(+5 Punkte)