Zeichenketten

Klasse `String`

Enthält zahlreiche Methoden – z.B.:

- `int length()`: Anzahl der Zeichen in Zeichenkette
- `String substring(int start)`:
  Unterzeichenkette ab Position `start`
- `boolean equalsIgnoreCase(String s)`:
  Vergleich zweier Zeichenketten, Groß-/Klein-schreibung wird ignoriert
- `boolean startsWith(String s)`: `true`, wenn Zeichenkette mit `s` beginnt
- `int indexOf(String s)`: Position des Beginns einer Unterzeichenkette `s`
- ... ⇒ Handbuch konsultieren !

Klasse String

Beispiel:

```
// Datei: Beispiel10.java

public class Beispiel10 {
        public static void main(String[] args) {
          double d=0;
          int i=0;
          boolean ddef=false, idef=false;
          for(int ac=0; ac<args.length; ac++)
           {
            if(args[ac].startsWith("d=")) {
              d = Double.parseDouble(args[ac].substring(2));
              ddef = true;
            }
           if(args[ac].startsWith("i=")) {
              i = Integer.parseInt(args[ac].substring(2));
              idef = true;
            }
          }
          if(ddef)
             System.out.println("d = " + d);
          if(idef)
             System.out.println("i = " + i);
    }
}
```

Klasse String

Beispiel:

vergleicht Anfang des Argu-ments mit „d=“

```
// Datei: Beispiel10.java

public class Beispiel10 {
        public static void main(String[] args) {
          double d=0;
          int i=0;
          boolean ddef=false, idef=false;
          for(int ac=0; ac<args.length; ac++)
           {
            if(args[ac].startsWith("d=")) {
               d = Double.parseDouble(args[ac].substring(2));
               ddef = true;
            }
           if(args[ac].startsWith("i=")) {
             i = Integer.parseInt(args[ac].substring(2));
             idef = true;
            }
          }
         if(ddef)
            System.out.println("d = " + d);
         if(idef)
            System.out.println("i = " + i);
    }
}
```

Klasse String

Beispiel:

```
// Datei: Beispiel10.java

public class Beispiel10 {
        public static void main(String[] args) {
          double d=0;
          int i=0;
          boolean ddef=false, idef=false;
          for(int ac=0; ac<args.length; ac++)
           {
            if(args[ac].startsWith("d=")) {
              d = Double.parseDouble(args[ac].substring(2));
              ddef = true;
            }
           if(args[ac].startsWith("i=")) {
              i = Integer.parseInt(args[ac].substring(2));
              idef = true;
            }
          }
          if(ddef)
             System.out.println("d = " + d);
          if(idef)
             System.out.println("i = " + i);
    }
}
```

Klasse String

Beispiel:

konver-tiert Rest des Argu-ments in double

```
// Datei: Beispiel10.java

public class Beispiel10 {
        public static void main(String[] args) {
          double d=0;
          int i=0;
          boolean ddef=false, idef=false;
          for(int ac=0; ac<args.length; ac++)
           {
            if(args[ac].startsWith("d=")) {
              d = Double.parseDouble(args[ac].substring(2));
              ddef = true;
            }
           if(args[ac].startsWith("i=")) {
             i = Integer.parseInt(args[ac].substring(2));
             idef = true;
            }
          }
        if(ddef)
           System.out.println("d = " + d);
        if(idef)
           System.out.println("i = " + i);
    }
}
```

Klasse String

Beispiel:

```
// Datei: Beispiel10.java

public class Beispiel10 {
        public static void main(String[] args) {
          double d=0;
          int i=0;
          boolean ddef=false, idef=false;
          for(int ac=0; ac<args.length; ac++)
           {
            if(args[ac].startsWith("d=")) {
              d = Double.parseDouble(args[ac].substring(2));
              ddef = true;
            }
           if(args[ac].startsWith("i=")) {
              i = Integer.parseInt(args[ac].substring(2));
              idef = true;
            }
          }
          if(ddef)
             System.out.println("d = " + d);
          if(idef)
             System.out.println("i = " + i);
    }
}
```

Klasse `String`

Beispiel:

```
// Datei: Beispiel10.java

public class Beispiel10 {
        public static void main(String[] args) {
          double d=0;
          int i=0;
          boolean ddef=false, idef=false;
          for(int ac=0; ac<args.length; ac++)
           {
            if(args[ac].startsWith("d=")) {
              d = Double.parseDouble(args[ac].substring(2));
              ddef = true;
            }
           if(args[ac].startsWith("i=")) {
            i = Integer.parseInt(args[ac].substring(2));
            idef = true;
           }
          }
        if(ddef)
           System.out.println("d = " + d);
        if(idef)
           System.out.println("i = " + i);
    }
}
```

analog für „`i=`“ und `int`

Klasse `String`

Beispiel:

```
// Datei: Beispiel10.java

public class Beispiel10 {
        public static void main(String[] args) {
          double d=0;
          int i=0;
          boolean ddef=false, idef=false;
          for(int ac=0; ac<args.length; ac++)
           {
            if(args[ac].startsWith("d=")) {
              d = Double.parseDouble(args[ac].substring(2));
              ddef = true;
            }
           if(args[ac].startsWith("i=")) {
            i = Integer.parseInt(args[ac].substring(2));
            idef = true;
           }
         }
        if(ddef)
           System.out.println("d = " + d);
        if(idef)
           System.out.println("i = " + i);
    }
}
```

analog für „`i=`“ und `int`

Beispielaufruf:

```
java Beispiel10 i=493 d=-123.456
```

Ausnahmebehandlung

Ausnahmebehandlung

Behandlung von Laufzeit-Fehlern in Java:

- Fehler (z.B. Division durch 0): Löse Ausnahme aus durch Erzeugen eines Objekts einer Ausnahmeklasse
- Methode kann Ausnahme selbst behandeln oder an aufrufende Methode weiterreichen
- Werden Ausnahme nur weitergereicht und nicht behandelt ⇒ Programm bricht mit Fehlermeldung ab

- Einige Ausnahmen müssen beachtet werden
- Beispiel:

```
throw new RuntimeException("Division durch Null");
```

in Klasse `Rational` oder `Complex`: Gibt Fehlermeldung aus und bricht Programm ab

Ausnahmen behandeln

```
try {
  Anweisungen1
 }
catch (Ausnahmetyp Bezeichner) {
  Anweisungen2
 }
```

- Tritt beim Ausführen von *Anweisungen1* eine Ausnahme vom Typ *Ausnahmetyp* auf, werden *Anweisungen2* angesprungen

Beispiel:

```
// Datei: Beispiel11.java

public class Beispiel11 {
          public static void main(String[] args) {
            int a=4, b=-4, c;
            try {
              c = a+b;
              b = a/c;
              System.out.println("b = " + b);
             }
            catch (ArithmeticException e) {
              System.out.println("Ausnahme: " +  e.getMessage() );
             }
     }
}
```

Ausnahmen weitergeben

Vollständige Methoden-Deklaration:

```
[Modifizierer] Typ Methodenname([Parameter])
 [throws Ausnahme-Liste] {
  Anweisungen
 }
```

- *Ausnahme-Liste*: Eine oder mehrere durch Kommas getrennte Ausnahmeklassen
- Bei der Methodendeklaration aufgeführte Ausnahmeklassen werden an die aufrufende Methode übergeben

Ausnahmen auslösen

- Ausnahme erzeugen: über Konstruktor der Ausnahmeklasse (ggfs. mit Text als Parameter)
- auslösen über:

```
throw Ausnahmeobjekt;
```

Beispiel (s. vorne):

```
throw new RuntimeException("Division durch Null");
```

Ein- / Ausgabe

Eingabe von Konsole

```
// Datei: Beispiel12.java
import java.io.*;    // Importiere Ein-/Ausgabeklassen

public class Beispiel12 {
        // Achtung: IOException moeglich
        public static void main(String[] args)
           throws IOException {
          // BufferedReader liest aus einem Reader
          BufferedReader ein = new BufferedReader(
            // Oeffene Eingabestrom auf Eingabe
            new InputStreamReader(System.in) );
          System.out.print("Bitte Text eingeben: ");
          // verwende Methode readLine() von BufferedReader
          String line = ein.readLine();
          System.out.println("Text: " + line);
          ein.close();        // Schliesse den BufferedReader
    }
}
```

- Ein-/Ausgabe-klassen müssen „`import`“iert werden
- verwende `InputStream` „`in`“ der Klasse System

- öffne einen `InputStreamReader` auf `InputStream System.in`
- öffne einen `BufferedReader` darüber
- dieser stellt die Methode `readLine()` zur Verfügung

Ausgabe in Datei

```
// Datei: Beispiel13.java
import java.io.*;       // Importiere Ein-/Ausgabeklassen

public class Beispiel13 {
        // Achtung: IOException moeglich
        public static void main(String[] args)
              throws IOException {
          // PrintWriter stellt print-Methoden zur Verfuegung
          PrintWriter Datei = new PrintWriter(
            // Oeffne Ausgabedatei "Test-datei.tst"
            new FileWriter("Test-datei.tst") );
          // Schreibe in Datei
          Datei.println("Hier\tist\neine\tTestzahl:\t" + 17);
          Datei.close();// Schliesse Datei
    }
}
```

- `Filewriter` schreibt in eine Datei

- `PrintWriter` stellt die Methoden `print()` und `println()` zur Verfügung (vgl. `System.out`)
- Die Methode `close()` schließt die Datei

Lesen aus Datei

```
// Datei: Beispiel14.java
import java.io.*;

public class Beispiel14 {
        public static void main(String[] args) throws IOException {
          LineNumberReader ein = new LineNumberReader(
            new FileReader("Beispiel14.java") );
          String line;
          while((line = ein.readLine()) != null)
            System.out.println(ein.getLineNumber() // Zeilennummer
            + " : " + line);
          ein.close();  // Schliesse Eingabe
    }
}
```

- `FileReader` erlaubt Zugriff auf Datei
- `LineNumberReader` liest aus Eingabestrom und stellt die Methode `getLineNumber()` zur Verfügung
- `BufferedReader`: dito, jedoch ohne Zeilennummer; stellt die Methode `readLine()` zur Verfügung (`readLine()` gibt am Dateiende `null` zurück)

Lesen aus Datei

```
// Datei: Beispiel14.java
import java.io.*;

public class Beispiel14 {
        public static void main(String[] args) throws IOException {
          LineNumberReader ein = new LineNumberReader(
            new FileReader("Beispiel14.java") );
          String line;
          while((line = ein.readLine()) != null)
            System.out.println(ein.getLineNumber() // Zeilennummer
            + " : " + line);
          ein.close();  // Schliesse Eingabe
    }
}
```

mehr im Handbuch

- `FileReader` erlaubt Zugriff auf Datei
- `LineNumberReader` liest aus Eingabestrom und stellt die Methode `getLineNumber()` zur Verfügung
- `BufferedReader`: dito, jedoch ohne Zeilennummer; stellt die Methode `readLine()` zur Verfügung (`readLine()` gibt am Dateiende `null` zurück)