

13.9 Bearbeitung von mehreren Objekten

- Datentyp Vector
- Sequentielle Verarbeitung
- Lesen von Dateien

Verbesserter Wetterbericht

Eine Zeitung erhält eine Liste mit Namen von Orten im In- und Ausland, sowie den Mittagstemperaturen des Vortags. Die Temperaturen sind entweder in Celsius oder in Fahrenheit angegeben.

Aufgabe:

Erstellung einer Liste aus Städtenamen sowie Temperatur in Celsius und Fahrenheit. **In der Liste sollen die Städtenamen, sowie die Temperaturen jeweils in einer Spalte senkrecht untereinanderstehen.**

Konsequenz: Die Ausgabe kann erst beginnen, nachdem sämtliche Daten eingelesen sind.

Vorige Lösung

```
String town = reader.readString("Type the name of the town: ");
while (town != null && !town.equals(""))
{
    String code = reader.readString("Type scale (C or F) and temperature: ");
    code = code.trim().toUpperCase();
    char tempScale = code.charAt(0);
    int temp = new Integer (code.substring(1)).intValue();
    if ( !(tempScale == 'F' || tempScale == 'C') )
    {
        System.out.println("WeatherList: bad scale, " + tempScale);
    }
    else
    {
        int fahrenheit;
        int celsius;
        if ( tempScale == 'F' )
        {
            fahrenheit = temp;
            celsius = (int) calc.fahrenheitIntoCelsius(temp);
        }
        else // tempScale == 'C'
        {
            celsius = temp;
            fahrenheit = (int) calc.celsiusIntoFahrenheit(temp);
        }
        System.out.println (town + ": " + celsius + "C " + fahrenheit + "F");
    }
    town = reader.readString("Type the name of the next town: ");
}
}
```

Eine Klasse für Wetterdaten

WeatherData

Attribut	Aufgabe
private String town	Name der Stadt
private int tempCelsius	Temperatur in Celsius
private int tempFahrenheit	Temperatur in Fahrenheit
Methode	Aufgabe
String town()	Akzessor
int tempCelsius()	Akzessor
int tempFahrenheit()	Akzessor

Einlesen von Wetterdaten

```
public static WeatherData readWeatherData (LineReader reader, TempCalculator calc)
{
    String town = reader.readString("Type the name of the town: ");
    if (town == null || town.equals("")) { return null; }
    String code = reader.readString("Type scale (C or F) and temperature: ");
    code = code.trim().toUpperCase();
    char tempScale = code.charAt(0); // attention
    int temp = new Integer (code.substring(1)).intValue();
    if ( !(tempScale == 'F' || tempScale == 'C') )
    {
        System.out.println("readWeatherData: bad scale, " + tempScale); return null;
    }
    else
    {
        int fahrenheit;
        int celsius;
        if ( tempScale == 'F' )
        {
            fahrenheit = temp;
            celsius = (int) calc.fahrenheitIntoCelsius(temp);
        }
        else // tempScale == 'C'
        {
            celsius = temp;
            fahrenheit = (int) calc.celsiusIntoFahrenheit(temp);
        }
        return new WeatherData(town, celsius, fahrenheit);
    }
}
}
```

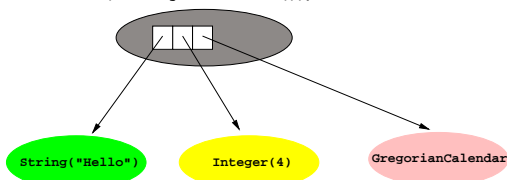
13.9.1 Datentyp Vector

- **Aggregat-Datentyp:** ein Vector-Objekt kann *mehrere* andere Objekte enthalten
Auch: *Container* Objekt/Klasse/Datentyp
- Enthaltene Objekte dürfen *beliebige, unterschiedliche* (Objekt-) Typen haben
Nicht aber: Elemente von primitiven Datentypen, wie int, double, usw
- Operationen (Ausschnitt)

new Vector ()	Erzeugen eines Vectors
addElement (obj)	Hinzufügen eines Elements
removeElement (obj)	Entfernen eines Elements
elements ()	Verarbeiten aller Elemente
- vordefiniert in Package java.util

Beispiel

```
Vector v = new Vector ();
v.addElement("Hello");
v.addElement(new Integer(4));
v.addElement(new GregorianCalendar());
```



Einlesen und Aufbau des Vectors

```
Vector allWeather = new Vector ();
WeatherData w = readWeatherData(reader, calc);
while (w != null)
{
    allWeather.addElement(w);
    w = readWeatherData(reader, calc);
}
}
```

Verarbeitung eines Datensatzes

```
int maxTownLength = 0;
int maxCelsiusLength = 0;
int maxFahrenheitLength = 0;

WeatherData w = ...;
int townLength = w.town().length();
if (maxTownLength < townLength)
{ maxTownLength = townLength; }
int celsiusLength = w.tempCelsius().toString().length();
if (maxCelsiusLength < celsiusLength)
{ maxCelsiusLength = celsiusLength; }
int fahrenheitLength = w.tempFahrenheit().toString().length();
if (maxFahrenheitLength < fahrenheitLength)
{ maxFahrenheitLength = fahrenheitLength; }
```

13.9.2 Sequentielle Verarbeitung

Muster: Durchlaufen aller Elemente eines Vectors

```
Enumeration enum = allWeather.elements();
while (enum.hasMoreElements())
{
    Object elem = enum.nextElement();
    // Verarbeitung von elem
}

• Methode elements() liefert ein Enumeration Objekt
• Enumeration hat Methoden
  - hasMoreElements() : boolean
    testet, ob weitere Elemente vorhanden
  - nextElement() : Object
    liefert das nächste Element
```

Typkonversion und die Klasse Object

Problem mit nextElement() : Object

- liefert Ergebnis vom Typ Object
- jede Klasse ist implizit Subklasse der Klasse Object (aus Package java.lang)
- einzige Möglichkeit, da jedes Element eines Vectors zu einer anderen Klasse gehören kann
- Aber: wir benötigen ein WeatherData-Objekt
- Lösung: Da WeatherData < Object, ist ein Typcast möglich. Der Ausdruck (WeatherData) allWeather.nextElement() hat den Typ WeatherData. Dabei wird getestet, dass allWeather.nextElement() tatsächlich ein Objekt der Klasse WeatherData liefert. Andernfalls erfolgt ein Laufzeitfehler.

Berechnung der Maxima

```
int maxTownLength = 0;
int maxCelsiusLength = 0;
int maxFahrenheitLength = 0;

Enumeration enum = allWeather.elements();
while (enum.hasMoreElements())
{
    w = (WeatherData) enum.nextElement();
    int townLength = w.town().length();
    if (maxTownLength < townLength)
    { maxTownLength = townLength; }
    int celsiusLength = w.tempCelsius().toString().length();
    if (maxCelsiusLength < celsiusLength)
    { maxCelsiusLength = celsiusLength; }
    int fahrenheitLength = w.tempFahrenheit().toString().length();
    if (maxFahrenheitLength < fahrenheitLength)
    { maxFahrenheitLength = fahrenheitLength; }
}
```

Erzeugen der Ausgabe

```
enum = allWeather.elements();
while (enum.hasMoreElements())
{
    w = (WeatherData) enum.nextElement();
    printLeftAligned(w.town(), maxTownLength + 1);
    printLeftAligned("C" + w.tempCelsius(), maxCelsiusLength + 2);
    printLeftAligned("F" + w.tempFahrenheit(), maxFahrenheitLength + 2);
    System.out.println ();
}
```

Hilfsfunktion

```
/** printLeftAligned prints a string in a field of given size
 * @param str string to be printed
 * @param maxLength size of output field */
public static void printLeftAligned(String str, int maxLength)
{
    System.out.print(str);
    int spaces = maxLength - str.length();
    while (spaces > 0)
    {
        System.out.print (" ");
        spaces = spaces - 1;
    }
}
```

Beispielausgabe

```
> java WeatherData2 WeatherData.txt
New York      C6  F44
Berlin        C0  F32
Paris         C4  F39
San Francisco C16 F62
Honolulu      C27 F81
Moskau        C-2 F28
Helsinki      C-6 F21
Dallas        C11 F52
```

13.9.3 Lesen von Dateien

Weitere Konstruktormethode von LineReader

```
/** Constructor LineReader constructs the input-view object
 * @param fileName - the name of the input file */
public LineReader(String fileName)
{
    InputStream is = null;
    try { is = new FileInputStream (fileName); }
    catch (IOException e)
    { is = System.in; }
    k = new BufferedReader(new InputStreamReader(is));
}

• new FileInputStream (fileName) liefert ein Objekt, über das die Datei
  fileName gelesen werden kann
```

Verwendung des erweiterten LineReader

```
public class WeatherList2
{
    public static void main(String[] args)
    {
        LineReader reader;
        if (args.length > 0)
            { reader = new LineReader(args[0]); }
        else
            { reader = new LineReader(System.in); }
    }
}
```