SOW-Projekt

Szenario

Das Unternehmen „XFirm“ möchte eine individuelle Unternehmenssoftware zur Verwaltung ihrer Mitarbeiter einsetzen.

Im Rahmen dieses Projekts soll das Model dieser Anwendung erstellt und über eine Konsolenanwendung getestet werden. Die Erstellung der kompletten MVC-Anwendung mit einer graphischen Benutzeroberfläche erfolgt später.

Die Anwendung wird in **Gruppenarbeit (2 Personen + eine Einzelperson)** realisiert werden. Sie soll mit der Programmiersprache Java erstellt werden**.**

Briefing

Die Mitarbeiterstammdaten wie Name, Adresse, Geburtsdatum und Eintrittsdatum werden in der Software gespeichert. Die Echtdaten werden in Form von Dateien vorliegen und müssen in die Anwendung portiert werden. Das Unternehmen schließt mit den Mitarbeitern unterschiedliche Arbeitsverträge ab. Der Lohn eines Arbeiters berechnet sich aus Stundenlohn, Anzahl der gearbeiteten Stunden, Überstundenzuschlag, Anzahl der Überstunden, plus eine Schichtzulage. Das Gehalt eines Angestellten errechnet sich aus der Summe von Grundgehalt, Ortszuschlag und Zulage. Ein Manager erhält zu einem Fixgehalt eine Provision. Diese berechnet sich aus einem mit dem Mitarbeiter verhandelten Prozentsatz seines erzielten Umsatzes. Das Gehalt eines Geschäftsführers ergibt sich aus seinem Managergehalt und einer Geschäftsführerzulage.

Die Software ermöglicht die elektronische Aufnahme eines neuen Mitarbeiters im System. Dabei soll automatisch sein Mitarbeiterkennzeichen erzeugt werden. Das 10-stellige Kennzeichen beinhaltet folgende Informationen:

- Hinweis auf das Geschlecht
- Geburtsjahr
- Anfangsbuchstaben von Vor- und Zunamen
- Laufende Nummer
- Hinweis auf die Mitarbeiterart

Wenn ein Mitarbeiter ausscheidet, wird er aus der aktuellen Mitarbeiterliste entfernt und in der Liste der ehemaligen Mitarbeiter aufgenommen. Neue Mitarbeiter können in das Unternehmen aufgenommen werden. Allerdings möchte das Unternehmen eine Mitarbeitergröße (hier testweise 20) nicht überschreiten.

Gehaltserhöhungen können sowohl für einzelne Mitarbeiter als auch für Mitarbeitergruppen (z.B. alle Angestellte) durchgeführt werden. Dabei wird das Grundgehalt um einen gewissen Prozentsatz erhöht. Bei Angestellten ist die Höhe des Prozentsatzes 5%, bei Managern liegt die Obergrenze bei 10%. Mitarbeiter, die auf Stundenlohnbasis bezahlt werden, erhalten eine individuell vereinbarte Erhöhung ihres Stundenlohns.

Für besonders gut geleistete Arbeit belohnt das Unternehmen ihre Mitarbeiter mit Prämien. Jede Gehaltserhöhung und Prämienzahlung soll mit Datum in der elektronischen Mitarbeiterakte festgehalten werden.

Die Software soll die Gehaltsabrechnung für einen bestimmten Monat durchführen. Sie kann mit Angabe des Monats angestoßen werden. Dazu werden alle Mitarbeiternamen mit -kennzeichen und dem aktuellen Monatsgehalt in eine Datei geschrieben, die man sich über die Anwendung anzeigenlassen kann.

Die elektronischen Mitarbeiterakten sollen auf dem Bildschirm angezeigt werden können.

Weiterhin gibt die Software Auskunft über

- den oder die Mitarbeiter mit dem niedrigsten Monatslohn
- die Verdienstspanne im Unternehmen
- über den jeweiligen Geschlechteranteil im Unternehmen
- die Höhe des Durchschnittsgehalts

- die Anzahl der Mitarbeiter
- das Durchschnittsalter der Mitarbeiter
- den Mitarbeiter mit der längsten Firmenzugehörigkeit

Bei der Entwicklung des Models wird besonderen Wert auf die **Programmarchitektur** gelegt. Dabei gelten die folgenden **Produktionsvorgaben**:

- Aus Gründen der Produktivität, Wartbarkeit und Erweiterbarkeit soll **Vererbung** von Klassen eingesetzt werden.
- Die Beziehungen zwischen den Objekten sollen durch entsprechende **Assoziationen** abgebildet werden.
- Die Instanzvariablen werden über setter- und getter-Methoden gesetzt. Dabei werden natürlich **Plausibilitätsprüfungen** vorgenommen und gegebenenfalls **Exceptions** ausgelöst.

- Mitarbeiterdaten sollen in **Textdateien** gepflegt werden. Die Mitarbeiterdaten sollen bei Programmstart aus der Datei gelesen werden. Die aktuellen Mitarbeiterdaten sollen zum Programmende wieder in eine Datei geschrieben werden. Arbeiten Sie sich dafür in die Klassen BufferedReader und BufferedWriter aus dem Package java.io ein. Mit Hilfe dieser Klassen lassen sich Daten aus einer Textdatei lesen bzw. in eine Textdatei schreiben.

Arbeitsauftrag 1: Planung

A.) Usecasediagramm

Zeichnen sie auf Papier (Foto mit ins Repository) eine Usecasediagramm, welche mindestens 4 Usecases und die Akteure zeigt. Achten Sie auf die Syntax

B.) Klassendiagram

Erstellen Sie eine **Planung** für die zu erstellende Anwendung. Machen Sie sich dabei Gedanken über die Programmarchitektur und erstellen Sie zur Vorbereitung der Programmierung mit dem UMLVioletEditor oder Umlet ein **Klassendiagramm** für die Anwendung.

Bitte beachten Sie die Art der Verbindungen und Assoziationen zwischen den Klassen: Normale Assoziation, Aggregation oder Komposition mit oder ohne Multiplizität.

B.) Aktivitätsdiagramm

Erzeugen Sie auch ein Aktivitätsdiagramm für die Ermittlung des Gehalts. Diese soll als Diagramm aufzeichnen, wie sie aus dem Grundgehalt und mit den zusätzlichen Erhöhungen und Prämien das Gehalt eines Mitarbeiters berechnen. Zeichnen Sie dies auch mit DrawIO, Violet oder Umlet.

Wo findet im Programm diese Struktur ihren Einsatz?

Arbeitsauftrag 2: Programmierung

Realisieren Sie die geplante Anwendung. **Beachten** Sie dabei die **Produktionsvorgaben**.

Alle Anforderungen an die Anwendung sollen über ein Menü im Rahmen einer Konsolenanwendung getestet werden können. Erstellen Sie für die Uploads ein **ZIP-File** des kompletten **Projektordners** und laden Sie das ZIP-File in Teams hoch.

Arbeitsauftrag 3: Dateiverarbeitung

Erstellen Sie eine Textdatei mit Testdaten für mindestens 12 Mitarbeiter. Informieren Sie sich darüber wie Daten mithilfe der Klasse BufferedReader aus einer Textdatei gelesen bzw. mithilfe der Klasse BufferedWriter in eine Textdatei geschrieben werden.

Bauen Sie das Laden der Daten aus einer Datei und das Speichern der aktuellen Daten in eine Datei in Ihre Anwendung ein.

WICHTIGE VORRAUSSETZUNG FÜR DIE BENOTUNG

Speichern Sie unterschiedliche Commits zum kompletten im Repository. Zumindest aber zu folgenden Zeiten. Laden Sie Markus Röder und Frau Schicha von Anfang an mit in das Repository ein. Stellen Sie sich am besten einen Alarm für diese Zeiten.

1. **Upload: Montag, den 01.07.24 bis 10:30 Uhr.**
2. **Upload: Montag, den 01.07.24 bis 13:30 Uhr.**
3. **Upload: Montag, den 01.07.24 bis 15:30 Uhr.**

4. **Upload: Dienstag, den 02.07.24 bis 10:30 Uhr.**
5. **Upload: Dienstag, den 02.07.24 bis 13:30 Uhr.**
6. **Upload: Dienstag, den 02.07.24 bis 15:30 Uhr.**

7. **Upload: Mittwoch, den 03.07.24 bis 10:30 Uhr.**
8. **Upload: Mittwoch, den 03.07.24 bis 13:30 Uhr.**
9. **Upload: Mittwoch, den 03.07.24 bis 15:30 Uhr.**

Bewertungskriterien

- Ein lokales und remote Repository angelegt
- Einladung aller Beteiligten ins Remote Repository
- Zumindest Commits und push zu festgelegten Zeiten (von beiden Studierenden)

Arbeitsauftrag 1: Planung

- Usecasediagramm
- Klassendiagramm
- Aktivitätsdiagramm

Arbeitsauftrag 2: Programmierung

- Anlegen unterschiedlicher Klassen
- Vererbung
- Abbilden der Beziehungen durch Assoziationen
- setter/getter-Methoden
- Plausibilitätsprüfungen bei den setter-Methoden
- Implementierung der notwendigen Methoden und Operatoren in den jeweiligen Klassen
- Menüführung auf Konsolenebene

Arbeitsauftrag 3: Dateiverarbeitung

- Festlegung des Formats der Textdatei
- Erzeugung einer Textdatei mit Testdaten
- Dateiverarbeitung in die Anwendung integrieren