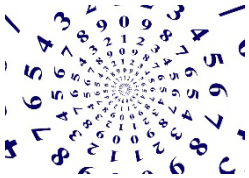
Fach: AEuP

Name: _____ Datum: _____

BSZ WIESAU Informatik Campus

Aufgabe

Zeilennummern



Lesen Sie die Java-Datei Beispiel.java des K-Laufwerks zeilenweise aus und schreiben Sie den Inhalt einer jeden Zeile inklusive führender Zeilennummer als Kommentar in eine Datei BeispielMitNummern.java. Diese Java-Datei soll nach wie vor ausführbar sein.

Für Schnelle:

Der Name, der zu lesenden Datei, soll bei Programmstart über die Konsole abgefragt werden. Verwenden Sie dazu den BufferedReader. Sollte die eingegebene Datei nicht existieren, so soll mit Hilfe der Klasse BufferedWriter diese Information in die Konsole gedruckt werden.

Kanzleikundenverwaltung

Für eine Anwaltskanzlei soll eine Kundenverwaltung geschrieben werden. Die zu verwendenden Klassen und Methoden sind Ihnen vorgegeben:

Klasse ***Program***:

static ArrayList<Kunde> kunden

public static void kundenauflisten()

public static void kundenladen(String datei)

public static void kundenspeichern(String datei)

Klasse ***Kunde***:

private int knr

private String kname

private LocalDate kgebdat

Die Kunden werden in der Datei „*kunden.txt*“ gespeichert. Beim Programmstart werden daraus die Kunden durch die Methode Kundenladen in die Liste „*kunden*“ eingelesen und dann im Programm verwendet. Bevor das Programm beendet wird, wird mit der Methode „*kundenspeichern*“ der Inhalt der Liste in die Datei gespeichert. Die Liste enthält nur „Kunde“-Objekte!

Der Aufbau der Textdatei „*kunden.txt*“ soll wie folgt sein:

- Pro Zeile wird ein Kunde gespeichert.
- Die einzelnen Felder sind durch ein „#“ getrennt.
- Folgende Felder sind in dieser Reihenfolge in jeder Zeile vorhanden: „laufende Nummer“, „Kundennummer“, „Kundenname“, „Kundengeburtsdatum“

Eine Datei nach diesen Angaben könnte als Beispiel so aussehen:

1#1#Huber Philip#1955-12-31

2#13#Maier Bernd#1999-01-01

3#11#Schmid Josef#1970-05-22

Nach Programmstart soll folgendes Menü erscheinen:

```
1. Kunde eingeben
2. Kunde löschen
3. Kunde bearbeiten
4. Kundenliste ausgeben
0. Programm beenden
```

Aufgaben:

- Kodieren Sie das Programm mit den vorgegebenen Klassen und Methoden.
- Für die Klasse Kunde ist noch kein Konstruktor definiert. Definieren und kodieren Sie einen Konstruktor für diese Klasse, mit dem man die Attribute „kundennummer“, „kundenname“ und „kundengeburtsdatum“ setzen kann. Fügen Sie weiterhin Set-& Getmethoden hinzu.
- Die Methode „kundenspeichern“ soll alle Einträge der Liste „kunden“ in die Datei „kunden.txt“ im vorher angegebenen Format speichern und zwar immer, wenn der Benutzer den Punkt Programm beenden wählt.
- **Ihr Programm soll völlig absturzsicher werden!** Überprüfen Sie dazu alle Methoden auf möglicherweise auftretende Laufzeitfehler und fangen Sie diese durch Verwendung von **try-catch** ab. Beachten Sie vor allem die Möglichkeit einer nicht vorhandenen oder schreibgeschützen „kunden.txt“.

Für Schnelle:

Die Kunden sollen sortiert ausgegeben werden können und zwar nach der Kundennummer, dem Kundennamen oder dem Kundengeburtsdatum. Dazu soll die Methode „kundenauflisten“ den Benutzer fragen, wonach er sortieren lassen möchte.

Bei der Eingabe des Geburtsdatums soll überprüft werden, ob es sich um ein „korrektes“ Datum handelt. Richtiges Format bzw. in der Vergangenheit liegend. Sollte ein falsches Datum eingebeben werden, so sollen sprechende Fehlermeldungen erfolgen sowie eine Wiederholung der Eingabe.

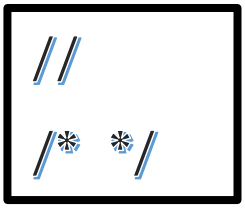
Fach: AEuP

Name: _____ Datum: _____

BSZ WIESAU Informatik Campus

Kommentare entfernen

Lesen Sie die Java-Datei *DateiMitKommentare.java* des K-Laufwerks und schreiben Sie deren Inhalt in eine neue Datei – ABER OHNE sämtliche Kommentare.



Kommentare innerhalb von Strings sollen dabei natürlich nicht entfernt werden!

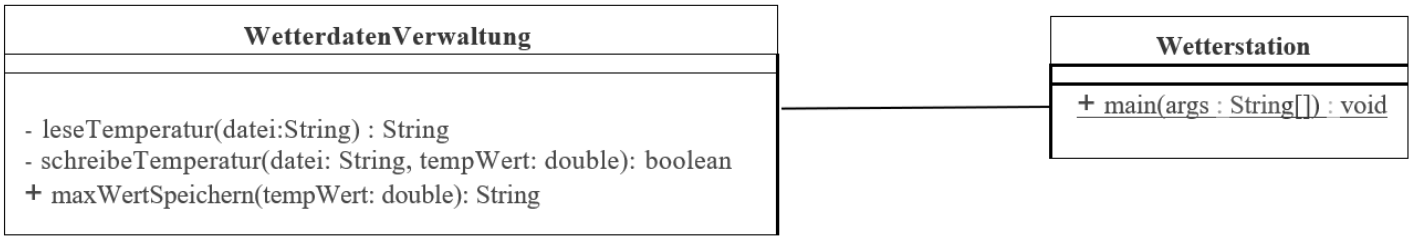
Tipp: Lesen Sie zeichenweise! (Alternativ: String entspricht einem char-Array!)

Erstellen Sie zunächst ein Struktogramm!

Aufgabe Wetterstation

Eine Wetterstation möchte die täglich aufgenommenen Wetterdaten programmiertechnisch erfassen und verwalten.

In der ersten Testphase, in der die eingelesene Höchsttemperatur gespeichert werden soll, ist folgendes Programm geplant:



Informationen zu den Klassen:

public class WetterdatenVerwaltung

`- leseTemperatur(datei: String) : String`

Methode für den lesenden Zugriff auf eine Datei.
Die Methode greift auf die durch den Übergabeparameter festgelegte Datei zu und liest den darin vorhandenen Wert aus. Dieser Wert wird an die aufrufende Stelle zurückgegeben.
Parameter:

- Dateiname (mit Dateierweiterung) als String

Rückgabewerte:

- Temperaturwert (aus der angegebenen Datei) als String
- "*Fehler beim Dateizugriff! – Lesefehler*!", wenn Lesezugriff IOException verursacht
- "*Fehler beim Dateizugriff! – angegebene Datei nicht vorhanden*!", wenn Datei nicht gefunden wurde (FileNotFoundException)

`- schreibeTemperatur(datei:String, tempWert:double) : boolean`

Methode für den schreibenden Zugriff auf eine Datei.
Die Methode greift auf die durch den Übergabeparameter festgelegte Datei zu und schreibt den als zweiten Parameter übergebenen Wert in die Datei. Ein bestehender alter Wert wird dabei überschrieben. Die Methode liefert einen boolean-Wert als Bestätigung zurück.
Parameter:

- Dateiname (mit Dateierweiterung) als String
- Temperaturwert als double

Rückgabewerte:

- *true,* wenn Speichervorgang erfolgreich abgeschlossen wurde
- *false,* bei IOException

`+ maxWertSpeichern(tempWert:double) : String`

Methode zum Abspeichern des maximalen Temperaturwertes. Der übergebene Temperaturwert wird mit dem in der Datei "**maxtemp.txt**" gespeicherten Wert verglichen. Ist der neue Temperaturwert höher, soll dieser in der Datei gespeichert werden. Zum Lesen des Wertes aus der Datei und zum Abspeichern des neuen Wertes werden die Methoden leseTemperatur(...) und schreibeTemperatur(...) der Klasse verwendet. Der aktuell gespeicherte Maximalwert (bzw. ein Fehlerwert) wird zurückgegeben.
Parameter:

- Temperaturwert als double

Rückgabewerte:

- aktuell gespeicherter Temperaturwert (maximale Temperatur) als String

- *"Fehler beim Dateizugriff! – Lesefehler!"*, wenn Lesezugriff IOException verursacht
- *"Fehler beim Dateizugriff! – angegebene Datei nicht vorhanden!"*, wenn Datei nicht gefunden wurde (FileNotFoundException)
- *"Fehler beim Dateizugriff – Schreibfehler!"*, wenn Schreibzugriff IOException verursacht

public class Wetterstation

Ausführbare Klasse zum Starten des Wetterdatenprogramms.
Innerhalb der main-Methode soll ein neuer Temperaturwert mittels Scanner eingelesen werden. Anschließend soll der neue Wert durch Aufruf der Methode maxWert-Speichern(...) mit dem bereits in der Datei "**maxtemp.txt**" gespeicherten Wert verglichen werden.

Durch entsprechende Bildschirmanzeigen sollen dem Benutzer Informationen über die Verarbeitung gegeben werden (aktuelle gespeicherte Maximaltemperatur, Fehlermeldung).

Beispiele für die Bildschirmanzeige:

- Fehlerlose Verarbeitung

```
Temperaturwert eingeben
12.5
---------------------------------------------------------
maximale Temperatur in °C:
25
```

- Fehler beim Lesen der Datei

```
Temperaturwert eingeben
12.5
---------------------------------------------------------
maximale Temperatur in °C:
Fehler beim Dateizugriff! - Lesefehler
```

- Fehler beim Schreiben der Datei

```
Temperaturwert eingeben
12.5
---------------------------------------------------------
maximale Temperatur in °C:
Fehler beim Dateizugriff! - Schreibfehler
```

Arbeitsauftrag

Erstellen Sie entsprechend der gegebenen Beschreibungen das Programm für die Wetterdatenverwaltung in zwei Schritten.

1. Schreiben Sie die Klasse WetterdatenVerwaltung mit den Methoden leseTemperatur() und schreibeTemperatur() und setzen Sie diese zunächst auf public. Testen Sie die beiden Methoden mithilfe einer eigenen ausführbaren Testdatei.
2. Verändern bzw. erweitern Sie die Klasse WetterdatenVerwaltung so, dass sie den gegebenen Beschreibungen entspricht. Legen Sie die Klasse Wetterstation an. Testen Sie ihr Programm.