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.

```java
/ * TextdateiAuswerten.java
package dateienAufg;
import java.io.BufferedReader;

/ * @author stk
public class TextdateiAuswerten
{
    private Path dateiPfad = null;
    private StringBuffer sbText = new StringBuffer(1000);
    private Exception eineException = null;
    private int iZeichenAnz = 0;
    private int iBuchstabenAnz = 0;
    private int iZiffernAnz = 0;

    public boolean werteDateiAus(String sPfad, Charset cs)
    {
        boolean status = true;
        int iZeichen;
        iBuchstabenAnz = 0;
        iZiffernAnz = 0;
        iZeichenAnz = 0;

        dateiPfad = Paths.get(sPfad);

        try (BufferedReader reader = Files.newBufferedReader(dateiPfad, cs))
        {
            iZeichen = reader.read();

            while (iZeichen != -1)
            {
              iZeichenAnz++;
              if (Character.isLetter(iZeichen))
              {
                  iBuchstabenAnz++;
              }
              else
                  if (Character.isDigit(iZeichen))
                  {
                      iZiffernAnz++;
                  }

              sbText.append((char)iZeichen);
              iZeichen = reader.read();
          }
      }
      catch (Exception oe)
      {
          eineException = oe;
          status = false;
      }
      return status;
  }

  public String getText()
  {
      return this.sbText.toString();
  }

  public Exception getEineException()
  {
      return this.eineException;
  }

  public int getiZeichenAnz()

  public int getiBuchstabenAnz()

  public int getiZiffernAnz()
}
```

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!
```

```
public class DateiVergleich
{
    /**
     * @param args
     */
    public static void main(String[] args)
    {
        int zeichen1;
        int zeichen2;
        Path dateiPfad1 = null;
        Path dateiPfad2 = null;
        String sDateiPfad;
        int position = 0;

        sDateiPfad = Eingabe.getString("Bitte geben Sie den Pfad zur Textdatei 1 ein > ");
        dateiPfad1 = Paths.get(sDateiPfad);
        sDateiPfad = Eingabe.getString("Bitte geben Sie den Pfad zur Textdatei 2 ein > ");
        dateiPfad2 = Paths.get(sDateiPfad);

        try (BufferedReader reader1 = Files.newBufferedReader(dateiPfad1, StandardCharsets.UTF_8);
             BufferedReader reader2 = Files.newBufferedReader(dateiPfad2, StandardCharsets.UTF_8))
        {
            System.out.printf("\n\n Pos. | %12s |%12s\n", dateiPfad1.getFileName(), dateiPfad2.getFileName());
            System.out.printf("------+--------------+--------------\n");

            do
            {
                zeichen1 = reader1.read();
                zeichen2 = reader2.read();
                position++;
                if (zeichen1 != -1 && zeichen2 != -1)
                {
                    if (zeichen1 == '\r')  // Datei enthält Zeilenvorschub
                    {   // Ersetze Zeilenvorschub ('\r''\n') durch darstellbares Zeichen
                        zeichen1 = '←';
                        reader1.read(); // zweites Zeichen lesen und vergessen
                    }
                    if (zeichen2 == '\r')  // wie bei zeichen1
                    {
                        zeichen2 = '←';
                        reader2.read();
                    }

                    System.out.printf(" %4d | %12c |%12c\n", position, (char)zeichen1, (char)zeichen2);
                }
            } while (zeichen1 == zeichen2 && zeichen1 != -1);

            if (zeichen1 == zeichen2)
            {
                System.out.printf("Vergleich in Ordnung!");
            }
            else
                if (zeichen1 == -1 || zeichen2 == -1)
                {
                    System.out.printf("Die Dateien sind unterschiedlich lang!");
                }
                else
                {
                    System.out.printf("Unterschied bei %d. Zeichen!", position);
                }
        }
        catch (Exception oe)
        {
            System.out.println("Fehler beim Öffnen: " + oe.getMessage());
        }
    }
}
```

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:

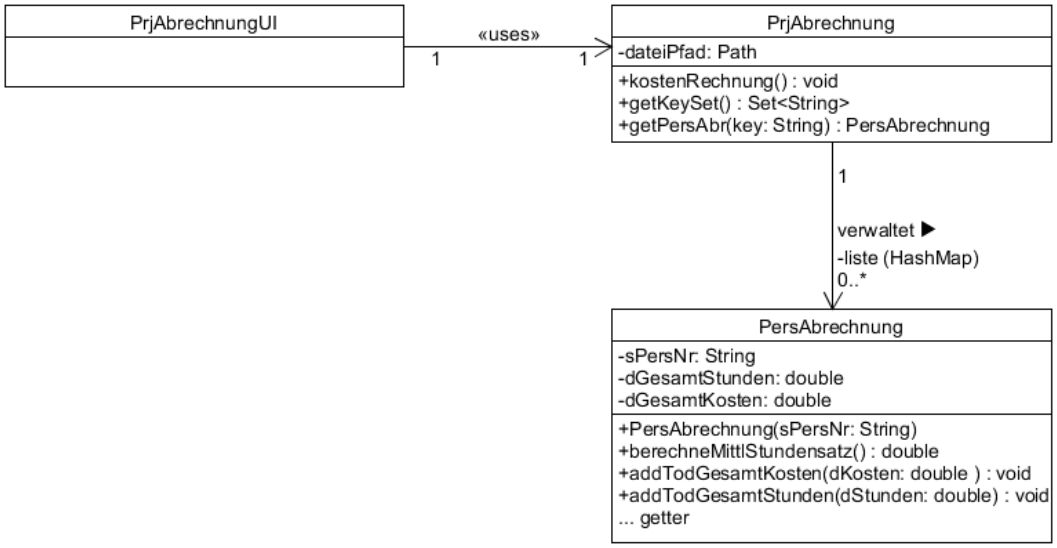
- Der Pfadstring für die Datei wird mit einem parametrierten Konstruktor festgelegt.
- Die Klasse besitzt eine Objektmethode `public void kostenrechnung(String sPersonalNr)` , die zu einer `sPersonalNr` die gesamte Arbeitszeit, die gesamten Arbeitskosten und den Mittelwert des Stundensatzes ermittelt 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.

```
public class PrjAbrechnung
{
    private Path dateiPfad;
    private String sPersNr;
    private double dGesamtKosten;
    private double dGesamtStunden;

    public PrjAbrechnung(String sDateiPfad)
    {
        this.dateiPfad = Paths.get(sDateiPfad);
        this.dGesamtKosten = 0;
        this.dGesamtStunden = 0;
    }
    public double getdGesamtKosten()
    public double getdGesamtStunden()
    public double berechneMittlStundensatz()
    {
        double dMittel = 0;

        if (this.dGesamtStunden != 0)
        {
            dMittel = this.dGesamtKosten / this.dGesamtStunden;
        }
        return dMittel;
    }
```

```
    public void kostenRechnung(String sPersNr) throws IOException
    {
        String eineZeile;
        String[] zeilenTeile;
        double dStunden;

        this.sPersNr = sPersNr;

        try (BufferedReader reader = Files.newBufferedReader(this.dateiPfad,
                                                             StandardCharsets.UTF_8))
        {
            eineZeile = reader.readLine();  // liest die nächste Zeile
            while (eineZeile != null)
            {   // Die folgende Anweisung trennt die Bestandteile der Zeile
                // und speichert die Teile in einen String Array
                zeilenTeile = eineZeile.split(";");

                if (zeilenTeile[0].equals(this.sPersNr))
                {
                    dStunden = Double.parseDouble(zeilenTeile[2]);
                    this.dGesamtStunden = this.dGesamtStunden + dStunden;
                    this.dGesamtKosten = this.dGesamtKosten + dStunden
                                       * Double.parseDouble(zeilenTeile[3]);
                }

                eineZeile = reader.readLine();  // liest die nächste Zeile
            }
        }
        catch (IOException oe)
        {
            throw oe;  // Exception zum Aufrufer weitergeben
        }
        catch (Exception oe)
        {
            throw oe;  // Exception zum Aufrufer weitergeben
        }
    }
}
```

```
public class PrjAbrechnungUI
{
    /**
     * @param args
     */
    public static void main(String[] args)
    {
        double dGesamtStunden;

        PrjAbrechnung abrMitarb = new PrjAbrechnung("Dateien/kosten.txt");

        try
        {
            abrMitarb.kostenRechnung("127");

            dGesamtStunden = abrMitarb.getdGesamtStunden();

            if (dGesamtStunden != 0)
            {
                System.out.printf("Mitarbeiter ID      = %8s\n", "127");
                System.out.printf("Gesamtarbeitszeit   = %8.2f Stunden\n",
                                  dGesamtStunden);
                System.out.printf("Gesamtarbeitskosten = %8.2f Euro\n",
                                  abrMitarb.getdGesamtKosten());
                System.out.printf("Durchschnittslohn   = %8.2f Euro/Stunde\n",
                                  abrMitarb.berechneMittlStundensatz());
            }
        }
        catch (Exception oe)
        {
            System.out.println("Fehler: " + oe.getMessage());
        }
    }
}
```

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).

```java
public class PrjAbrechnungUI_v5
{
    /**
     * @param args
     */
    public static void main(String[] args)
    {
        double dGesamtStunden;
        PersAbrechnung temp;

        PrjAbrechnung_v5 abrechnung = new PrjAbrechnung_v5("Dateien/kosten.txt");

        try
        {
            abrechnung.kostenRechnung();

            Set<String> keys = abrechnung.getKeySet();

            for (String key : keys)
            {
                temp = abrechnung.getPersAbr(key);
                dGesamtStunden = temp.getdGesamtStunden();
                if (dGesamtStunden != 0)
                {
                    System.out.printf("Mitarbeiter ID      = %8s\n", temp.getsPersNr());
                    System.out.printf("Gesamtarbeitszeit   = %8.2f Stunden\n",
                                                            dGesamtStunden);
                    System.out.printf("Gesamtarbeitskosten = %8.2f Euro\n",
                                                              temp.getdGesamtKosten());
                    System.out.printf("Durchschnittslohn   = %8.2f Euro/Stunde\n",
                                                         temp.berechneMittlStundensatz());
                }

            }

        }
        catch (Exception oe)
        {
            System.out.println("Fehler: " + oe.getMessage());
        }
    }
}
```

```java
public class PersAbrechnung implements Comparable<PersAbrechnung>
{
    private String sPersNr;
    private double dGesamtKosten;
    private double dGesamtStunden;

    public PersAbrechnung(String sPersNr)
    {
        this.sPersNr = sPersNr;
        this.dGesamtKosten = 0;
        this.dGesamtStunden = 0;
    }

    public double getdGesamtKosten()

    public double getdGesamtStunden()

    public double berechneMittlStundensatz()
    {
        double dMittel = 0;

        if (this.dGesamtStunden != 0)
        {
            dMittel = this.dGesamtKosten / this.dGesamtStunden;
        }
        return dMittel;
    }

    @Override
    public int compareTo(PersAbrechnung anderePers)
    {
       return this.sPersNr.compareTo(anderePers.sPersNr);

    }

    public void addTodGesamtKosten(double dGesamtKosten)
    {
        this.dGesamtKosten = this.dGesamtKosten + dGesamtKosten;
    }

    public void addTodGesamtStunden(double dGesamtStunden)
    {
        this.dGesamtStunden = this.dGesamtStunden + dGesamtStunden;
    }

    public String getsPersNr()
}
```

```java
public class PrjAbrechnung_v5
{
    private Path dateiPfad;
    private HashMap<String, PersAbrechnung> liste = new HashMap<>();

    public PrjAbrechnung_v5(String sDateiPfad)
    {
        this.dateiPfad = Paths.get(sDateiPfad);
    }

    public void kostenRechnung() throws IOException
    {
        String eineZeile;
        String[] zeilenTeile;
        PersAbrechnung temp = null;
        PersAbrechnung gesucht;
        double dStunden;
        int index;

        try (BufferedReader reader = Files.newBufferedReader(this.dateiPfad, StandardCharsets.UTF_8))
        {
            eineZeile = reader.readLine();  // liest die nächste Zeile
            while (eineZeile != null)
            {
                zeilenTeile = eineZeile.split(";");
                gesucht = liste.get(zeilenTeile[0]);

                if (gesucht == null)
                {
                    temp = new PersAbrechnung(zeilenTeile[0]);
                    liste.put(zeilenTeile[0], temp);
                }
                else
                {
                    temp = gesucht;
                }

                dStunden = Double.parseDouble(zeilenTeile[2]);
                temp.addTodGesamtStunden(dStunden);
                temp.addTodGesamtKosten(dStunden * Double.parseDouble(zeilenTeile[3]));
                eineZeile = reader.readLine();  // liest die nächste Zeile
            }
        }
        catch (IOException oe)
        {
            throw oe;
        }
        catch (Exception oe)
        {
            throw oe;
        }
    }
    public int getSize()
    public PersAbrechnung getPersAbr(String key)
    {
        return liste.get(key);
    }

    public Set<String> getKeySet()
    {
        Set<String> it = liste.keySet();
        return it;
    }
}
```

1.6. Es soll eine Klasse erstellt werden, die aus einer Eingabedatei Nachname und Vorname