gds2 Gottlieb-Daimler-Schule 2
Technisches Schulzentrum Sindelfingen
mit Abteilung Akademie für Datenverarbeitung

Sequentielles Lesen von Textdateien

Ziel: Kennenlernen elementarer Methoden zum sequentiellen, zeichenweisen Lesen von Textdateien unter Berücksichtigung der Kodierung der zu lesenden Textdatei.

1.1. Es soll eine Java Konsolanwendung erstellt werden, die folgende Funktionalität besitzt:

- Die Anwendung liest einen Dateipfad vom Benutzer ein.
- Es wird versucht ein BufferedReader Objekt für diese Datei zu erzeugen. Im Fehlerfall bekommt der Anwender eine Fehlermeldung.
- Wurde das BufferedReader Objekt erfolgreich erzeugt, so wird die Datei Zeichen für Zeichen gelesen und am Bildschirm angezeigt. Gleichzeitig wird mitgezählt, wie viele Zeichen die Datei insgesamt enthält, wie viele davon Buchstaben sind, wie viele Ziffern und wie viele sonstige Zeichen. Diese Informationen werden am Ende ebenfalls am Bildschirm angezeigt. (siehe dazu auch Lsg_03_JavaStandardKlassen_A1_1BisA1_5 Aufg. 1.3)

1.2. Die Problemstellung von Aufgabe 1.1 soll jetzt mit einem stärker objektorientierten Ansatz gelöst werden. Das heißt:

- Das Lesen und die eigentliche Auswertung soll in einer Fachkonzeptklasse erfolgen. Die Ergebnisse der Auswertung werden in einem Objekt dieser Fachkonzeptklasse gespeichert und können mit get-Methoden von dort abgerufen werden.
- Für das Lesen und die Auswertung von Dateien besitzt diese Fachkonzeptklasse eine Objektmethode mit folgender Signatur:

```
public boolean werteDateiAus(String sPfad, Charset cs)
```

  Der Rückgabewert signalisiert, ob beim Zugriff auf die Datei ein Fehler aufgetreten ist. Exceptions werden nicht weitergereicht, sondern ebenfalls in einem Attribut gespeichert. Gibt die Methode false zurück, so kann die Exception über eine get-Methode abgerufen und die Fehlermeldung angezeigt werden.
- In einer UI oder GUI Klasse wird diese Fachklasse jetzt angewendet. Es soll die gleiche Bildschirmanzeige generiert werden, wie in Aufgabe 1.1.

1.3. Entwickeln Sie ein Programm, das den Inhalt zweier Textdateien vergleichen kann. Das Programm soll die gelesenen Zeichen mitzählen. Stellt das Programm an einer Stelle einen Unterschied fest, soll es abbrechen und ausgeben, bei welchem Zeichen der Unterschied aufgetreten ist.

Bildschirmausgabe 1

```
Bitte 1. Dateinamen eingeben: test1.txt
Bitte 2. Dateinamen eingeben: test2.txt


 Pos. |    test1.txt |    test2.txt
------+--------------+--------------
    1 |            G |            G
    2 |            D |            D
    3 |            S |            S
    4 |            2 |            2
    5 |              |            !

Unterschied beim 5. Zeichen!
```

Bildschirmausgabe 2

```
Bitte 1. Dateinamen eingeben: test1.txt
Bitte 2. Dateinamen eingeben: test3.txt


 Pos. |    test1.txt |    test3. txt
------+--------------+--------------
    1 |            G |            G
    2 |            D |            D
    3 |            S |            S
    4 |            2 |            2
    5 |              |

Vergleich OK!
```

1.4. **Problembeschreibung**: Zu Abrechnungszwecken wurden in der UTF-8 kodierten Textdatei `kosten.txt` für alle Projektmitarbeiter Arbeitszeitdaten und Kostensätze abgespeichert. Die Datensätze haben die Form:

PersonalNr;Datum;Arbeitszeit in Stunden;Stundensatz

Beispielausschnitt aus `kosten.txt`:

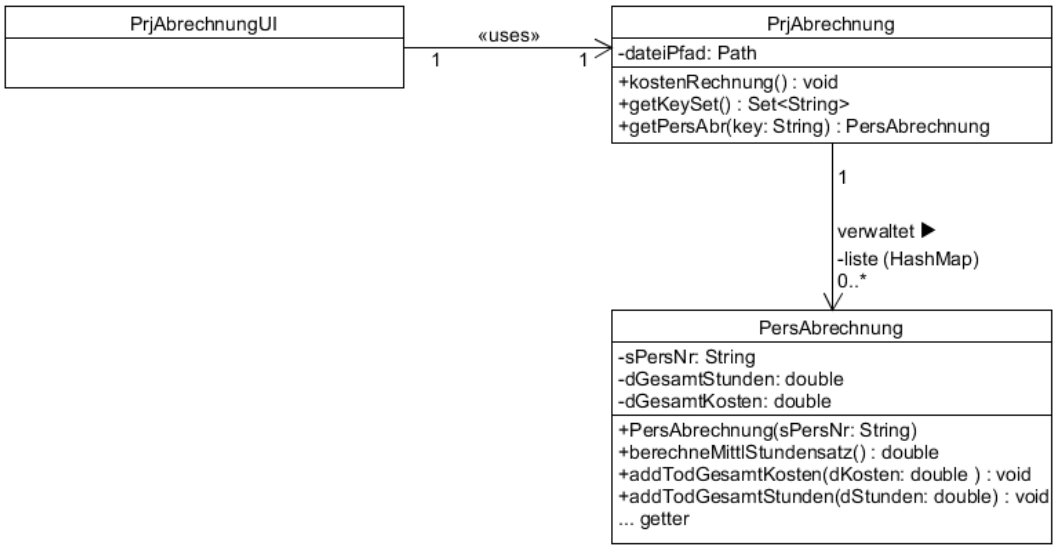
```
127;13.05.2016;3;120
274;13.05.2016;4;100
127;14.05.2016;5;135
```

Jeder Mitarbeiter ist durch seine Personalnummer identifiziert. Ein Mitarbeiter kann zu verschiedenen Zeitpunkten gearbeitet haben und taucht dann deshalb in mehreren Datensätzen in der Datei auf. Wenn der Mitarbeiter in verschiedenen Funktionen am Projekt gearbeitet hat, dann wird eventuell ein unterschiedlicher Stundensatz für ihn abgerechnet.

Aufgabe: Sie sollen eine Klasse zur Auswertung der Datei entwickeln, die folgenden Anforderungen genügt:

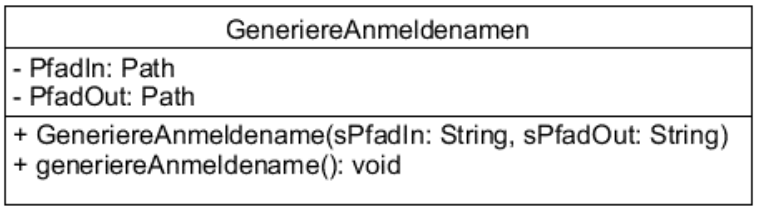
- Der Pfadstring für die Datei wird mit einem parametrierten Konstruktor festgelegt.
- Die Klasse besitzt eine Objektmethode `public void kostenrechnung(String sPersonalNr)` , die zu einer `sPersonalNr` die gesamte Arbeitszeit, die gesamten Arbeitskosten und den Mittelwert des Stundensatzes ermittelt und in Attributen der Klasse speichert werden, so dass diese dann mit get-Methoden abgerufen werden können.
  Tritt in der Methode eine Exception auf, so wird diese zum Aufrufer weitergegeben.
- In einer UI oder GUI Klasse wird diese Fachklasse jetzt angewendet und die Ergebnisse am Bildschirm angezeigt.

1.5. **Zusatzaufgabe**: Die Klasse von 1.4 soll so verändert werden, dass die Kostenrechnung für jeden Mitarbeiter durchgeführt wird, der sich in der Datei befindet. In der UI oder GUI Klasse werden dann die Ergebnisse für alle Mitarbeiter angezeigt.
Hinweise (s. auch nächste Seite):



- Es bietet sich an, für jede gefundene Personalnummer ein „PersAbrechnung“-Objekt anzulegen, indem die Werte dieser Person gespeichert werden.
- Dieses Objekt speichert man in einer HashMap mit der Personalnummer als Key. Die HashMap bietet eine effiziente Möglichkeit festzustellen, ob für eine bestimmte Personalnummer bereits ein Eintrag vorhanden ist.
- Damit die UI-Klasse alle Ergebnisse anzeigen lassen kann, sollte sie ein KeySet für die HashMap benutzen. Dieses KeySet sollte die UI-Klasse aus der Klasse `PrjAbrechnung` abrufen können (s. Klassendiagramm und s. Skript zu Collections).

1.6. Es soll eine Klasse erstellt werden, die aus einer Eingabedatei Nachname und Vorname von Benutzern liest und daraus Loginnamen erzeugt und in einer Ausgabedatei speichert.



Die Loginnamen werden nach folgenden Regeln gebildet:

- Der Login-Name besteht immer aus acht Zeichen.
- Die ersten sechs Zeichen des Login-Namens sind identisch mit den ersten sechs Zeichen des Nachnamens in Großschreibung.
- Falls der Nachname kürzer als sechs Zeichen ist, werden die fehlenden Zeichen mit dem Zeichen `'x'` aufgefüllt.
- Das 7. und 8. Zeichen des Login-Namens ist der erste und zweite Buchstabe des Vornamens. Annahme: Vornamen haben immer mindestens drei Zeichen.

- Die Eingabedatei enthält pro Zeile einen Nach- und Vornamen durch Leerzeichen getrennt.

- Die erzeugten Login-Namen werden zeilenweise gespeichert.

Jackson structured Programming und Anwendung beim „Gruppenwechsel“

Ziel: Anwendung der strukturierten Methode nach Jackson am Beispiel des sogenannten „Gruppenwechsels“.

2.1. **Problemstellung**: Es soll eine Artikelbestandsliste von Arzneimitteln - sortiert nach Artikelgruppen – erstellt werden.

Vorgehensweise:

- Datenstrukturbaum (in der besprochenen Notation) der Ausgabeliste erstellen. Die Ausgabeliste findet sich in Moodle: `GWListe1.txt`
  Die Eingabedatei findest sich ebenfalls in Moodle: `GWDaten.txt`
- Aus dem Datenstrukturbaum die Programmstruktur herleiten (mit zusätzlichen Operationen), ähnlich wie im Beispiel „Kostenstellen Liste“ besprochen.
- Das entsprechende Programm implementieren und testen.

2.2. Zusatzaufgabe: Zusätzlich soll für jeden Datensatz nach dem Lesen eine Prüfung durchgeführt werden, ob die Daten korrekt sind:

- Gruppennr. und Artnr. dürfen nur Ziffern enthalten
- Preis, Bestand und Mind.-B. müssen numerisch sein.
- Die Spalte zur Giftklasse darf nur die Buchstaben A bis D enthalten
- Artikelbezeichnung und Handelsform müssen mit einem Großbuchstaben beginnen, gefolgt von beliebigen Buchstaben und dem Bindestrich.

- Im Fehlerfall darf der Datensatz nicht verarbeitet werden. Stattdessen wird der fehlerhafte Datensatz mit der Zeilennummer am Bildschirm ausgegen, Bsp.:

```
Fehler in Zeile   1: 01 abcd A Decorpa  Granulat  13.85 010 009
```

- Es bietet sich an, mit der Methode `matches` der Klasse `Pattern` zu Arbeiten: s. Doku.

- Zum Testen kann die Datei `FalscheGWDaten.txt` verwendet werden.

gds2 Gottlieb-Daimler-Schule 2
Technisches Schulzentrum Sindelfingen
mit Abteilung Akademie für Datenverarbeitung

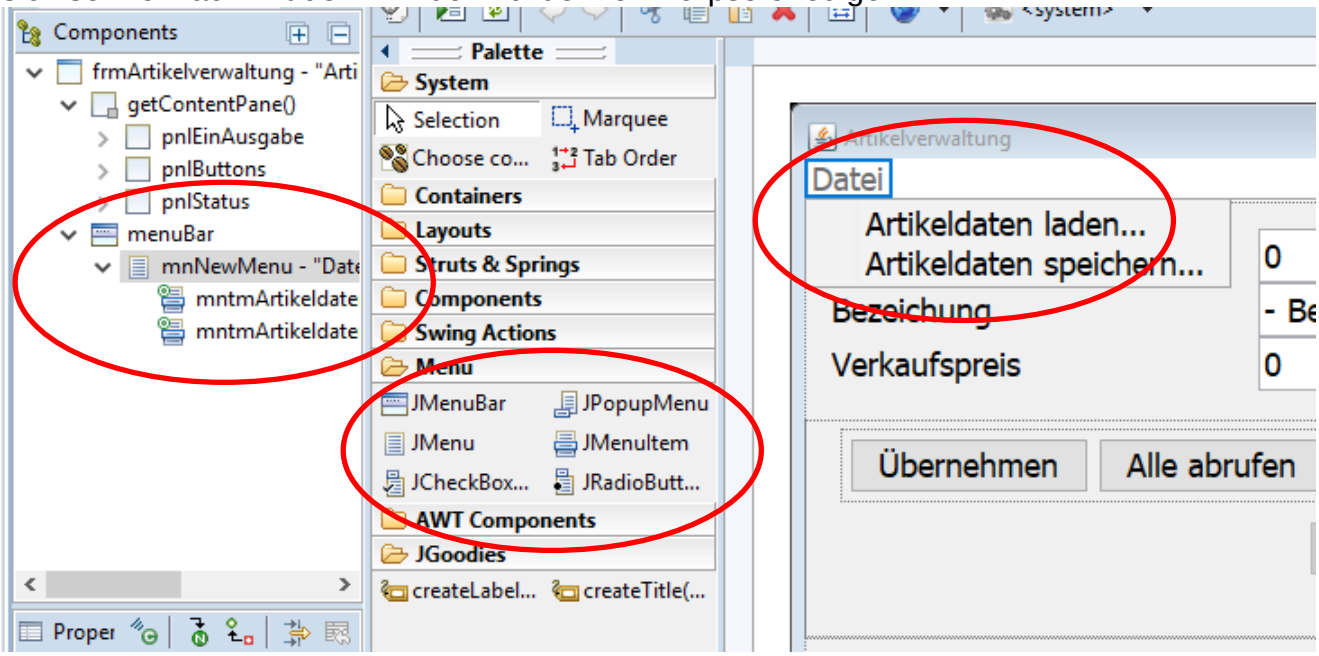
Serialisierung von Objekten und Persistierung im Dateisystem

Ziel: Objektdaten serialisiert im Dateisystem speichern und wieder einlesen

3.1. Im Aufgabenblatt „A_JavaGUI“ in Aufgabe 2.3 haben wir ein bestehendes Programm zur Erfassung und Verwaltung von Artikel Daten um eine GUI erweitert. Der Quellcode dieses Programms findet sich in Moodle.

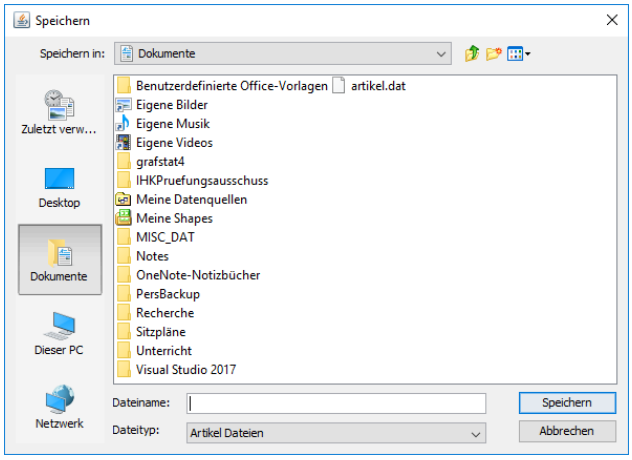
Diese Programm soll nun noch dahin gehend ergänzt werden, dass man die erfassten Artikeldaten in einer Datei persistieren kann, um sie später wieder lesen zu können.

Im ersten Schritt soll die GUI um eine Menüzeile (`JMenuBar`) ergänzt werden. Der `JMenuBar` kann dann ein `JMenu` mit `JMenuItems` hinzugefügt werden. Das ganze lässt sich sehr einfach mit dem WindowBuilder von Eclipse erledigen.



3.2. Der zweite Schritt ist die Programmierung des ActionListeners für den Menüpunkt „Artikeldaten speichern…“. Der Einfachheit halber wählen wir die Variante mit einem anonymen ActionListener: Im WindowBuilder das Kontextmenü von „Artikeldaten speichern…“.aufrufen: → Add Event Handler → Action → Action performed

In der Ereignismethode `actionPerformed` soll zur Auswahl des Dateipfads ein Dateiauswahldialog (JFileChooser) angezeigt werden:
→ siehe nächste Seite



```
public void actionPerformed(ActionEvent e)
{
  JFileChooser dateiAuswahl = new JFileChooser();
  dateiAuswahl.setFileFilter(new FileNameExtensionFilter("Artikel Dateien", "dat"));
  if (dateiAuswahl.showSaveDialog(frmArtikelverwaltung) ==
                                                    JFileChooser.APPROVE_OPTION)
  {
    // Die Datei soll automatisch die Endung .dat erhalten
    artikelDatei =
              Paths.get(dateiAuswahl.getSelectedFile().getAbsolutePath() + ".dat");
    if (artikelListe.speichereArtikelDaten(artikelDatei))
    {
      JOptionPane.showMessageDialog(frmArtikelverwaltung,
          "Artikeldaten wurden gespeichert");
    }
    else
    {
      JOptionPane.showMessageDialog(frmArtikelverwaltung,
          "Artikeldaten konnten nicht gespeichert werden", "Fehler",
          JOptionPane.WARNING_MESSAGE);
    }
  }
```

3.3. Wie man im Quellcode oben sieht, ist die Methode `speichereArtikelDaten()` eine Objektmethode der Klasse `ArtikelContainer` und muss also dort implementiert werden. Diese Methode ist dafür zuständig, die im `ArtikelContainer` gespeicherten Artikelobjekte zu persistieren.
Hinweis: Achten Sie darauf, dass die Klasse Artikel das `Interface Serializable` implementieren muss.
Implementieren Sie die Methode und analysieren Sie die erzeugte Datei mit einem Hex-Editor (siehe Tauschlaufwerk).

3.4. Ergänzen Sie nun das Programm noch um den notwendigen Code, um die Artikelobjekte wieder aus der Datei zu lesen.
Hinweis: Die actionPerformed Methode sieht sehr ähnlich aus, allerdings verwendet man hier die Methode `showOpenDialog` statt `showSaveDialog`.
Außerdem muss man natürlich keine Dateiendung an den Pfad anhängen.

3.5. Es gibt prinzipiell zwei Möglichkeiten den `ArtikelContainer` zu persistieren:

1. Die Objekte der Klasse `Artikel` in einer Schleife einzeln persistieren.
2. Das Objekt der Klasse `ArrayList<Artikel>` komplett persistieren.

Realisieren nun die Variante die Sie in Aufgabe 3.4 nicht gewählt hatten.