Wir werden nun die Risikoverwaltung vollständig als JavaFX-Anwendung realisieren. Die Risikoverwaltung soll dabei eine Drei-Schichtenarchitektur besitzen. Treffen Sie zunächst die folgenden Vorbereitungen, um eine JavaFX-GUI-Anwendung realisieren zu können. Gehen Sie bei der Programmierung schrittweise vor. Die ersten vier Arbeitsschritte sind erforderlich, um eine klare Trennung zwischen Fachlogik und Interaktion zu erhalten.

1. Überschreiben Sie in den konkreten Risiko-Klassen die `toString()`-Methode der Klasse `Object`. Geben Sie über die `toString`-Methode die Informationen zurück, die auch über die `druckeDaten()`-Methode ausgegeben wird. Vermeiden Sie dabei möglichst doppelten Code.

2. Ergänzen Sie die konkreten Risiko-Klassen um einen Standardkonstruktor und um getter- und setter-Methoden für die fachlichen Attribute.

3. Ändern Sie die Methode „`sucheRisikoMitMaxRueckstellung`“ der Klasse `Risikoverwaltung`. Die Methode soll das gefundene Objekt als Rückgabewert liefern. Die Methode darf keine direkte Ausgabe vornehmen.

4. Ergänzen Sie die Klasse `Risikoverwaltung` um die Methode `public Iterator<Risiko> iterator()`. Diese Methode liefert ein Iterator-Objekt, um die interne Risikoliste zu durchlaufen.

Erstellen Sie dann die Grundstruktur der 3-Schichten-Architektur und bereiten Sie danach die Integration der in Praktikum 8 erstellten Fensterklassen vor.

5. Erstellen Sie die drei Pakete `pk1.rv.gui`, `pk1.rv.fachlogik` und `pk1.rv.datenhaltung`.

6. Nehmen Sie die Implementierung der Fenster aus dem Praktikum 8 und integrieren Sie den Code in das Projekt der Risikoverwaltung (falls noch nicht geschehen). Verteilen Sie alle Klassen der Risikoverwaltung auf die geeigneten Pakete.[1]

7. Falls im Konstruktor der Klasse `RiskoErfassungView` eine Referenz auf eine konkrete Risiko-Instanz übergeben wird, dann sollen die zugehörigen Attributwerte in die Eingabefelder eingeblendet werden. Falls der Button „Neu“ geklickt wird, dann werden die Eingabedaten in das `Risiko`-Objekt übertragen.

8. Verfahren Sie analog mit den beiden übrigen Erfassungs-Fenstern.

Kümmern Sie sich nun um die Anbindung der Datenhaltung. Beachten Sie dabei die Pakete.

9. Trennen Sie die Fachkonzeptklassen von der Datenhaltung über ein DAO. Verwenden Sie dabei die folgende Schnittstelle:

[1] Die alte `Menu`-Klasse werden Sie nicht mehr benötigen.

```
public interface IDao {
   void speichern(List<Risiko> liste) throws PersistenzException;
   List<Risiko> laden() throws PersistenzException;
}
```

Dabei werden technische Ausnahmen auf die Ausnahme `PersistenzException` abgebildet.

10. Implementieren Sie die Schnittstelle `IDao`, indem Sie die Serialisierung aus dem Praktikum 6 einbinden. Schreiben Sie das statische Attribut für die ID-Verwaltung in den Ausgabestrom (falls noch nicht geschehen).[2] Ermitteln Sie später dieses Attribut über die eingelesene ID.

Sorgen Sie abschließend für die Ereignisbehandlung.

11. Versehen Sie das Menü aus dem Praktikum 8 mit einer geeigneten Ereignisbehandlung. Über die Menüs sollen die folgenden Funktionen aufgerufen werden können (die Funktionen haben Sie bereits in den bisherigen Praktika implementiert):

    (a) *Laden* : Eine gespeicherte Risikoliste wird geladen

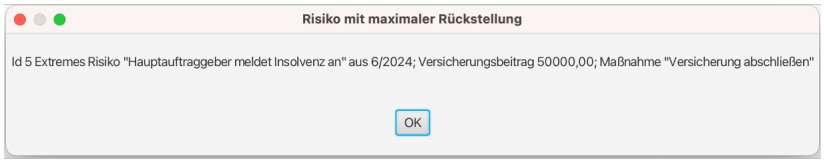
    (b) *Speichern* : Die aktuelle Risikoliste wird persistent gespeichert (hier: serialisiert).

    (c) *Risikoliste in Datei* : Die komplette Risikoliste wird in lesbarer Form in eine Datei geschrieben. Der Dateiname wird vorher über ein Fenster abgefragt. Sie können die Hilfsklasse `InputView` verwenden (siehe Ilias).

    (d) *Neues Risiko* : Ein neuer Risiko-Datensatz wird angelegt. Die Eingabe der Daten erfolgt über eine Instanz der Klasse `RisikoErfassungView`. Nach der Analyse des Risikowerts wird nachfolgend ggf. ein weiteres Fenster geöffnet, um die zusätzlichen Daten für ein inakzeptables Risiko oder ein extremes Risiko abzufragen (siehe Praktikum 3 und 8).

[2] Aufgrund der Aufteilung der unterschiedlichen Risikoklassen erfolgt die Vergabe der IDs bei der Eingabe von Risiken nicht mehr lückenlos. Der ID-Zähler kann daher nicht aus der Listenlänge ermittelt werden.

(e) *Risiko mit maximaler Rückstellung* : Ein Risiko mit maximaler Rückstellung wird gesucht und ausgegeben (Sie können die Hilfsklasse `MessageView` verwenden). Beispiel:



(f) *Summe aller Rückstellungen*: Die Summe aller Rückstellungen wird berechnet und in einem Fenster angezeigt.

Das Hauptfenster werden Sie im nächsten Praktikum fertigstellen.

Sorgen Sie im Fall von Ausnahmen für eine geeignete Meldung an den Anwender. Achten Sie darauf, die Fachkonzeptklassen geeignet an die GUI und die Datenhaltung anzubinden. Sie können in der GUI-Schicht mit einer zusätzlichen Klasse `Controller` arbeiten, welche den Aufruf der einzelnen Fenster (*Views*) steuert. Der *Controller* kann die Fenster und die benötigten Methoden der Klasse `Risikoverwaltung` aufrufen.[3]

[3]Die Klassen `InputView` und `MessageView` aus dem Ilias bieten einen einfachen Ersatz für die Swing-Methoden `JOptionPane.showInputDialog` und `JOptionPane.showMessageDialog`.