EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Running External Programs from Java with `Runtime.exec()`

Programming Course: Computational Linguistics I – Verena Henrich

The Need to Run External Programs from Java

- Java is designed to be platform independent, i.e., independent from any underlying operating system and platform-dependent programs.
- But for some applications it might be necessary to execute/ access existing non-Java, platform-dependent programs.
- For example, if we want to create a linguistically-annotated corpus using TreeTagger for lemmatization and part-of-speech tagging, we need to run TreeTagger on the command line before we can parse the output file with Java.
- It would be nice to be able to use TreeTagger directly in/from Java.

Run External Programs from Java with Runtime.exec

- Class **`Runtime`** allows to access the runtime environment in which your Java application is running:

```
Runtime runtime = Runtime.getRuntime();
```

- Several **`exec`** methods in class **`Runtime`** allow you to run platform-dependent programs (e.g. command line programs such as **`ls`**, **`grep`**, **`uniq`**, etc.) from your Java application:

```
Process p = runtime.exec("ls");
```

- This command executes the specified string command (**`"ls"`**) in a separate process.

Run External Programs from Java with Runtime.exec

- Remember: A process is an environment where a program is executed.
- **`Runtime.exec`** returns an object of type **`Process`** that can be used to control the process and obtain information about it.
- The class **`Process`** provides methods for getting input from the process, performing output to the process, destroying (killing) the process, etc.
- For example, use methods **`getInputStream()`**, **`getOutputStream()`**, and **`getErrorStream()`** to get the results from the program that you executed.

Run TreeTagger from Java with Runtime.exec

```
Runtime runtime = Runtime.getRuntime();

Process process = runtime.exec("/afs/sfs.uni-tuebingen.de/
lehre/culy/TreeTagger/cmd/tree-tagger-german-utf8 input.txt");

InputStream inStream = process.getInputStream();
InputStreamReader sReader = new InputStreamReader(inStream);
BufferedReader bufferedReader = new BufferedReader(sReader);

String line;
while ((line = bufferedReader.readLine()) != null) {
    String word = line.split("\t")[0];
    String pos = line.split("\t")[1];
    String lemma = line.split("\t")[2];
    System.out.println(word + "#" + pos + "#" + lemma);
}
```

Problems with Runtime.exec

- Starting an operating system process is highly system-dependent.
- This means that executing **`Runtime.exec`** (usually) makes your program platform-dependent.
- Among the many things that can go wrong are:
  - The operating system program file was not found.
  - Access to the program file was denied.
  - The working directory does not exist.
- In such cases an **`IOException`** exception will be thrown.

Two Possibilities to Use TreeTagger within Java

1.) with **`Runtime.exec`**

- makes your program platform-dependent
- is easier to use for now although you have to parse the output by yourself

2.) with a Java Wrapper (cf. http://code.google.com/p/tt4j/):

- this wrapper is more difficult to understand
- it was written with a focus on platform-independence (but it is still not out-of-the-box platform-independent)
- more adequate integration in a Java program

→ once you know how to use the wrapper, it is definitely the preferred way!

- Remember that TreeTagger is not a Java program and needs to be compiled separately for each operating system (i.e., it is platform-dependent).