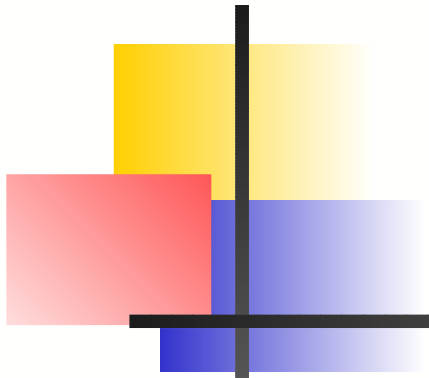


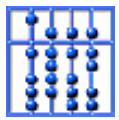
Grafische Compilerschnittstellen

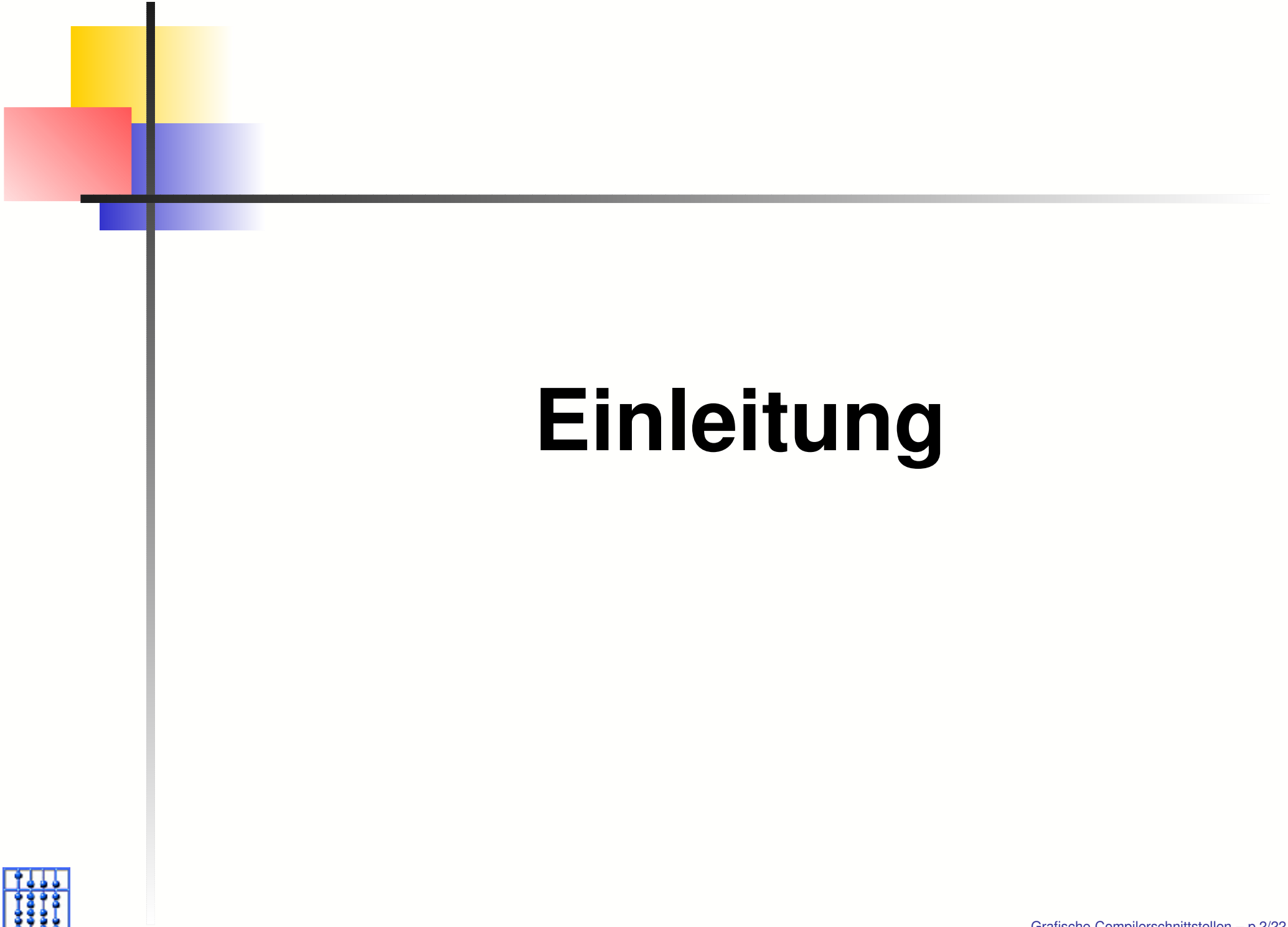
Praktikum des Übersetzerbaus

Michael Petter

petter@in.tum.de

TU-München



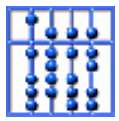


Einleitung

Motivation

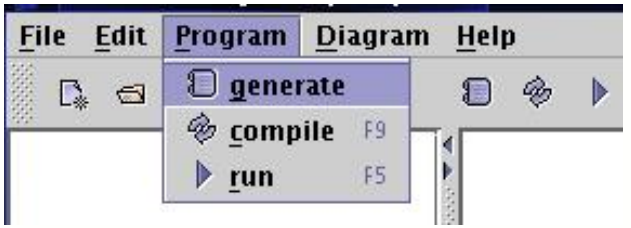
Die Verwendung eines CUP-JFlex-Parsers gestaltet sich mitunter ziemlich mühsam!

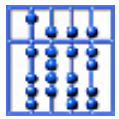
```
java -jar emugen.jar -f -fo
   emufile.emu
javac -cp .:emu_runtime.jar
   Adressbuch.java
java -cp .:emu_runtime.jar
   AdressbuchPanel.java
```

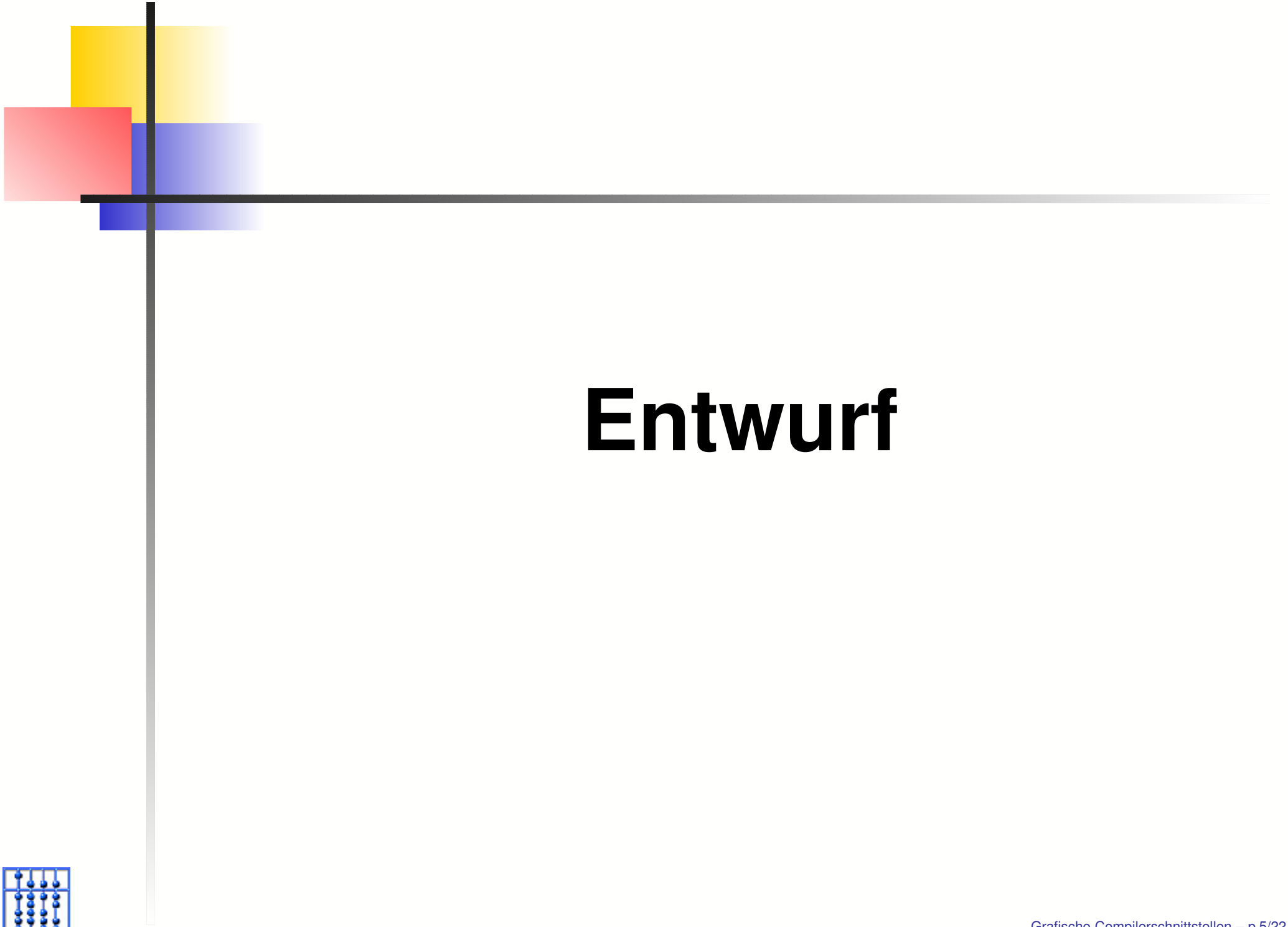


Motivation

Wünschenswert für benutzerfreundliche Programme sind aber unkomplizierte, selbsterklärende grafische Oberflächen:





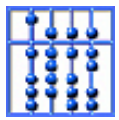


Entwurf

Vorgehensweise

Um die Resultatklassen anzuzeigen müssen mehrere Schritte durchlaufen werden:

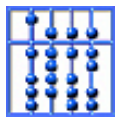
- Schaffung eines temporären Verzeichnisses



Vorgehensweise

Um die Resultatklassen anzuzeigen müssen mehrere Schritte durchlaufen werden:

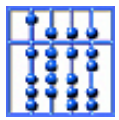
- Schaffung eines temporären Verzeichnisses
- Parseraufruf mit Ausgabe in das temporäre Verzeichnis



Vorgehensweise

Um die Resultatklassen anzuzeigen müssen mehrere Schritte durchlaufen werden:

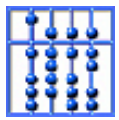
- Schaffung eines temporären Verzeichnisses
- Parseraufruf mit Ausgabe in das temporäre Verzeichnis
- Übersetzung der *.java*-Dateien mit *javac*



Vorgehensweise

Um die Resultatklassen anzuzeigen müssen mehrere Schritte durchlaufen werden:

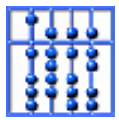
- Schaffung eines temporären Verzeichnisses
- Parseraufruf mit Ausgabe in das temporäre Verzeichnis
- Übersetzung der *.java*-Dateien mit *javac*
- Aufruf der gerade erzeugten Klassen



Vorgehensweise

Um die Resultatklassen anzuzeigen müssen mehrere Schritte durchlaufen werden:

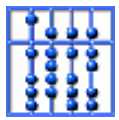
- Schaffung eines temporären Verzeichnisses
- Parseraufruf mit Ausgabe in das temporäre Verzeichnis
- Übersetzung der *.java*-Dateien mit *javac*
- Aufruf der gerade erzeugten Klassen
- Aufräumen im temporären Verzeichnis



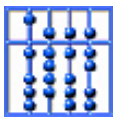
Notwendige Technik

Um diese Aufgaben plattformunabhängig und komfortabel lösen zu können werden folgende Techniken benötigt:

- Management temporärer Dateien
- Einführung in Reflections
- Reflections für *javac*
- Reflections für “Plugins”



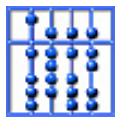
Exkurs: Temporäre Dateien

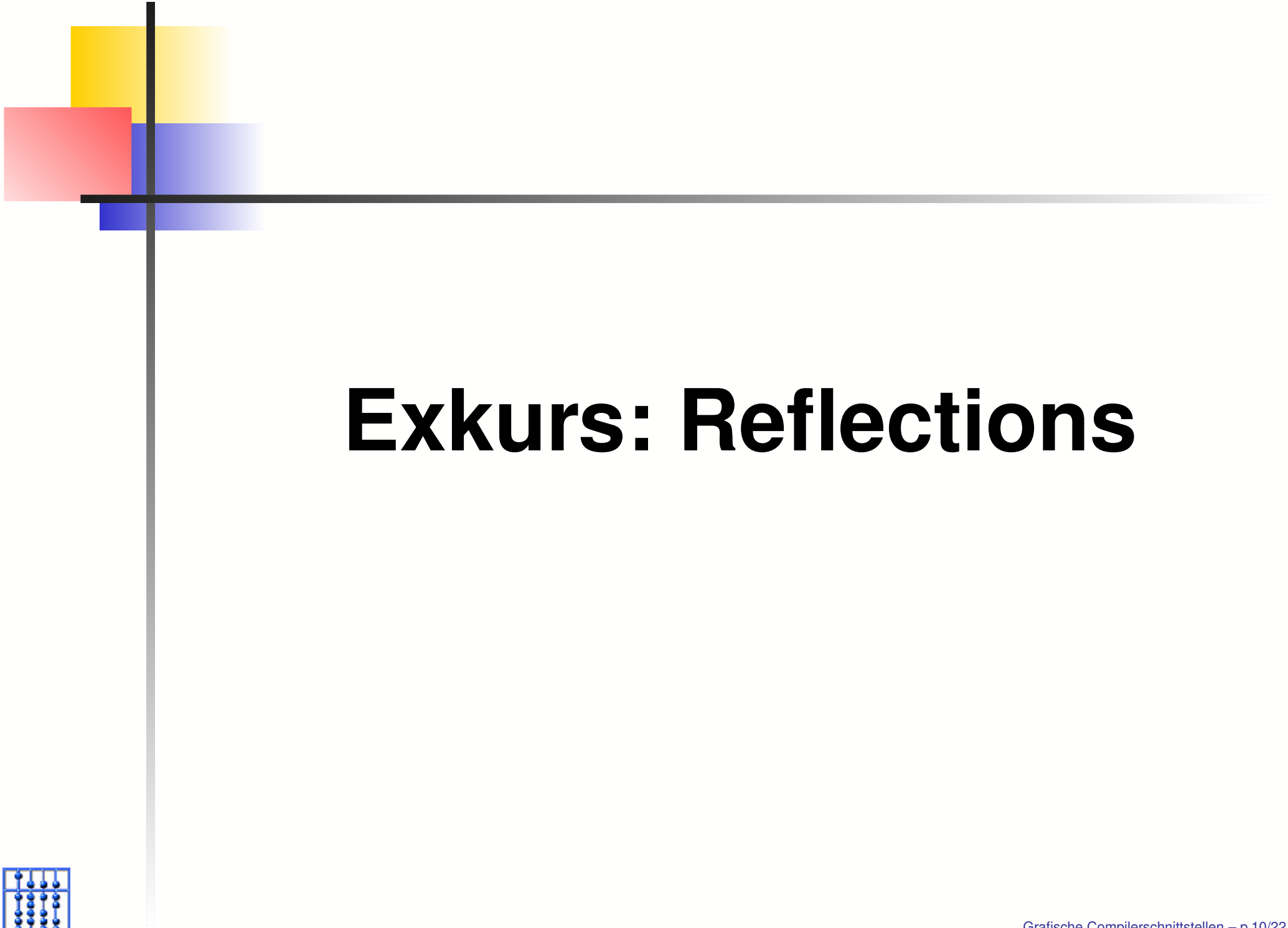


Temporäre Dateien

Temporäre Dateien werden in verschiedenen Betriebssystemen an verschiedenen Orten aufbewart; um trotzdem in den Genuss von temporärem Speicherplatz zu kommen verwendet man Funktionen der Java-API:

```
File tempDir =
   File.createTempFile("tmp",null);
tempDir.delete;
tempDir.mkdir();
tempDir.deleteOnExit();
```





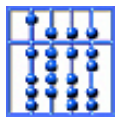
Exkurs: Reflections

Reflections - was ist das?

Von Sun mit Java 1.1 eingeführte Technik; Zitat aus der Sun-API-Dokumentation:

> Reflection allows programmatic access to information about the fields, methods and constructors of loaded classes, and the use [of] reflected fields, methods, and constructors

Zugehörige Klassen und Methoden findet man im Paket *java.lang.reflect*, und unter anderem auch in der Klasse *java.lang.Class*

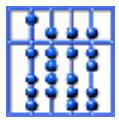


Reflections - wozu?

Beispielcode:

```
Class klasse = System.forName("paket.Klasse");
Method methode = klasse.getMethod("main",
        Class[]{ String[].class });
methode.invoke(klasse.newInstance(),null);
```

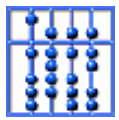
- lädt die Klasse *paket.Klasse*
- lädt die Methode *main* von *klasse*
- führt diese Methode aus



Reflections - beliebige Klassen

Um mit Reflections auch Klassen anzusprechen, die außerhalb des Programm-eigenen Klassenpfades liegen, ist etwas Bastelei nötig - es muss der Class-Loader ausgetauscht werden:

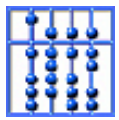
```
URL [] basis = new URL[1];
basis[0] = tempDir.toURL();
URLClassLoader loader =
        new URLClassLoader(basis);
Class klasse =
        loader.loadClass("paket.Klasse");
```



Reflections - der Java Übersetzer

Gute und schlechte Nachrichten:

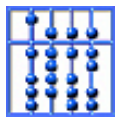
- Praktischerweise wird der SUN-Java-Übersetzer als Java-Programm ausgeliefert. Dadurch kann jedes Java-Programm auch selbst den Java-Compiler verwenden!



Reflections - der Java Übersetzer

Gute und schlechte Nachrichten:

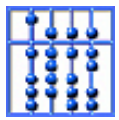
- Praktischerweise wird der SUN-Java-Übersetzer als Java-Programm ausgeliefert. Dadurch kann jedes Java-Programm auch selbst den Java-Compiler verwenden!
- leider liegt der Übersetzer für gewöhnlich nicht im Klassenpfad und wird auch nicht auf allen Computern am selben Ort installiert



Reflections - der Java Übersetzer

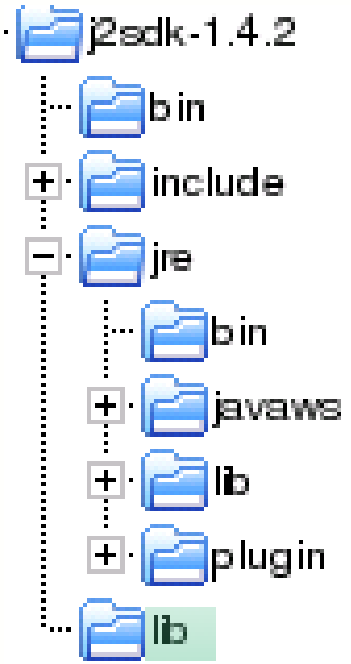
Gute und schlechte Nachrichten:

- Praktischerweise wird der SUN-Java-Übersetzer als Java-Programm ausgeliefert. Dadurch kann jedes Java-Programm auch selbst den Java-Compiler verwenden!
- leider liegt der Übersetzer für gewöhnlich nicht im Klassenpfad und wird auch nicht auf allen Computern am selben Ort installiert
- wir müssen also die Laufzeitumgebung nach dem Übersetzer absuchen!



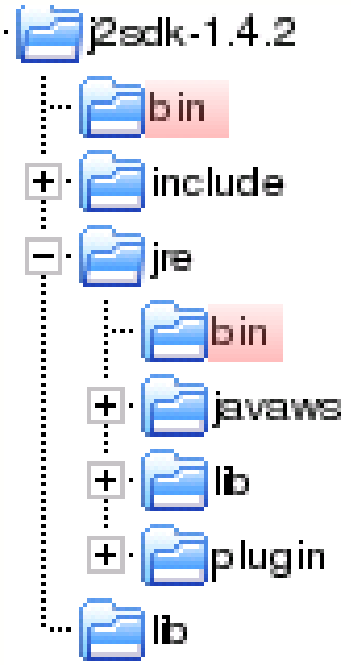
Exkurs: SUN-Java-Compiler

Auf der Suche

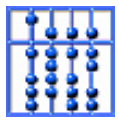


- Der Java-Übersetzer liegt in der Datei *tools.jar*

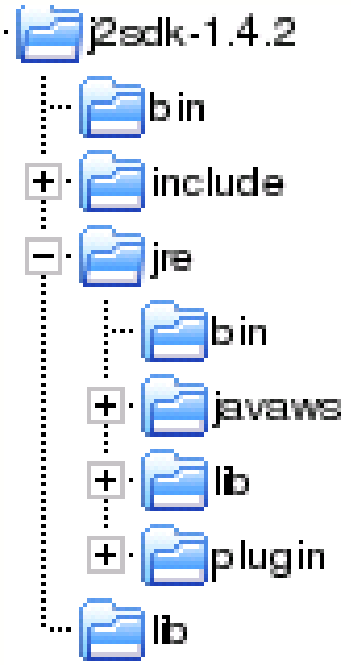
Auf der Suche



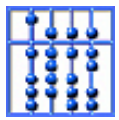
- Der Java-Übersetzer liegt in der Datei *tools.jar*
- In einer J2SDK Distribution gibt es zwei Virtual Machines



Auf der Suche



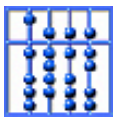
- Der Java-Übersetzer liegt in der Datei *tools.jar*
- In einer J2SDK Distribution gibt es zwei Virtual Machines
- Beide Möglichkeiten müssen berücksichtigt werden



Wer suchet...

Zur Suche bietet sich also an:

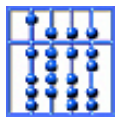
```
String sep =
   System.getProperty("file.separator");
String home =
   System.getProperty("java.home");
File var1 = new File(
   home+sep+"lib"+sep+"tools.jar");
File var2 = new File(
   home+sep+".."+sep+"lib"+sep+"tools.jar");
```

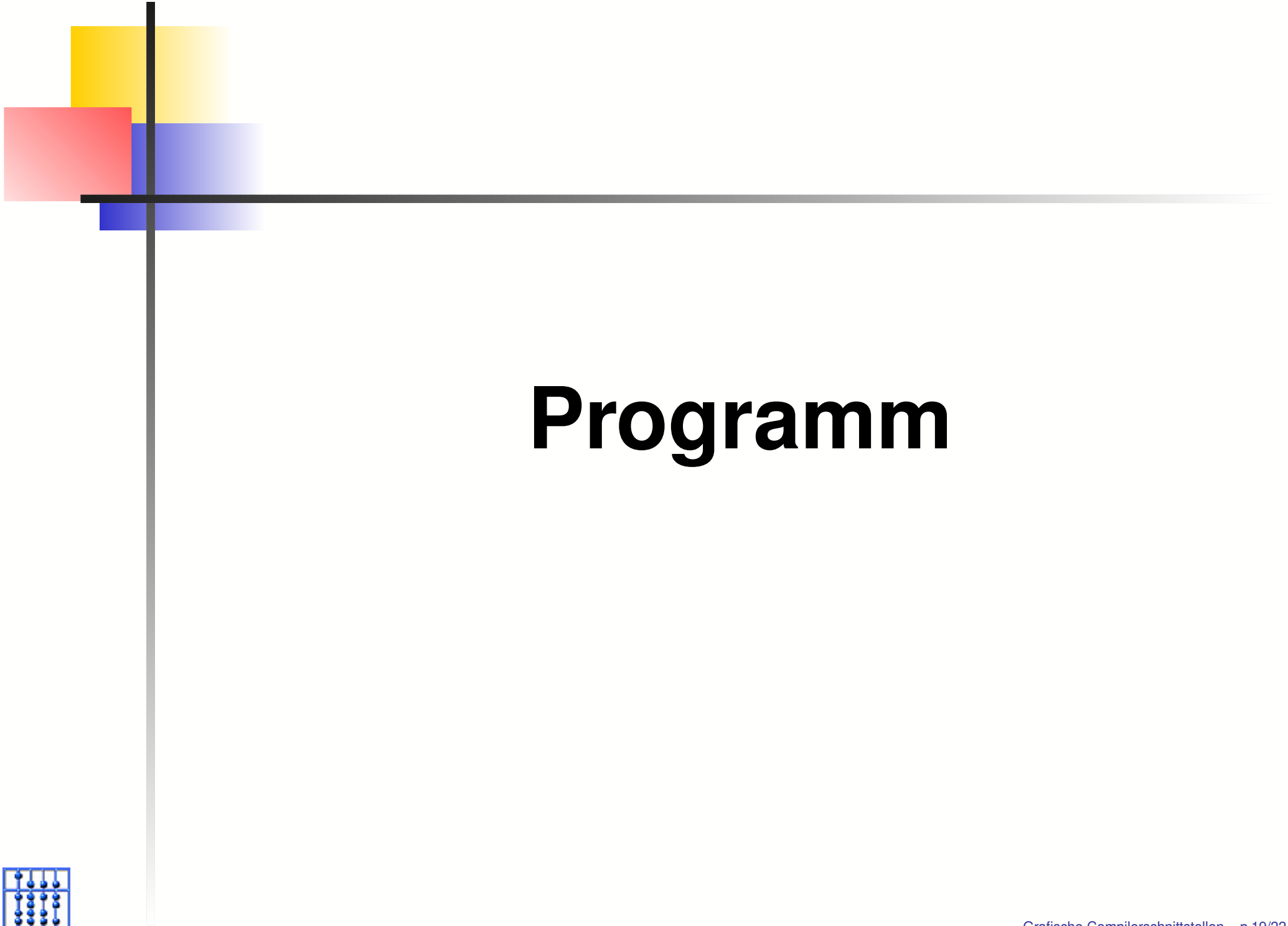


Des Rätsels Lösung

Was nun noch unbekannt ist, ist die Signatur der Methode für den SUN-Java-Compiler. Offizielle Dokumentation über diese ist leider nicht zu bekommen; daher muss man sich auf dekompilierte Klassen verlassen :-)
Nach unserem Wissensstand würde man *javac* so aufrufen:

```
String [] params =
    new String[] {"myClass.java"};
com.sun.tools.javac.Main.compile(params);
```

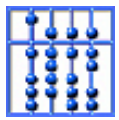




Programm

Programmablauf

1. Temporäres Verzeichnis erzeugen
2. CUP/JFlex Übersetzer aufrufen
3. Sun-Java-Compiler laden
4. Java-Übersetzer aufrufen
5. erzeugte Hauptklasse laden
6. Hauptklasse aufrufen
7. Aufräumen



Anhang

Quellen und Weiterführendes

- SUN: The Reflection API
- SUN Java API Documentation
- Michael Petter: SEP VisualEmugen