Universität Bremen

Praktische Informatik 1
Bibliotheksklassen nutzen

Thomas Röfer

Cyber-Physical Systems
Deutsches Forschungszentrum für Künstliche Intelligenz

Multisensorische Interaktive Systeme
Fachbereich 3, Universität Bremen

Bild: ©Michael Ihle

Universität Bremen

Nützliche Bibliotheksklassen und -Schnittstellen

- **String**: Umgang mit Zeichenketten
- **List<E>** (**ArrayList<E>**, **LinkedList<E>**): Verwaltet Listen von Werten, deren Reihenfolge erhalten bleibt
- **Set<E>** (**HashSet<E>**, **TreeSet<E>**): Verwaltet Mengen von Werten. Feststellen, ob ein Wert enthalten ist, geht schnell
- **Map<K, V>** (**HashMap<K, V>**, **TreeMap<K, V>**): Verwaltet Abbildungen von Schlüsseln auf Werte. Auffinden von Werten anhand ihres Schlüssels geht schnell
- **List<E>**, **Set<E>** und **Map<K, V>** sind Schnittstellen (**Interfaces**), die festlegen, welche Funktionalität bereitgestellt wird, sie aber nicht implementieren

Universität Bremen

String

- Alle Strings sind **konstant**, d.h. **String** enthält keine Methoden, die den Inhalt eines Strings verändern
- Der Inhalt von Strings wird mit **equals** oder **equalsIgnoreCase** verglichen
  - **a.equals(b)** statt **a == b**
  - Dies gilt nicht nur für **String**s, sondern generell für Objekte!

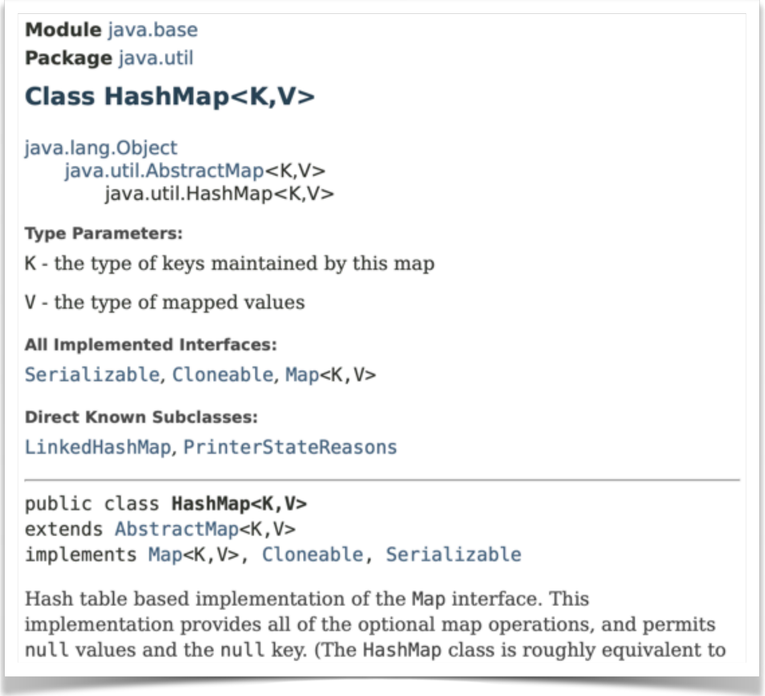
```
private String getReplacement(final String word)
{
    for (final String[ ] replacement : replacements) {
        if (word.equalsIgnoreCase(replacement[0])) {
            return replacement[1];
        }
        else if (word.equalsIgnoreCase(replacement[1])) {
            return replacement[0];
        }
    }
    return word;
}
```

- Beispiele für Methoden: **length**, **contains**, **substring**, **replace**, **split**, **toLowerCase**, **charAt**

Universität Bremen

Dokumentation zu Bibliotheksklassen

- **Name** der Klasse
- **Typ-Parameter** der Klasse
- Welche **Schnittstellen** implementiert sie?
- Allgemeine Beschreibung des **Zwecks** der **Klasse**
- Liste der **Konstruktoren** und **Methoden** der Klasse (alphabetisch sortiert)
- Beschreibung des **Zwecks** jedes **Konstruktors** und jeder **Methode** (logisch gruppiert)
- **Parameter** (und **Ergebnistypen**) jedes Konstruktors und jeder Methode



Module java.base
Package java.util

Class HashMap<K,V>

java.lang.Object
  java.util.AbstractMap<K,V>
    java.util.HashMap<K,V>

Type Parameters:

K - the type of keys maintained by this map

V - the type of mapped values

All Implemented Interfaces:

Serializable, Cloneable, Map<K,V>

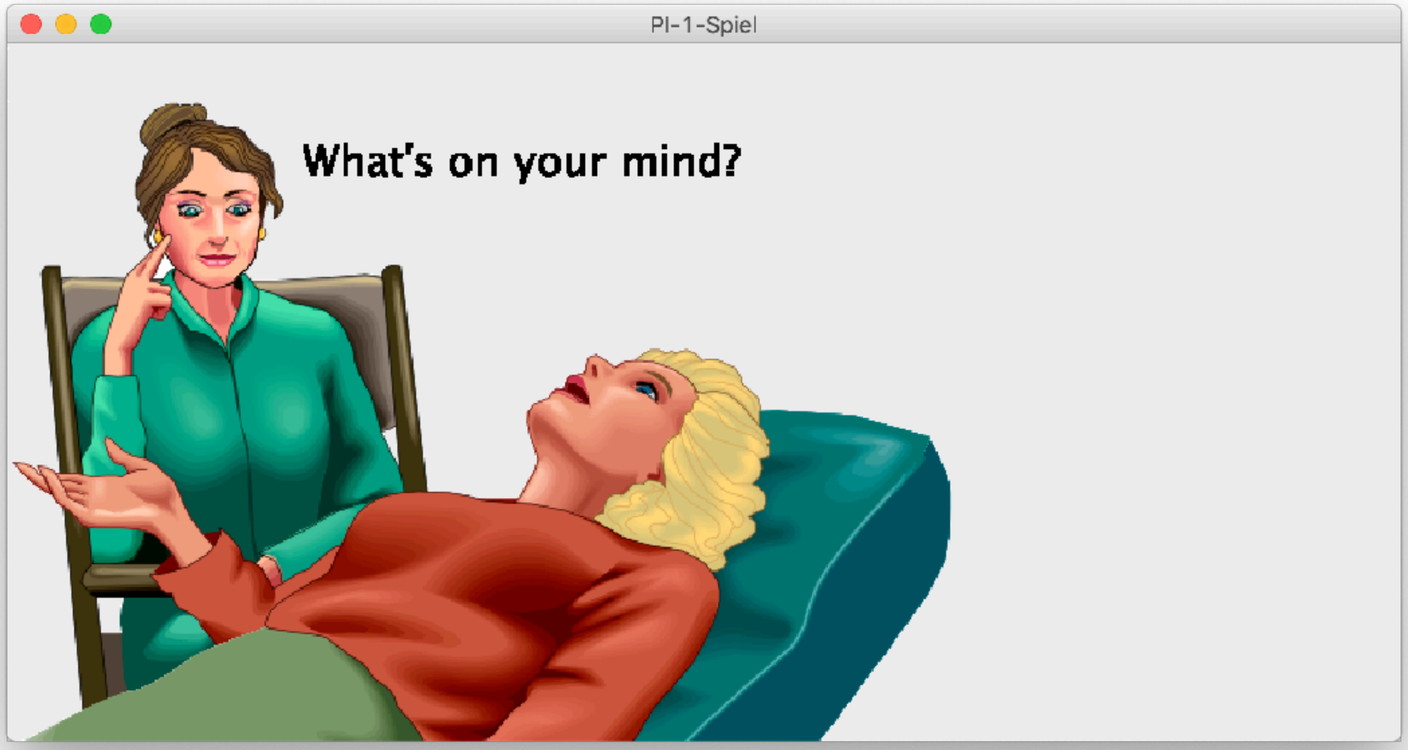
Direct Known Subclasses:

LinkedHashMap, PrinterStateReasons

```
public class HashMap<K,V>
extends AbstractMap<K,V>
implements Map<K,V>, Cloneable, Serializable
```

Hash table based implementation of the Map interface. This implementation provides all of the optional map operations, and permits null values and the null key. (The HashMap class is roughly equivalent to

ELIZA: Demo



Universität Bremen

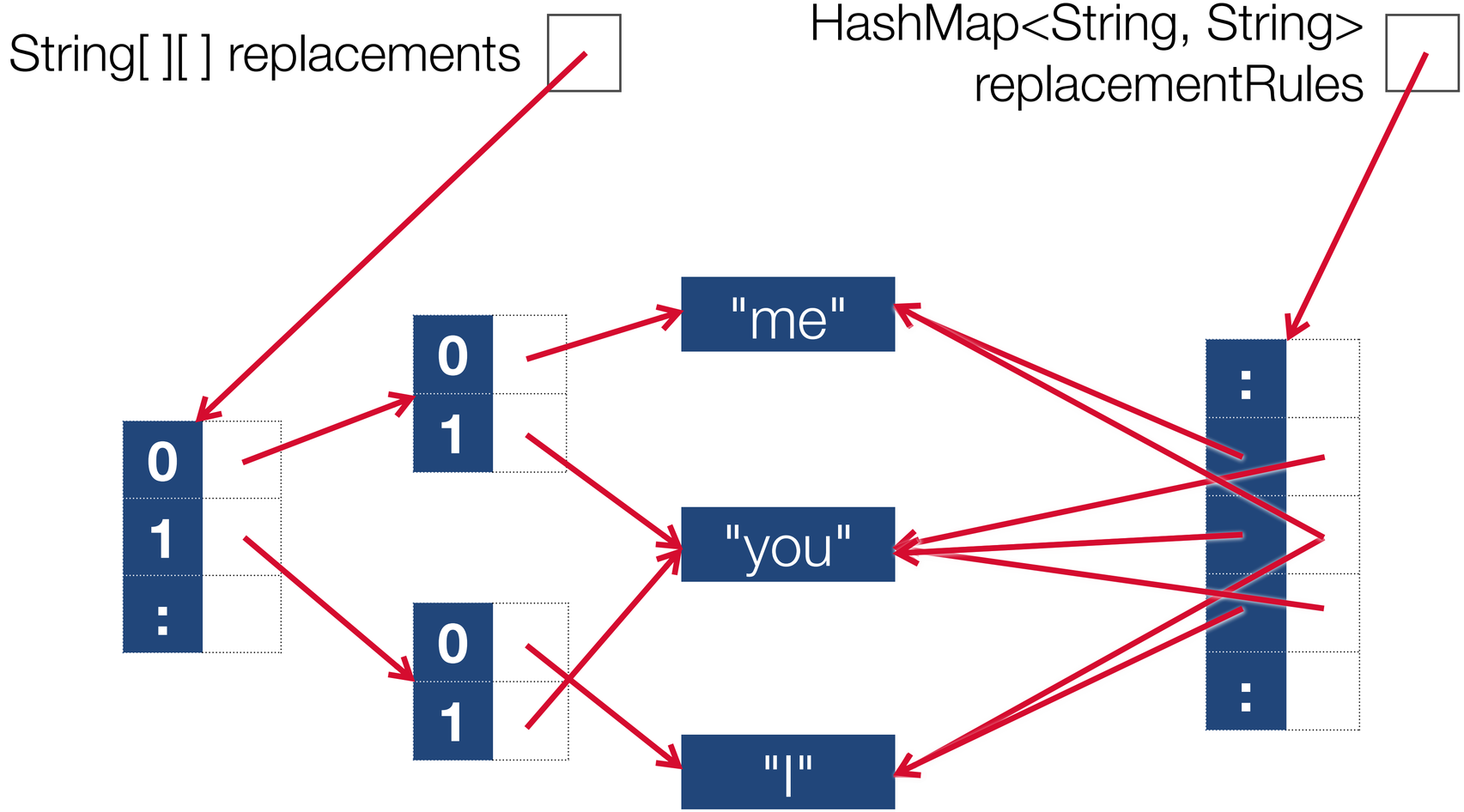
Map<K, V> (HashMap<K, V>)

- Bildet Schlüssel vom Typ **K** auf Werte vom Typ **V** ab
- **V put(K key, V value)** speichert ein Paar aus Schlüssel und Wert und liefert den alten Wert zurück
- **V get(Object key)** liefert den Wert zum Schlüssel **key** oder **null**, falls **key** nicht enthalten ist
  - **V getOrDefault(Object key, V defaultValue)** erlaubt stattdessen, einen Standardwert anzugeben
- **V remove(Object key)** entfernt das Schlüssel/Wertepaar aus der **Map** und liefert den Wert zurück

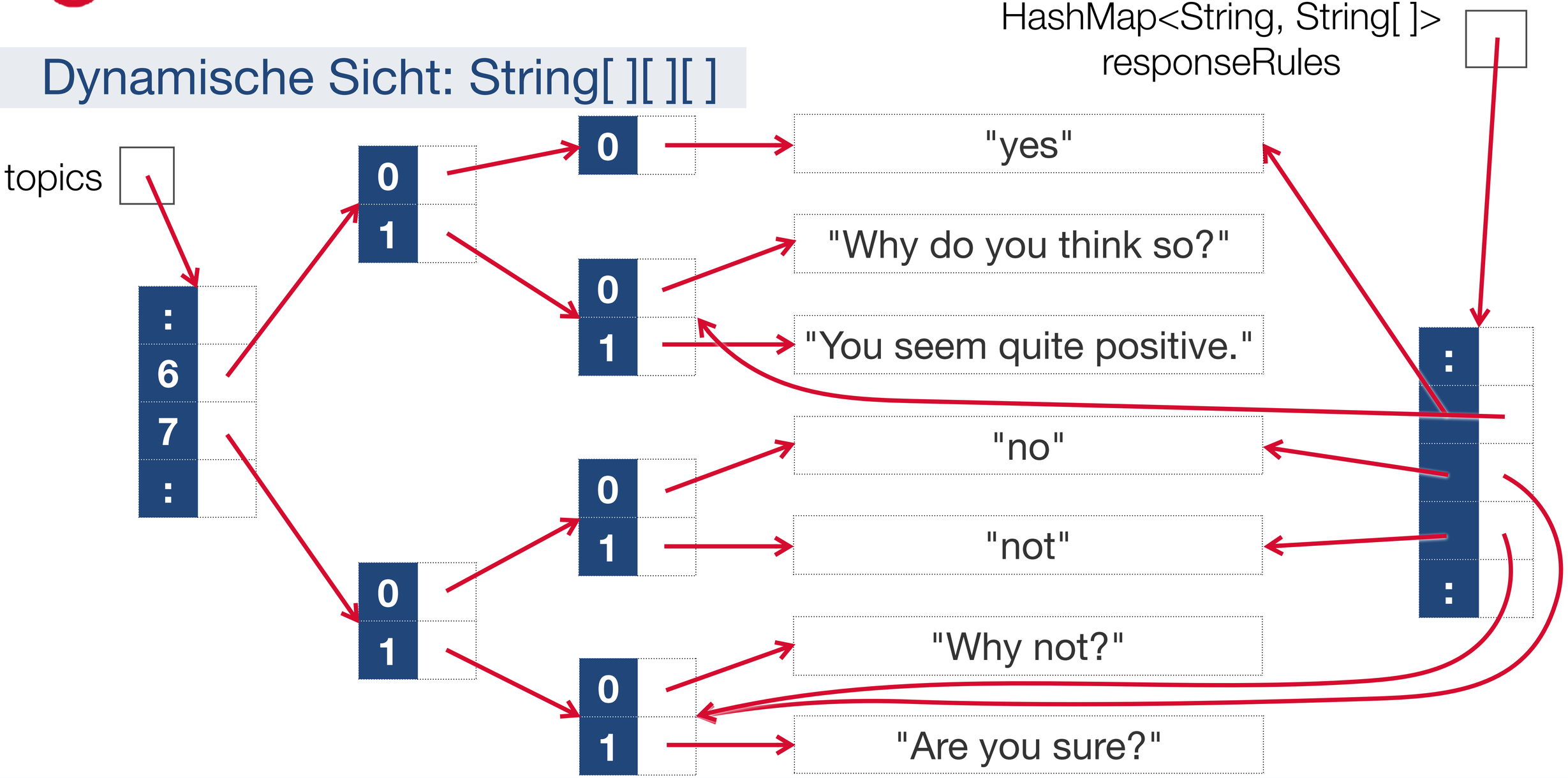
```
private final Map<String, String>
        rules = new HashMap<>();
// ...
final String[ ][ ] replacements = {
    {"me", "you"},
    {"I", "you"},
    {"am", "are"},
    {"my", "your"}
};
for (final String[ ] r : replacements) {
    rules.put(r[0], r[1]);
    rules.put(r[1], r[0]);
}
```

```
String s = rules.get("I"); // "you"
```

Dynamische Sicht



Dynamische Sicht: String[ ][ ][ ]



Universität Bremen

Klassenmethoden und Klassenattribute

- **Klassenmethoden** arbeiten unabhängig von einem Objekt
- **Klassenattribute** gibt es nur einmal pro Klasse (oft für Konstanten genutzt)
- Von außerhalb ihrer Klasse können beide durch Voranstellen von **Klassenname.** verwendet werden (Alternative: **import static**)
  - Aufruf über (nicht verwendetes) Objekt geht auch
- Sie werden vereinbart, indem ihre Signatur das Wort **static** enthält

```
Game.playSound("step.wav");
if (key == KeyEvent.VK_RIGHT) {
```

```
class MyMath
{
    static final int DRÖLF = -123;

    static int sqr(final int x)
    {
        return x * x;
    }
}
```

Universität Bremen

Hüllklassen (Wrapper-Klassen)

- Verpacken von Werten primitiver Datentypen in Objekten, damit diese z.B. in Sammlungen gespeichert werden können
- Konstruktion: ***Type Type*.valueOf(*type* t)**
- Wert abfragen: ***type type*Value()**
- Klassenmethoden
  - ***type Type*.parse*Type*(String s)**
  - **String *Type*.toString(*type* t)**

Datentyp	Hüllklasse
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

```
Integer i = Integer.valueOf(17);
int j = i.intValue();
String s = Integer.toString(4);
int k = Integer.parseInt(s);
```

Universität Bremen

Hüllklassen (Wrapper-Klassen)

- Zahlentypen
  - ***Type*.MAX_VALUE**
  - ***Type*.MIN_VALUE**
- **Float** und **Double**
  - Achtung: **MIN_VALUE** ist hier kleinste positive Zahl
  - ***Type*.NEGATIVE_INFINITY**
  - ***Type*.POSITIVE_INFINITY**
  - ***Type*.NaN**

```
int minimum(final int[ ] array)
{
    int min = Integer.MAX_VALUE;
    for (final int value : array) {
        if (value < min) {
            min = value;
        }
    }
    return min;
}
```

```
double d1 = -1.0 / 0.0; // Double.NEGATIVE_INFINITY
double d2 = 1.0 / 0.0; // Double.POSITIVE_INFINITY
double d3 = 0.0 / 0.0; // Double.NaN
```

Autoboxing

```
ArrayList<Integer> a = new ArrayList<Integer>();
a.add(17); // entspricht a.add(Integer.valueOf(17));
```

- Java wandelt automatisch einen Wert eines primitiven Datentyps in ein Objekt der entsprechenden Hüllklasse um
- Dadurch lassen sich Werte von Basisdatentypen wie Objekte behandeln, z.B. in Sammlungen einfügen
- Funktioniert nur in Ausdrücken, Zuweisungen und Parameterübergaben
- Achtung: **==** vergleicht Referenzen von geboxten Objekten, nicht deren Inhalt (**equals** tut das)
  - Geht für kleine Zahlen teilweise aber doch (Fallstrick!)

```
5.toString(); // Geht nicht!
```

```
Integer a = 127, b = 127;
// a == b ist true
Integer c = 128, d = 128;
// c == d ist false!
```

Universität Bremen

Autounboxing

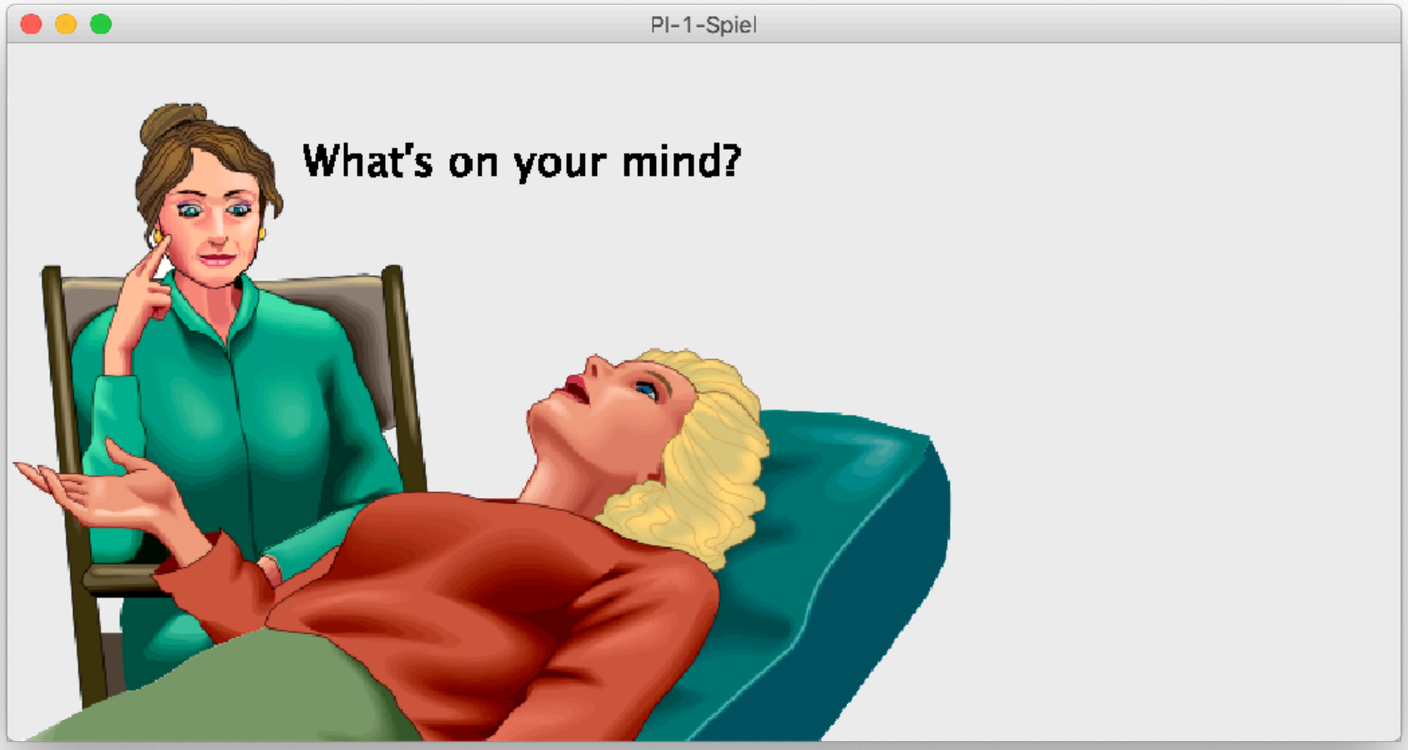
- Java wandelt automatisch ein Objekt einer Hüllklasse in einen Wert des entsprechenden primitiven Datentyps um

```
final Integer[ ] a = {1, 2, 3, 4, 5};
Integer sum = 0;
Integer i = 0;
while (i < a.length) {
    sum = sum + a[i];
    i = i + 1;
}
```

```
final Integer[ ] a = {
    Integer.valueOf(1), Integer.valueOf(2),
    Integer.valueOf(3), Integer.valueOf(4),
    Integer.valueOf(5)
};
Integer sum = Integer.valueOf(0);
Integer i = Integer.valueOf(0);
while (i.intValue() < a.length) {
    sum = Integer.valueOf(sum.intValue()
            + a[i.intValue()].intValue());
    i = Integer.valueOf(i.intValue() + 1);
}
```

Universität Bremen

Eigenständige Programme: Demo



Universität Bremen

Eigenständige Programme

- Enthält eine Klasse eine Funktion mit der Signatur **public static void main(final String[ ] args)**, so kann diese von der Konsole aus aufgerufen werden
  - Virtuelle Maschine **java** (plus Parameter für die JVM)
  - Name der Klasse, die **main** enthält
  - Eventuelle Parameter werden als String-Array an **main** übergeben

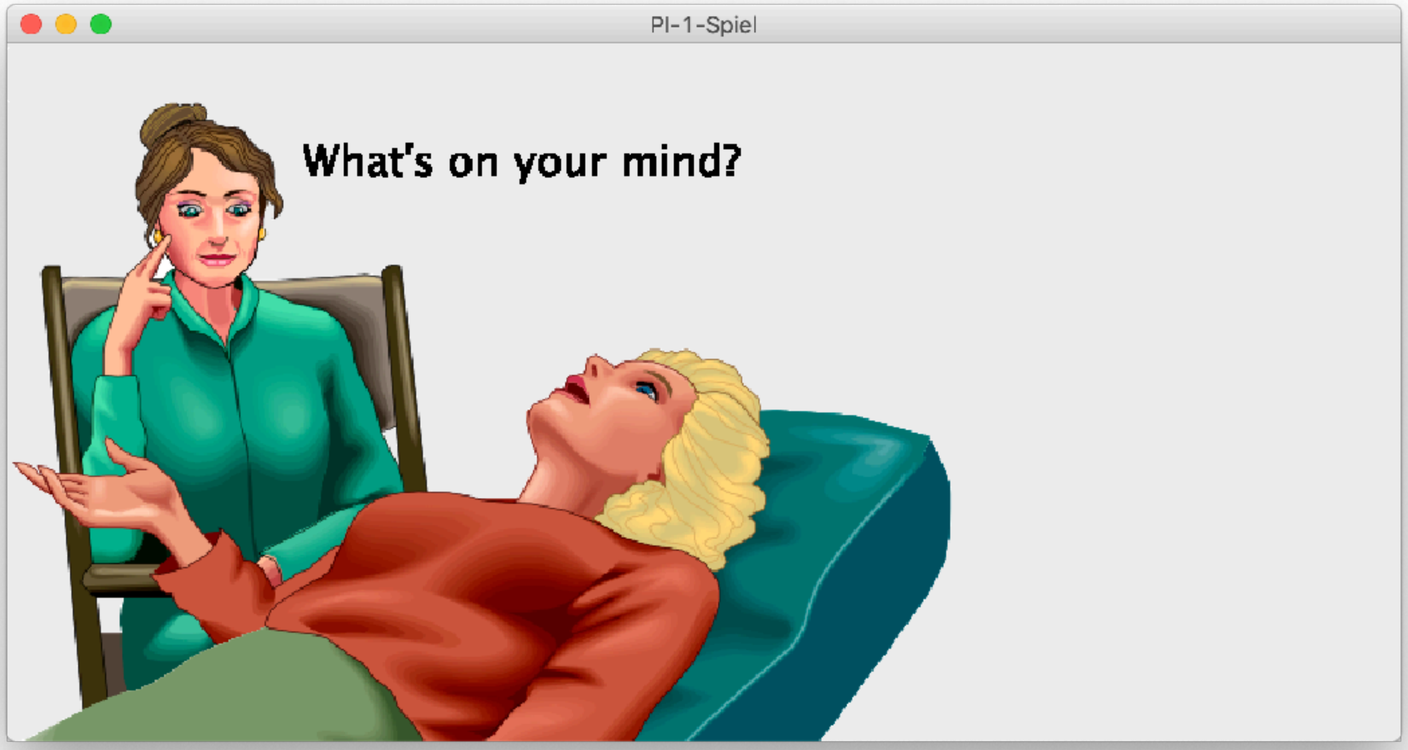
java Klasse "Get Your Kicks on" Route 66

length	3
0	"Get Your Kicks on"
1	"Route"
2	"66"

- Achtung: Wenn ihr nur BlueJ installiert habt, gibt es kein System-weites Java und die Ausführung aus der Konsole funktioniert nicht!

Universität Bremen

Spiel als jar-Archiv exportieren: Demo



Universität Bremen

Java-Archiv (.jar)

- Enthält gleich mehrere Java-Klassen, z.B. alle Klassen eines Programms
  - Ist eigentlich ein zip-Archiv
  - In BlueJ: **Projekt|Als jar-Archiv speichern…**
- Starten
  - **java -cp Archiv.jar Klasse Parameter**
  - Mit Manifest-Datei (**META-INF/MANIFEST.MF**):
    java -jar Archiv.jar Parameter
    - BlueJ legt automatisch eine an, wenn Hauptklasse ausgewählt wird

```
Manifest-Version: 1.0
Main-Class: Office
```

Universität Bremen

Zusammenfassung der Konzepte

- **List**, **Set**, **Map**
- **String**
- **HashMap**
- **Klassenmethoden** und **Klassenattribute**
- **Hüllklassen** und **Auto(un)boxing**
- **Eigenständiges Programm**