Prämienrechner

Prüfung 1

Fach	CS2 Programmierung
Klasse	20I, 21NI
Lehrperson	Sagi Nedunkanal
Semester	FS 2022
Prüfungsbeginn	02.06.2022
Abgabetermin	11.06.2022, 24:00 Uhr

1. Ausgangslage

Es soll ein Prämienrechner realisiert werden, der abhängig von Franchise und Versicherungsmodell eine Krankenversicherungsprämie ausrechnet.

Die Aufgabenstellung ist nicht in allen Details ausspezifiziert. Bei Unklarheiten wird erwartet, dass beim Dozenten nachgefragt wird.

2. Ausbaustufen

Es reicht die Standard-Version zu implementieren, um die volle Punktzahl zu erreichen. Die *Pro-Version* ist als Herausforderung für die fortgeschrittenen Studierenden gedacht und geben keine Zusatzpunkte. Nur die Standard-Version ist notenrelevant.

3. Anforderungen

3.1. Benutzeroberfläche

Standard:
Schreibe eine WPF-Applikation. Die Applikation soll drei Bereiche darstellen:

1. Einen Bereich für die Erfassung neuer Versicherungsnehmer
2. Einen Bereich mit einer Auflistung aller Versicherungsnehmer, und bei der Selektion eines Versicherungsnehmers eine Anzeige der Details
3. Einen Dashboard-Bereich für die Anzeige von Kennzahlen

Wähle geeignete GUI-Controls, um die Bedienoberfläche optisch ansprechend zu gestalten und intuitiv zu bedienen. Verwendet das MVVM-Muster.

Pro:
Verwende Layout-Manager, um ein GUI zu entwerfen, welches sich an die Veränderung der Fenstergrösse anpasst.

3.2. Versicherungsnehmererfassung

Standard:
Man kann Versicherungsnehmer erfassen, mutieren und löschen. Ein Versicherungsnehmer hat die Attribute: Vorname, Nachname, Franchise, Versicherungsmodell (enum) und die berechneten Werte für Monatsprämie und Jahresprämie.

Pro:
Für die Versicherten kann ein Bild aus einer Liste von Avataren ausgewählt werden. Das Bild muss nicht aus der Datenbank kommen. Stattdessen können Bilder als Teil der Solution verwaltet werden und die DB führt nur die Bildernamen.

3.3. Versicherungsnehmeranzeige

Standard:
Die Versicherungsnehmer werden in einer Liste angezeigt. Beim Selektieren eines Listeneintrags werden nebenan die Detailinformationen angezeigt. Änderungen der Werte in der Detailansicht führen zur Aktualisierung der Listenansicht (Databinding).

3.4. Dashboard

Standard:

Das Dashboard zeigt die monatlich eingenommenen Prämien kumuliert an (siehe Tabelle unten) und die totale Anzahl Versicherungsnehmer.

```
Jan.|Feb.|März|Apr.|Mai|Jun.|Jul.|Aug.|Sept.|Okt.|Nov.|Dez.|
------------------------------------------------------------
  10|  20|  30|  40| 50|  60|  70|  80|   90| 100| 110| 120|

Anzahl Versicherungsnehmer: 3
```

Pro:
Die kumulierten Prämien werden als Grafik dargestellt.
Es werden weitere Kennzahlen angezeigt wie Erwartete Prämieneinnahme, Tatsächliche Prämieneinnahmen, Monatliche Auszahlungen, Monatliches Saldo.

3.5. Versicherungsprämien berechnen

Für die Berechnung der Prämien soll folgende Tabelle benutzt werden:

Franchise	**300**	**500**	**1'000**	**1'500**	**2'000**	**2'500**
Monatliche Prämie	302	293	266	239	211	184
Modell						
- keins	-0%	-0%	-0%	-0%	-0%	-0%
- HMO	-6%	-6%	-6%	-6%	-6%	-6%
- Hausarzt	-12%	-12%	-12%	-12%	-12%	-12%
- Telmed	-15%	-15%	-15%	-15%	-15%	-15%

4. Datenhaltung

Standard:
Die Daten sollen in einer «SQL Server»-Datenbank gehalten werden. Der Zugriff soll mit Entity Framework durchgeführt werden.

Der Name der Datenbank soll via `App.config` wie folgt benannt werden:
«`Pruefung1<Vorname>`» => Beispiel: «`Pruefung1Sagi`»

Es soll die EF-Code-first-Methode verwendet werden. Die Datenbank soll durch Ausführen der Migrationsskripte angelegt werden. Wenn die Applikation zum ersten Mal startet, sollen die Testdaten abgefüllt werden.

Zeichne eine Skizze des Entitätendiagramms. Das Diagramm kann von Hand gezeichnet und als Foto in der Dokumentation abgelegt werden.

Pro:
Die Bilder für die Avatare können lokal gehalten werden und müssen nicht aus der Datenbank kommen.

5. Qualitätssicherung

5.1. Testdaten

Standard:
Beim Applikationsstart sollen bereits mindestens 3 Versicherungsnehmer erfasst sein.

Jeder Versicherungsnehmer soll eine andere Franchise und ein anderes Versicherungsmodell haben.

5.2. Unittests

Standard:
Es sollen mindestens drei Unittest für die Prämienberechnung geschrieben werden.
Beispiel:

// Arrange	Der Versicherungsnehmer wählt die Franchise 300 und Versicherungsmodell «keine».
// Act	Die Monatsprämie wird berechnet.
// Assert	Die erwartet Ergebnis ist: Monatsprämie = CHF 302.-

Pro:
Schreibe weitere Unittests, um die Testabdeckung zu erhöhen.
Gehe testgetrieben nach dem Red-Green-Refactor-Muster vor:

1. *Red: Überlege Dir zuerst, wie die Testfälle aussehen könnten und schreibe Tests, die kompilieren, aber fehlschlagen (weil noch keine Implementierung existiert).*
2. *Green: Implementiere die Funktionalität, so dass der Testfall grün wird.*
3. *Refactor: Verbessere die Struktur des Codes (Codeduplikation entfernen, Methoden verkleinern usw.) Die Unittests bieten das Sicherheitsnetz für sorgenloses Refactoring.*

5.3. Versionsverwaltung

Verwende die Git-Versionsverwaltung, um regelmässig Sicherungen des aktuellen Standes zu erstellen (Commit) und hochzuladen (Push). Mindestens drei Commits sind gefordert.

6. Dokumentation

Standard:
Es soll eine minimale Dokumentation erstellt werden:

- Lösungskizze (kann von Hand gezeichnet, fotografiert und als JPEG abgelegt werden; kein im Nachhinein generiertes Klassendiagramm).
- Screenshots aller Fenster.
- Die Skizze des Entitätendiagramms.
- Anleitung für die DB-Erstellung, damit der IT-Betrieb die Datenbank anhand der Migrationsskripte anlegen kann, sprich die Konsolenbefehle, die Visual Studio ausgeführt werden müssen.

Eine Textdatei und ein paar JPEGs reichen. Es wird keine ausführliche Projektdokumentation verlangt!

7. Bewertungskriterien

1. Umsetzung der funktionalen Anforderungen (1/4)
2. Bewertung des GUIs (1/4)
3. Bewertung des Codes (1/4)
4. Testen der Applikation (1/4)

- Erlaubt: Internet als Quelle, Beispiele aus dem Unterricht usw.
- Nicht erlaubt: Kopieren grösserer Codemengen, Mitarbeit Dritter usw.
  - Selbstkontrolle: «Kann ich jede Zeile Code erklären?»
- Missachtung der Prüfungsregeln führen zu einer Note 1.
- Verspätete Abgabe führt zu Notenabzug:
  - Bis 6h Verspätung → 0.5 Note Abzug
  - Bis 12h Verspätung → 1.0 Note Abzug
  - >12h Verspätung → Note 1.0

8. Abgabe

Die Applikation wird via GitHub abgegeben:

1. Den Quellcode ablegen unter `CS2\02 Prüfungen\Prüfung 1\Quellcode\Prämienrechner`
2. Dabei die vorbereitete Solution-Datei verwenden: «`Vorname Nachname CS2 Prüfung 1.sln`»
3. Das Kompilat als ZIP ablegen unter `CS2\Prüfungen\Prüfung 1\Kompilat\Prämienrechner.zip`
4. Die Dokumentation ablegen unter `CS2\Prüfungen\Prüfung 1\Dokumentation`
5. Alles hochladen

Verwende die Versionsverwaltung, um regelmässig Sicherungen des aktuellen Standes zu erstellen (Commit) und hochzuladen (Push).