

Tutorium Programmieren

Tut Nr.8: FileReader, String API Michael Friedrich | 17. / 19.12.2013

INSTITUT FÜR THEORETISCHE INFORMATIK



Outline/Gliederung



- Blatt3
- Blatt4
- 3 FileReader
- 4 String API

Michael Friedrich - Prog Tut Nr. 8

Quiz





- nur geforderte Dateien hochladen!
- achtet auf die Aufgabenstellung!

Blatt4

- Modellierung! ⇒ Methoden, nicht alles in die main
- Ausgabe! ⇒ Leerzeichen k\u00f6nnen euch in der Abschlussaufgabe teuer kosten
- Aufgabe lesen! zB args ⇒ bei Unklarheit fragen!



String API



- nur geforderte Dateien hochladen!
- achtet auf die Aufgabenstellung!

Blatt4

- Modellierung! ⇒ Methoden, nicht alles in die main
- Ausgabe! ⇒ Leerzeichen k\u00f6nnen euch in der Abschlussaufgabe teuer kosten
- Aufgabe lesen! zB args ⇒ bei Unklarheit fragen!



String API



- nur geforderte Dateien hochladen!
- achtet auf die Aufgabenstellung!

Blatt4

- Modellierung! ⇒ Methoden, nicht alles in die main
- Ausgabe! ⇒ Leerzeichen k\u00f6nnen euch in der Abschlussaufgabe teuer kosten
- Aufgabe lesen! zB args ⇒ bei Unklarheit fragen!



17. / 19.12.2013



- nur geforderte Dateien hochladen!
- achtet auf die Aufgabenstellung!

Blatt4

- Modellierung! ⇒ Methoden, nicht alles in die main
 - Ausgabe! ⇒ Leerzeichen k\u00f6nnen euch in der Abschlussaufgabe teuer kosten
- Aufgabe lesen! zB args ⇒ bei Unklarheit fragen!





- nur geforderte Dateien hochladen!
- achtet auf die Aufgabenstellung!
 - Modellierung! ⇒ Methoden, nicht alles in die main
 - Ausgabe! ⇒ Leerzeichen k\u00f6nnen euch in der Abschlussaufgabe teuer kosten
 - Autgabe lesen! zB args ⇒ bei Unklarheit fragen!



Blatt3

String API



- nur geforderte Dateien hochladen!
- achtet auf die Aufgabenstellung!
 - Modellierung! ⇒ Methoden, nicht alles in die main
 - Ausgabe! ⇒ Leerzeichen k\u00f6nnen euch in der Abschlussaufgabe teuer kosten
 - Aufgabe lesen! zB args ⇒ bei Unklarheit fragen!



Michael Friedrich - Prog Tut Nr. 8

Blatt3



- Fehler meinerseits...Kein Abzug, nur Anmerkung in der Korrektur.
 - Exceptions unbedingt zu vermeiden!

Lücke füllen und damit implizit löschen

```
for(int j = indexFound; j < elements.length; j++) {
elements[j] = elements[j+1];
}
lastElem--;
}</pre>
```

Führt zu einer Exception! Ähnlich auch in Aufgabenteil C

- \Rightarrow Entweder: Nur bis zu (elements.length 1) gehen und letztes Element außerhalb der for Schleife manipulieren
- ⇒ ODER : Indizierung anders wählen: elements[j-1] = elements[j] im for-Rumpf

4 L	 ,	٠.	Jr F	 -	4	=	r	=	بار	, (>	,



4/9

- Fehler meinerseits...Kein Abzug, nur Anmerkung in der Korrektur.
 Aber in Zukunft Fehler!
 - Exceptions unbedingt zu vermeiden!

Lücke füllen und damit implizit löschen

```
for(int j = indexFound; j < elements.length; j++) {
elements[j] = elements[j+1];
}
lastElem--;
}</pre>
```

Führt zu einer Exception! Ähnlich auch in Aufgabenteil C

- \Rightarrow Entweder: Nur bis zu (elements.length 1) gehen und letztes Element außerhalb der for Schleife manipulieren
- ⇒ ODER : Indizierung anders wählen: elements[j-1] = elements[j] im for Pumpf

			1 U 7 1 D 7 1 E 7 1 E 7	= +)40
Blatt3	Blatt4	FileReader	String API	Quiz



- Fehler meinerseits...Kein Abzug, nur Anmerkung in der Korrektur.
 Aber in Zukunft Fehler!
 - Exceptions unbedingt zu vermeiden!

Lücke füllen und damit implizit löschen

```
for(int j = indexFound; j < elements.length; j++) {
elements[j] = elements[j+1];
}
lastElem--;
}</pre>
```

Führt zu einer Exception! Ähnlich auch in Aufgabenteil C

- ⇒ Entweder: Nur bis zu (elements.length 1) gehen und letztes Element außerhalb der for Schleife manipulieren
- ⇒ ODER : Indizierung anders wählen: elements[j-1] = elements[j] im

Blatt3 Blatt4 FileReader String API Quiz



- Fehler meinerseits...Kein Abzug, nur Anmerkung in der Korrektur.
 Aber in Zukunft Fehler!
 - Exceptions unbedingt zu vermeiden!

Lücke füllen und damit implizit löschen

```
for(int j = indexFound; j < elements.length; j++) {
elements[j] = elements[j+1];
}
lastElem--;
}</pre>
```

Führt zu einer Exception! Ähnlich auch in Aufgabenteil C

- ⇒ Entweder: Nur bis zu (elements.length 1) gehen und letztes Element außerhalb der for Schleife manipulieren
- ⇒ ODER : Indizierung anders wählen: elements[j-1] = elements[j] im

Blatt3 Blatt4 FileReader String API Quiz



- Fehler meinerseits...Kein Abzug, nur Anmerkung in der Korrektur.
 Aber in Zukunft Fehler!
 - Exceptions unbedingt zu vermeiden!

Lücke füllen und damit implizit löschen

```
for(int j = indexFound; j < elements.length; j++) {
elements[j] = elements[j+1];
}
lastElem--;
}</pre>
```

Führt zu einer Exception! Ähnlich auch in Aufgabenteil C

- ⇒ Entweder: Nur bis zu (elements.length 1) gehen und letztes Element außerhalb der for Schleife manipulieren
- ⇒ ODER : Indizierung anders wählen: elements[j-1] = elements[j] im for-Rumpf

Blatt4

Blatt3

17. / 19.12.2013



- Fehler meinerseits...Kein Abzug, nur Anmerkung in der Korrektur.
 Aber in Zukunft Fehler!
 - Exceptions unbedingt zu vermeiden!

Lücke füllen und damit implizit löschen

```
for(int j = indexFound; j < elements.length; j++) {
elements[j] = elements[j+1];
}
lastElem--;
}</pre>
```

Führt zu einer Exception! Ähnlich auch in Aufgabenteil C

- ⇒ Entweder: Nur bis zu (elements.length 1) gehen und letztes Element außerhalb der for Schleife manipulieren
- \Rightarrow ODER : Indizierung anders wählen: elements[j-1] = elements[j] im for-Rumpf

Blatt3 Blatt4 FileReader String API Quiz



- Kapselung! Jetz Pflicht...Wiederholungsbedarf?
- "Faustregeln"
 - Attribute private
 - Methoden meistens public (außer Hilfsmethoden)
 - Konstruktor IMMER public
- Vergesst nicht, Checkstyle upzudaten
- Listen erstellen wie letztes Tut

Blatt4



String API



- Kapselung! Jetz Pflicht...Wiederholungsbedarf?
- "Faustregeln"
 - Attribute private
 - Methoden meistens public (außer Hilfsmethoden)
 - Konstruktor IMMER public
- Vergesst nicht, Checkstyle upzudaten
- Listen erstellen wie letztes Tut

Blatt4



Blatt3



- Kapselung! Jetz Pflicht...Wiederholungsbedarf?
- "Faustregeln"
 - Attribute private
 - Methoden meistens public (außer Hilfsmethoden)
 - Konstruktor IMMER public
- Vergesst nicht, Checkstyle upzudaten
- Listen erstellen wie letztes Tut

Blatt4



Blatt3



- Kapselung! Jetz Pflicht...Wiederholungsbedarf?
- "Faustregeln"
 - Attribute private
 - Methoden meistens public (außer Hilfsmethoden)
 - Konstruktor IMMER public
- Vergesst nicht, Checkstyle upzudaten
- Listen erstellen wie letztes Tut

Blatt4



Blatt3



- Kapselung! Jetz Pflicht...Wiederholungsbedarf?
- "Faustregeln"
 - Attribute private
 - Methoden meistens public (außer Hilfsmethoden)
 - Konstruktor IMMER public
- Vergesst nicht, Checkstyle upzudaten
- Listen erstellen wie letztes Tut



17. / 19.12.2013



- Kapselung! Jetz Pflicht...Wiederholungsbedarf?
- "Faustregeln"
 - Attribute private
 - Methoden meistens public (außer Hilfsmethoden)
 - Konstruktor IMMER public
- Vergesst nicht, Checkstyle upzudaten
- Listen erstellen wie letztes Tut

Blatt4



17. / 19.12.2013



- Kapselung! Jetz Pflicht...Wiederholungsbedarf?
- "Faustregeln"
 - Attribute private
 - Methoden meistens public (außer Hilfsmethoden)
 - Konstruktor IMMER public
- Vergesst nicht, Checkstyle upzudaten
- Listen erstellen wie letztes Tut

Blatt4



17. / 19.12.2013



- Kapselung! Jetz Pflicht...Wiederholungsbedarf?
- "Faustregeln"
 - Attribute private
 - Methoden meistens public (außer Hilfsmethoden)
 - Konstruktor IMMER public
- Vergesst nicht, Checkstyle upzudaten
- Listen erstellen wie letztes Tut

Blatt4



Blatt3



- Kapselung! Jetz Pflicht...Wiederholungsbedarf?
- "Faustregeln"
 - Attribute private
 - Methoden meistens public (außer Hilfsmethoden)
 - Konstruktor IMMER public
- Vergesst nicht, Checkstyle upzudaten
- Listen erstellen wie letztes Tut



17. / 19.12.2013



```
public static void main(String[] args) {
if (args.length != 1) {
System.out.println(USAGE):
System.exit(1);
FileReader in = null:
try {
in = new FileReader(args[0]);
} catch (FileNotFoundException e) {
System.out.println(ERROR_MESSAGE);
System.exit(1);
RufferedReader reader = new RufferedReader
      (in):
trv {
String line = reader.readLine():
while (line != null) {
// TODO: process line here
line = reader.readLine():
} catch (IOException e) {
System.out.println(ERROR_MESSAGE);
System.exit(1):
```

FileReader öffnet die Datei

- BufferedReader nimmt den Datenstrom aus der Datei entgegen
 → Datei lässt sich Zeilenweise auslesen
- Fehlerbehandlung muss hier geschehen, lernen wir aber erst nächstes Jahr
 → Ihr könnt das nickend annehmen und benutzen
 ⇒ TODO euer
 Angriffspunkt



Blatt3

FileReader



```
public static void main(String[] args) {
if (args.length != 1) {
System.out.println(USAGE):
System.exit(1);
FileReader in = null:
try {
in = new FileReader(args[0]);
} catch (FileNotFoundException e) {
System.out.println(ERROR_MESSAGE);
System.exit(1);
RufferedReader reader = new RufferedReader
      (in):
trv {
String line = reader.readLine():
while (line != null) {
// TODO: process line here
line = reader.readLine():
} catch (IOException e) {
System.out.println(ERROR_MESSAGE);
System.exit(1):
```

- FileReader öffnet die Datei
- BufferedReader nimmt den Datenstrom aus der Datei entgegen
 - → Datei lässt sich Zeilenweise auslesen
- Fehlerbehandlung muss hier geschehen, lernen wir aber erst nächstes Jahr
 → Ihr könnt das nickend annehmen und benutzen
 ⇒ TODO euer
 Angriffspunkt



Blatt3 Blatt4 FileReader String API Quiz



```
public static void main(String[] args) {
if (args.length != 1) {
System.out.println(USAGE):
System.exit(1);
FileReader in = null:
try {
in = new FileReader(args[0]);
} catch (FileNotFoundException e) {
System.out.println(ERROR_MESSAGE);
System.exit(1);
RufferedReader reader = new RufferedReader
      (in):
trv {
String line = reader.readLine():
while (line != null) {
// TODO: process line here
line = reader.readLine():
} catch (IOException e) {
System.out.println(ERROR_MESSAGE);
System.exit(1):
```

- FileReader öffnet die Datei
- BufferedReader nimmt den Datenstrom aus der Datei entgegen
 → Datei lässt sich
 Zeilenweise auslesen
- Fehlerbehandlung muss hier geschehen, lernen wir aber erst nächstes Jahr
 → Ihr könnt das nickend annehmen und benutzen
 ⇒ TODO euer
 Angriffspunkt

String API



FileReader



```
public static void main(String[] args) {
if (args.length != 1) {
System.out.println(USAGE):
System.exit(1);
FileReader in = null:
try {
in = new FileReader(args[0]);
} catch (FileNotFoundException e) {
System.out.println(ERROR_MESSAGE);
System.exit(1);
RufferedReader reader = new RufferedReader
      (in):
trv {
String line = reader.readLine():
while (line != null) {
// TODO: process line here
line = reader.readLine():
 catch (IOException e) {
System.out.println(ERROR_MESSAGE);
System.exit(1):
```

- FileReader öffnet die Datei
- BufferedReader nimmt den Datenstrom aus der Datei entgegen
 → Datei lässt sich Zeilenweise auslesen
- Fehlerbehandlung muss hier geschehen, lernen wir aber erst nächstes Jahr
 - → Ihr könnt das nickend annehmen und benutzen
 ⇒ TODO euer
 Angriffspunkt



Blatt3 Blatt4 FileReader String API



```
public static void main(String[] args) {
if (args.length != 1) {
System.out.println(USAGE):
System.exit(1);
FileReader in = null:
try {
in = new FileReader(args[0]);
} catch (FileNotFoundException e) {
System.out.println(ERROR_MESSAGE);
System.exit(1);
RufferedReader reader = new RufferedReader
      (in):
trv {
String line = reader.readLine():
while (line != null) {
// TODO: process line here
line = reader.readLine():
 catch (IOException e) {
System.out.println(ERROR_MESSAGE);
System.exit(1):
```

- FileReader öffnet die Datei
- BufferedReader nimmt den Datenstrom aus der Datei entgegen → Datei lässt sich Zeilenweise auslesen
- Fehlerbehandlung muss hier geschehen, lernen wir aber erst nächstes Jahr → Ihr könnt das nickend annehmen und benutzen

FileReader



Michael Friedrich - Prog Tut Nr. 8

Blatt3



```
public static void main(String[] args) {
if (args.length != 1) {
System.out.println(USAGE):
System.exit(1);
FileReader in = null:
try {
in = new FileReader(args[0]);
} catch (FileNotFoundException e) {
System.out.println(ERROR_MESSAGE);
System.exit(1);
RufferedReader reader = new BufferedReader
      (in):
trv {
String line = reader.readLine():
while (line != null) {
// TODO: process line here
line = reader.readLine():
 catch (IOException e) {
System.out.println(ERROR_MESSAGE);
System.exit(1):
```

- FileReader öffnet die Datei
- BufferedReader nimmt den Datenstrom aus der Datei entgegen
 → Datei lässt sich Zeilenweise auslesen
- Fehlerbehandlung muss hier geschehen, lernen wir aber erst nächstes Jahr
 → Ihr könnt das nickend annehmen und benutzen
 ⇒ TODO euer Angriffspunkt

String API



FileReader

Blatt3



7/9

Benötigte Theorie: REGular EXpressions (siehe GBI) hier erstmal Trennzeichen, wie z.Bsp. . (Punkt)

String[] split(String regex) Trennt den String am Trennzeichen und liefert ein Array zurück

Ausdruck entspricht

- String toLowerCase() Macht jedes Zeichen eines String klein
 - analog: String toUpperCase()





Benötigte Theorie: **REGular Expressions** (siehe GBI) hier erstmal Trennzeichen, wie z.Bsp. . (Punkt)

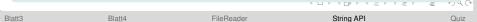
Werkzeugkasten, um Strings zu manipulieren

String[] split(String regex) Trennt den String am Trennzeichen und liefert ein Array zurück

```
"boo:and:foo".split(":"); // Ergebnis: {" bo o ","and "," f o o "}
"boo:and:foo".split("o"); // Ergebnis: {" b ",""," : and : f "}
```

```
"A1".matches([A-Z][0-9]) //true
"a8".matches([A-Z][0-9]) //false
```

- String toLowerCase() Macht jedes Zeichen eines String klein
 - analog: String toUpperCase()





7/9

Benötigte Theorie: **REGular Expressions** (siehe GBI) hier erstmal Trennzeichen, wie z.Bsp. . (Punkt)

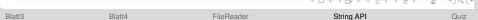
Werkzeugkasten, um Strings zu manipulieren

String[] split(String regex) Trennt den String am Trennzeichen und liefert ein Array zurück

```
"boo:and:foo".split(":"); // Ergebnis: {" bo o ","and "," f o o "}
"boo:and:foo".split("o"); // Ergebnis: {" b ",""," : and : f "}
```

```
"A1".matches([A-Z][0-9]) //true
"a8".matches([A-Z][0-9]) //false
```

- String toLowerCase() Macht jedes Zeichen eines String klein
 - analog: String toUpperCase()





7/9

Benötigte Theorie: **REGular Expressions** (siehe GBI) hier erstmal Trennzeichen, wie z.Bsp. . (Punkt)

Werkzeugkasten, um Strings zu manipulieren

String[] split(String regex) Trennt den String am Trennzeichen und liefert ein Array zurück

```
"boo:and:foo".split(":"); // Ergebnis: {" bo o ","and "," f o o "}
"boo:and:foo".split("o"); // Ergebnis: {" b ",""," : and : f "}
```

```
"A1".matches([A-Z][0-9]) //true
"a8".matches([A-Z][0-9]) //false
```

- String toLowerCase() Macht jedes Zeichen eines String klein
 - analog: String toUpperCase()





7/9

Benötigte Theorie: **REGular Expressions** (siehe GBI) hier erstmal Trennzeichen, wie z.Bsp. . (Punkt)

Werkzeugkasten, um Strings zu manipulieren

 String[] split(String regex) Trennt den String am Trennzeichen und liefert ein Array zurück

```
"boo:and:foo".split(":"); // Ergebnis: {" bo o ","and "," f o o "}
"boo:and:foo".split("o"); // Ergebnis: {" b ",""," : and : f "}
```

```
"A1".matches([A-Z][0-9]) //true
"a8".matches([A-Z][0-9]) //false
```

- String toLowerCase() Macht jedes Zeichen eines String klein
 - analog: String toUpperCase()





Benötigte Theorie: REGular EXpressions (siehe GBI) hier erstmal Trennzeichen, wie z.Bsp. . (Punkt)

Werkzeugkasten, um Strings zu manipulieren

String[] split(String regex) Trennt den String am Trennzeichen und liefert ein Array zurück

```
"boo:and:foo".split(":"); // Ergebnis: {" bo o ", and ", f o o "}
  "boo:and:foo".split("o"); // Ergebnis: {" b ",""," : and : f "}
```

```
"A1".matches([A-Z][0-9]) //true
"a8".matches([A-Z][0-9]) //false
```





Benötigte Theorie: **REGular EXpressions** (siehe GBI) hier erstmal Trennzeichen, wie z.Bsp. . (Punkt)

Werkzeugkasten, um Strings zu manipulieren

 String[] split(String regex) Trennt den String am Trennzeichen und liefert ein Array zurück

```
"boo:and:foo".split(":"); // Ergebnis: {" bo o ","and "," fooo"}
"boo:and:foo".split("o"); // Ergebnis: {" b ",""," : and : f "}
```

 boolean matches(String regex) Prüft, ob ein String dem regulären Ausdruck entspricht

```
"A1".matches([A-Z][0-9]) //frue  
"a8".matches([A-Z][0-9]) //false
```

- String toLowerCase() Macht jedes Zeichen eines String klein
 - analog: String toUpperCase()

Blatt4





7/9

Benötigte Theorie: **REGular EXpressions** (siehe GBI) hier erstmal Trennzeichen, wie z.Bsp. . (Punkt)

Werkzeugkasten, um Strings zu manipulieren

String[] split(String regex) Trennt den String am Trennzeichen und liefert ein Array zurück

```
"boo:and:foo".split(":"); // Ergebnis: {" bo o ","and "," f o o "}
"boo:and:foo".split("o"); // Ergebnis: {" b ",""," : and : f "}
```

```
"A1".matches([A-Z][0-9]) // true  
"a8".matches([A-Z][0-9]) // false
```

- String toLowerCase() Macht jedes Zeichen eines String klein
 - analog: String toUpperCase()





7/9

Benötigte Theorie: **REGular EXpressions** (siehe GBI) hier erstmal Trennzeichen, wie z.Bsp. . (Punkt)

Werkzeugkasten, um Strings zu manipulieren

String[] split(String regex) Trennt den String am Trennzeichen und liefert ein Array zurück

```
"boo:and:foo".split(":"); // Ergebnis: {" bo o ","and "," f o o "}
"boo:and:foo".split("o"); // Ergebnis: {" b ",""," : and : f "}
```

```
"A1".matches([A-Z][0-9]) // true  
"a8".matches([A-Z][0-9]) // false
```

- String toLowerCase() Macht jedes Zeichen eines String klein
 - analog: String toUpperCase()





Werkzeugkasten, um Strings zu manipulieren

Blatt4





Werkzeugkasten, um Strings zu manipulieren

- String trim() Entfernt Whitespaces am Anfang und am Ende des Strings
- int compareTo Lexikographischer Vergleich, d.h. alphabetisch

```
"Adam".compareTo("Adam"); // Ergebnis == 0
"Adam".compareTo("Eva"); // Ergebnis < 0
"Eva".compareTo("Adam"); // Ergebnis > 0
"Adam".compareTo("adam"); // Ergebnis < 0
"Eva".compareTo("adam"); // Ergebnis < 0
```

Blatt4



Blatt3



Werkzeugkasten, um Strings zu manipulieren

- String trim() Entfernt Whitespaces am Anfang und am Ende des Strings
- int compareTo Lexikographischer Vergleich, d.h. alphabetisch

```
"Adam".compareTo("Adam"); / / Ergebnis == 0
"Adam".compareTo("Eva"); / / Ergebnis < 0
"Eva".compareTo("Adam"); / / Ergebnis > 0
"Adam".compareTo("adam"); / / Ergebnis < 0
"Eva".compareTo("adam"); / / Ergebnis < 0
```





Werkzeugkasten, um Strings zu manipulieren

- String trim() Entfernt Whitespaces am Anfang und am Ende des Strings
- int compareTo Lexikographischer Vergleich, d.h. alphabetisch

```
"Adam".compareTo("Adam"); / / Ergebnis == 0
"Adam".compareTo("Eva"); / / Ergebnis < 0
"Eva".compareTo("Adam"); / / Ergebnis > 0
"Adam".compareTo("adam"); / / Ergebnis < 0
"Eva".compareTo("adam"); / / Ergebnis < 0
```





Werkzeugkasten, um Strings zu manipulieren

- String trim() Entfernt Whitespaces am Anfang und am Ende des Strings
- int compareTo Lexikographischer Vergleich, d.h. alphabetisch

```
"Adam".compareTo("Adam"); / / Ergebnis == 0
"Adam".compareTo("Eva"); / / Ergebnis < 0
"Eva".compareTo("Adam"); / / Ergebnis > 0
"Adam".compareTo("adam"); / / Ergebnis < 0
"Eva".compareTo("adam"); / / Ergebnis < 0
```



17. / 19.12.2013

Michael Friedrich - Prog Tut Nr. 8

JahresAbschlussQuiz



siehe pdf

Oder: Habt ihr noch WehWehchen, die wir uns anschauen sollen? Zum Beispiel: Listen, Sichtbarkeiten, sonstige Anliegen...?

