Kapitel 10 - Utilities: Math, Date, Calendar, System, Random

Inhalt - Überblick

Exceptions

- Enthält Methoden zur Fließkommaarithmetik
- Winkelfunktionen (sin, cos, tan)

```
double result = 0;
result = Math.sin(1);
result = Math.cos(1);
result = Math.tan(1);
```

[MathProgram.java]

- Minimum und Maximum

```
result = Math.max(1, 2);
result = Math.max(1L, 2L);
result = Math.max(1.0F, 2.0F);
result = Math.max(1.0D, 2.0D);

result = Math.min(1, 2);
result = Math.min(1L, 2L);
result = Math.min(1.0F, 2.0F);
result = Math.min(1.0D, 2.0D);
```

[MathProgram.java]

- Exponentialfunktion, natürlicher Logarithmus

```
// Exponentialfunktion
result = Math.exp(1);

// Natürlicher Logarithmus
result = Math.log(1);
```

- Logarithmus zu einer anderen Basis $\log_b r = \frac{\log_a r}{\log_a b}$

```
public static double log(double b, double x){
    return Math.log(x) / Math.log(b);
}
```

[MathProgram.java]

- Potenzieren, Wurzel

```
// Potenzieren: "2 hoch 8"
result = Math.pow(2, 8);

// Wurzel
result = Math.sqrt(100);

// Alternative:
result = Math.pow(100, 0.5);
```

[MathProgram.java]

- Runden

```
// Runden
result = Math.round(12.897);

// Runden mit Genauigkeitsangabe (Nachkommastellen)
result = round(12.897343, 3);

public static double round(double x, int precision){
    double f = Math.pow(10, precision);
    double result = x * f;
    result = Math.round(result);
    result = result / f;
    return result;
}
```

[MathProgram.java]

- Nächstgrößere bzw. nächstkleinere Zahl (Abschneiden)

```
// nächstgrößere Ganzzahl
result = Math.ceil(17.55);

// nächstkleinere Ganzzahl
result = Math.floor(17.55);
```

[MathProgram.java]

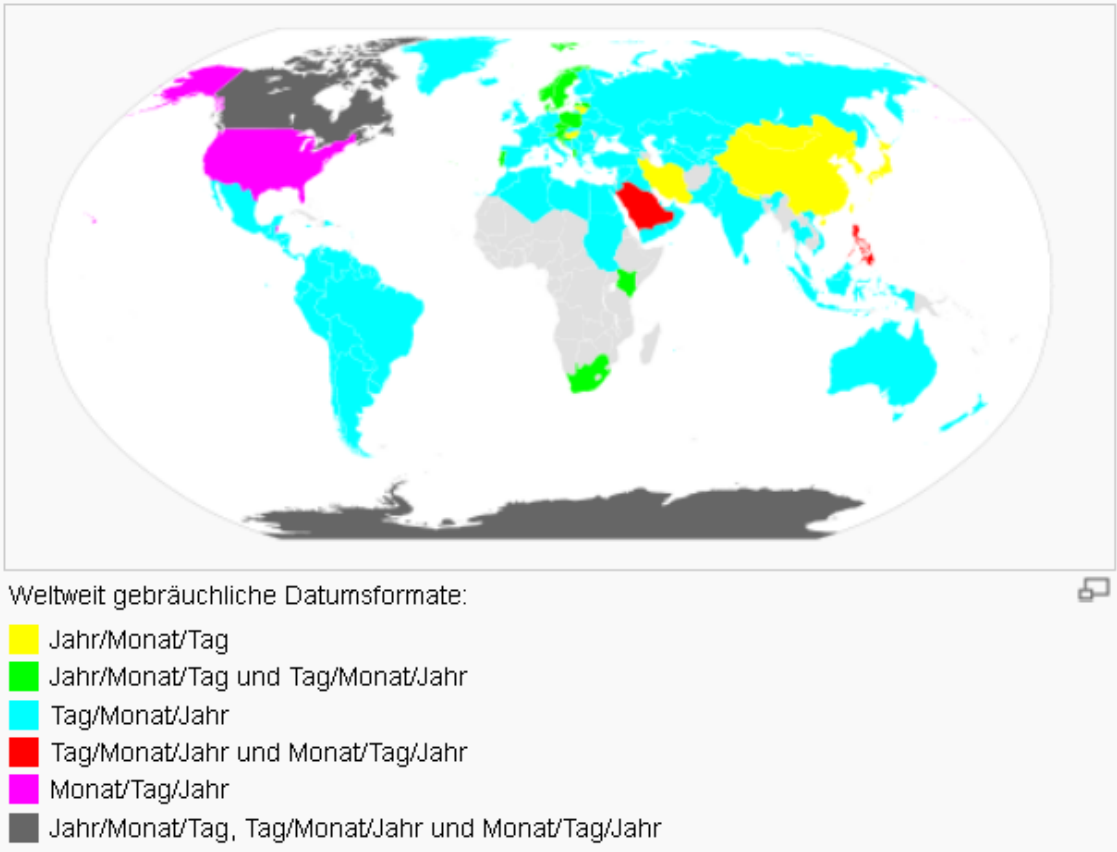
Method	Description
exp(a)	Natural number **e** raised to the power of **a**.
log(a)	Natural logarithm (base **e**) of **a**.
floor(a)	The largest whole number less than or equal to **a**.
max(a,b)	The larger of a and b.
pow(a,b)	The number **a** raised to the power of **b**.
sqrt(a)	The square root of **a**.
sin(a)	The sine of **a**. (Note: all trigonometric functions are computed in radians)

Date, Calendar

- Date wird zur Darstellung von Datumswerten genutzt
- Hierzu zählen auch Uhrzeiten

- Zum Rechnen mit Datumswerten steht die Klasse GregorianCalendar zur Verfügung.

- Weltweit gebräuchliche Datumsformate

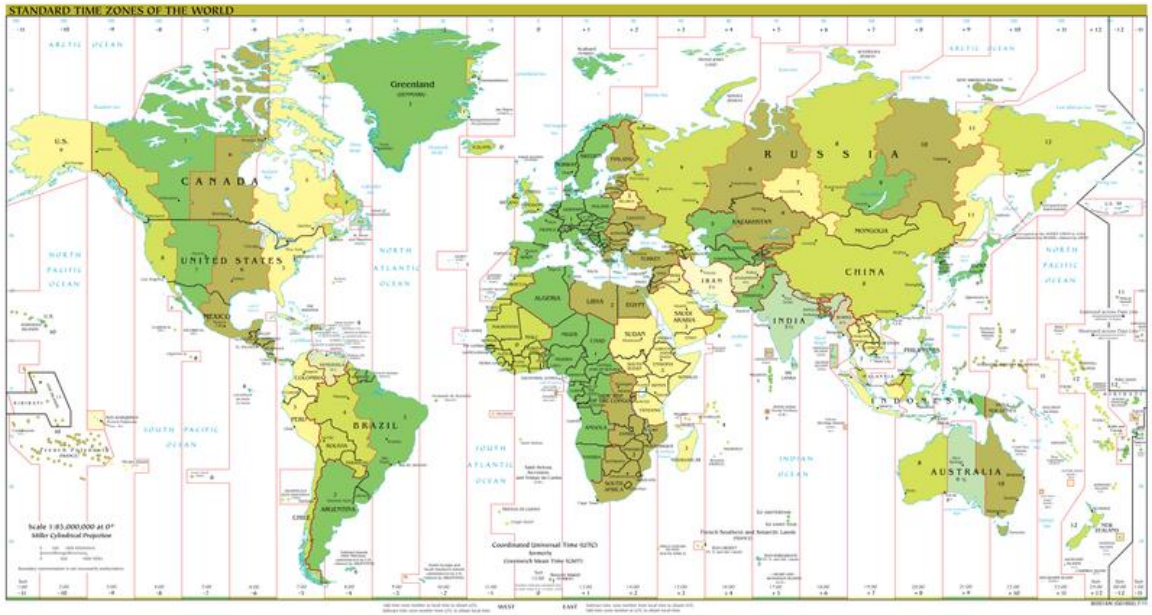


Quelle: http://de.wikipedia.org/wiki/Datumsformat

[MathProgram.java]

Datumsarithmetik

- Datumsarithmetik ist keine „ganz einfache Sache“
  - Internationale Ausgabeformate (12 / 24h)
  - Zeitzonen (GMT / UTC)
  - Schaltjahre und Schaltsekunden



24-Stunden-Zählung	2-mal-12-Stunden-Zählung
00:00	12:00 (mitternachts)*
00:59	12:59 a.m. (ante meridiem)
01:00	01:00 a.m.
02:00	02:00 a.m.
…	…
10:00	10:00 a.m.
11:00	11:00 a.m.
12:00	12:00 (mittags, *noon*)**
12:59	12:59 p.m. (post meridiem)
13:00	01:00 p.m.
14:00	02:00 p.m.
…	…
22:00	10:00 p.m.
23:00	11:00 p.m.
24:00 (selten)	12:00 (mitternachts)*

* zweideutiges Datum selten aufgelöst durch 0:00 a.m.

** 12:00 mittag ist weder a.m. noch p.m., alternativ 11:59

[MathProgram.java]

- Die Anzahl der Sekunden seit dem 1.1.1970 UTC
- „The Epoch“
- Das Unix-Millenium wurde am 9. September 2001 in Kopenhagen gefeiert:

- Jahr 2038-Problem: 2.147.483.647

- Ursprünglich für Datumsangaben vorgesehen: java.util.Date
- Repräsentation von Zeitpunkten in Java
- Kann bspw. keine Zeitzonenangaben verarbeiten
- Kennt keine Schaltsekunden

Constructor Summary

`Date()` Allocates a `Date` object and initializes it so that it represents the time at which it was allocated, measured to the nearest millisecond.
`Date(int year, int month, int date)` **Deprecated.** *As of JDK version 1.1, replaced by `Calendar.set(year + 1900, month, date)` or `GregorianCalendar(year + 1900, month, date)`.*
`Date(int year, int month, int date, int hrs, int min)` **Deprecated.** *As of JDK version 1.1, replaced by `Calendar.set(year + 1900, month, date, hrs, min)` or `GregorianCalendar(year + 1900, month, date, hrs, min)`.*
`Date(int year, int month, int date, int hrs, int min, int sec)` **Deprecated.** *As of JDK version 1.1, replaced by `Calendar.set(year + 1900, month, date, hrs, min, sec)` or `GregorianCalendar(year + 1900, month, date, hrs, min, sec)`.*
`Date(long date)` Allocates a `Date` object and initializes it to represent the specified number of milliseconds since the standard base time known as "the epoch", namely January 1, 1970, 00:00:00 GMT.
`Date(String s)` **Deprecated.** *As of JDK version 1.1, replaced by `DateFormat.parse(String s)`.*

java.util.Calendar

- Zeitzonen
- Rechnen mit Datumswerten (z. B. 36 Stunden addieren)
  - Methode: add(int field, int amount)
  - Berücksichtigt „Grenzwertüberschreitungen“, bspw.

```
cal.add(Calendar.HOUR, 48);
```

- Datumswerte vergleichen
- Datumswerte in Zeitzonen umrechnen
- Einzelne Komponenten eines Datums (z. B. Stunde) auslesen
- Konvertierung zwischen Date und Calendar (setTime, getTime)

[DateProgram.java]

- Formatierte Anzeige von Datumswerten in einem String

```
// Datumsanzeige formatieren
DateFormat sdf  = SimpleDateFormat.getInstance();
System.out.println(sdf.format(cal.getTime()));
```

```
23.06.11 19:47
```

- Format-Pattern siehe JavaDoc, beispielsweise:

```
new SimpleDateFormat("yyyy.MM.dd G 'at' HH:mm:ss z");
```

```
2011.06.23 n. Chr. at 19:54:52 MESZ
```

System

- Utility-Klasse um auf die Systemumgebung zuzugreifen
- Zugreifen auf **Java-Umgebungseigenschaften**

```
System.getProperty("file.separator");
```

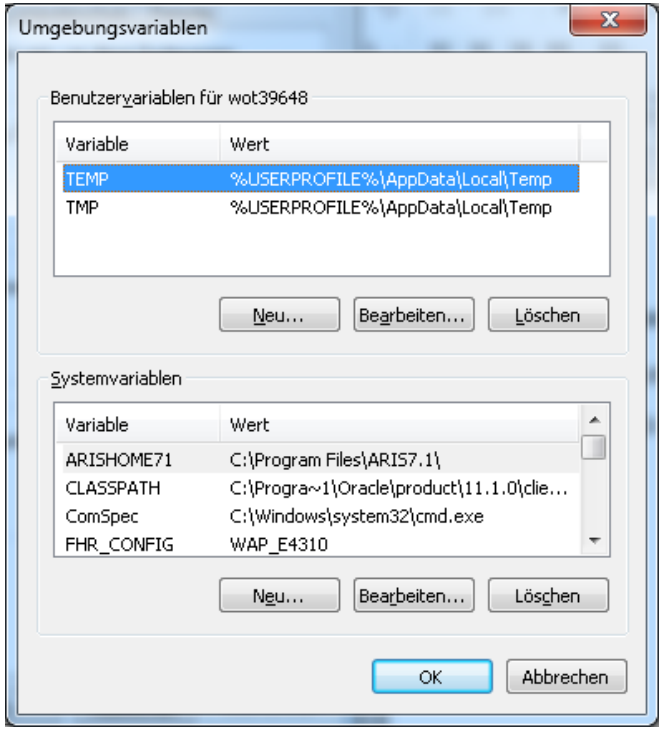
Property	Bedeutung
java.version	Java-Versionsnummer
java.vendor	Herstellerspezifische Zeichenkette
java.vendor.url	URL (also ein Internet-Link) zum Hersteller
java.home	Installationsverzeichnis
java.class.version	Versionsnummer der Java-Klassenbibliothek
java.class.path	Aktueller Klassenpfad
os.name	Name des Betriebssystems
os.arch	Betriebssystem-Architektur
os.version	Versionsnummer des Betriebssystems
file.separator	Trennzeichen für die Bestandteile eines Pfadnamens
path.separator	Trennzeichen für die Laufwerksangabe eines Pfadnamens
line.separator	Zeichenkette für Zeilenschaltung
user.name	Name des angemeldeten Benutzers
user.home	Home-Verzeichnis
user.dir	Aktuelles Arbeitsverzeichnis

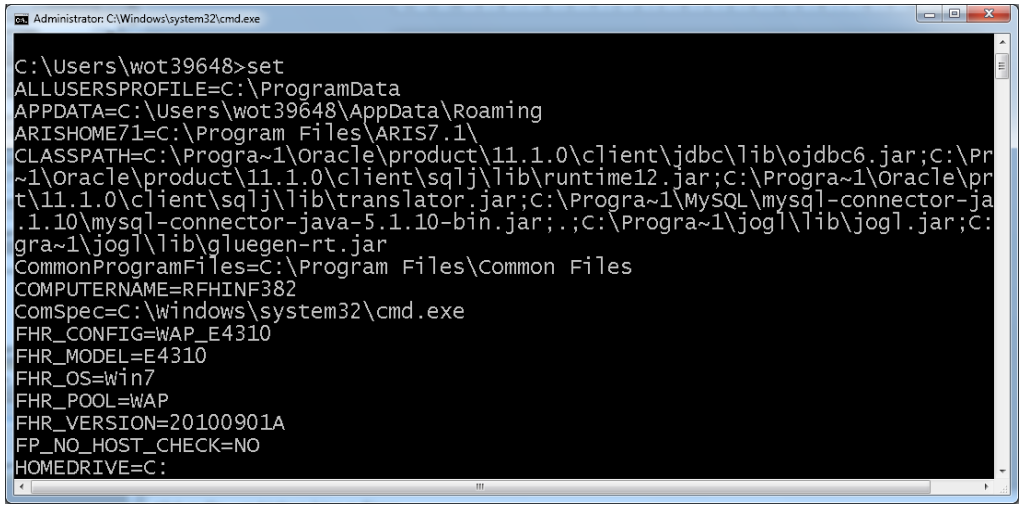
...

[SystemProgram.java]

- Zugreifen auf **Betriebssystem-Umgebungseigenschaften**

```
System.getenv("CLASSPATH");
```





- Das Programm mit einem **errorlevel** beenden

- Dabei steht 0 für ein fehlerfreies Programmende:

```
System.exit(0);
```

- Werte > 0 zeigen einen Fehler an, entsprechend einer selbst zu definierenden Fehlercode-Liste:

```
System.exit(9);
```

- Den GarbageCollector manuell aufrufen

```
System.gc();
```

- Den Systemzeitgeber abfragen (Systemuhr)

```
System.currentTimeMillis();
```

- Ein externes Programm aufrufen

```
// Ein externes Programm aufrufen
Runtime rt = Runtime.getRuntime();
rt.exec("notepad");
```

- Die Anzahl der CPUs abfragen

```
int cpus = rt.availableProcessors();
```

Random

- Random mit einem festen Seed anlegen
- Zu jedem Seed gehört eine feste Folge von gleichverteilten Zufallszahlen

```
Random ran = new Random(100);
ran.nextBoolean();
ran.nextDouble();
ran.nextFloat();
ran.nextInt();
ran.nextLong();
```

Seed

- Random ohne Seed; Die Systemzeit wird ersatzweise genutzt

```
Random ran2 = new Random();
```

- Wie kann man den Wertebereich der Zufallszahlen einschränken?

- „Bessere“ Zufallszahlen (s. Krüger, S. 1203)
- Kryptographische Zufallszahlen

```
SecureRandom sr = null;
try {
    sr = SecureRandom.getInstance("SHA1PRNG");
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```