

# Advanced JAPE

## Module 1

June 2019

© 2019 The University of Sheffield

This material is licenced under the Creative Commons

Attribution-NonCommercial-ShareAlike Licence

(<http://creativecommons.org/licenses/by-nc-sa/3.0/>)

# Outline

- 1 Debugging JAPE Grammars
- 2 Using Java in JAPE
  - A Very Brief Introduction to the GATE API
  - How to Include Java in JAPE
  - Common idioms

# Outline

- 1 Debugging JAPE Grammars
- 2 Using Java in JAPE
  - A Very Brief Introduction to the GATE API
  - How to Include Java in JAPE
  - Common idioms

## Debugging JAPE Grammars

- Read the error messages, they are helpful!
  - line numbers etc. refer to the original JAPE files
  - description usually highlights the exact problem

```
file:/home/gate/plugins/ANNIE/resources/NE/name.jape:  
Encountered " <kleeneOp> "? "" at line 1580, column 10.  
Was expecting one of:  
    "\" " ...  
    <ident> ...  
    "|" ...  
    "{" ...  
    "(" ...  
    ")" "
```

## Debugging JAPE Grammars

When trying to understand how annotations were created by a grammar try the new **enableDebugging** option:

- **addedByPR:** the name of the JAPE PR running the grammar that produced the annotation
- **addedByPhase:** the name of the phase (usually the filename) in which the annotation was created
- **addedByRule:** the name of the rule responsible for creating the annotation

# Outline

## 1 Debugging JAPE Grammars

## 2 Using Java in JAPE

- A Very Brief Introduction to the GATE API
- How to Include Java in JAPE
- Common idioms

## Beyond Simple Actions

It's often useful to do more complex operations on the RHS than simply adding annotations, e.g.

- Set a new feature on one of the matched annotations
- Delete annotations from the input
- More complex feature value mappings, e.g. concatenate several LHS features to make one RHS one.
- Collect statistics, e.g. count the number of matched annotations and store the count as a document feature.

JAPE has no special syntax for these operations, but allows blocks of arbitrary Java code on the RHS.

- Don't worry if you are not a (Java) developer
- The rest of this section will show you a number of 'recipes' which you can edit slightly for specific tasks
- These ideas can be cut-and-pasted together to perform more complex actions
- If you do want to understand these examples in more detail then the GATE API will be covered in the developer track on Friday



- The examples covered in this session cover common scenarios
  - accessing annotations and features
  - removing annotations
  - accessing document features
  - using the text under an annotation
- There are lots more examples on the GATE wiki
  - `https://gate.ac.uk/wiki/jape-repository/`

## A Very Brief Introduction to the GATE API

- While you don't need to be a developer to use Java in a JAPE rule, you do need to know a bit about the GATE API so you know roughly what things do and which bits you can cut-and-paste.
- The next few slides describe the main ideas of documents, annotation sets, annotations, and features that you've already met in the GUI in terms of the API.

## GATE Feature Maps

### Feature Maps...

- are simply Java Maps, with added support for firing events.
- are used to provide parameter values when creating and configuring CREOLE resources.
- are used to store metadata on many GATE objects.

### All GATE resources are feature bearers

(they implement `gate.util.FeatureBearer`):

```
1 public interface FeatureBearer{  
2     public FeatureMap getFeatures();  
3  
4     public void setFeatures(FeatureMap features);  
5 }
```

## GATE Documents

A GATE Document comprises:

- a DocumentContent object;
- a Default annotation set (which has no name);
- zero or more named annotation sets;

A Document is also a type of Resource, so it also has:

- a name;
- features.

## Main Document API Calls

```
1 // Obtain the document content
2 public DocumentContent getContent();
3 // Get the default annotation set.
4 public AnnotationSet getAnnotations();
5 // Get a named annotation set.
6 public AnnotationSet getAnnotations(String name);
7 // Get the names for the annotation sets.
8 public Set<String> getAnnotationSetNames();
9 // Get all named annotation sets.
10 public Map<String, AnnotationSet>
11     getNamedAnnotationSets();
12 // Convert to GATE stand-off XML
13 public String toXml();
14 // Convert some annotations to inline XML.
15 public String toXml(Set aSourceAnnotationSet,
16     boolean includeFeatures);
```

## Annotation Sets

### GATE Annotation Sets...

- maintain a set of **Node** objects (which are associated with offsets in the document content);
- and a set of annotations (which have a start and an end node).
- implement the **gate.AnnotationSet** interface;
- ... which extends **Set<Annotation>**.
- implement several **get ()** methods for obtaining the included annotations according to various constraints.
- are created, deleted, and managed by the Document they belong to.

**TIP:** always use a Document object to create a new annotation set! Do not use the constructor!

# Main AnnotationSet API Calls

## Nodes

```
1 // Get the node with the smallest offset.
2 public Node firstNode();
3 // Get the node with the largest offset.
4 public Node lastNode();
```

## Creating new Annotations

```
1 // Create (and add) a new annotation
2 public Integer add(Long start, Long end,
3     String type, FeatureMap features);
4 // Create (and add) a new annotation
5 public Integer add(Node start, Node end,
6     String type, FeatureMap features)
```

## AnnotationSet API (continued)

### Getting Annotations by ID, or type

```
1 // Get annotation by ID
2 public Annotation get(Integer id);
3 // Get all annotations of one type
4 public AnnotationSet get(String type)
5 // Get all annotation types present
6 public Set<String> getAllTypes()
7 // Get all annotations of specified types
8 public AnnotationSet get(Set<String> types)
```

## AnnotationSet API (continued)

### Getting Annotations by position

```
1 // Get all annotations starting at a given
2 // location, or right after.
3 public AnnotationSet get(Long offset)
4 // Get all annotations that overlap an interval
5 public AnnotationSet get(Long startOffset,
6     Long endOffset)
7 // Get all annotations within an interval.
8 public AnnotationSet getContained(Long startOffset,
9     Long endOffset)
10 // Get all annotations covering an interval.
11 public AnnotationSet getCovering(String neededType,
12     Long startOffset, Long endOffset)
```

## AnnotationSet API (continued)

### Combined get methods

```
1 // Get by type and feature constraints.
2 public AnnotationSet get(String type,
3     FeatureMap constraints)
4 // Get by type, constraints and start position.
5 public AnnotationSet get(String type,
6     FeatureMap constraints, Long offset)
7 // Get by type, and interval overlap.
8 public AnnotationSet get(String type,
9     Long startOffset, Long endOffset)
10 // Get by type and feature presence
11 public AnnotationSet get(String type,
12     Set featureNames)
```

# Annotations

## GATE Annotations...

- are metadata associated with a document segment;
- have a type (**String**);
- have a start and an end Node (**gate.Node**);
- have features;
- are created, deleted and managed by annotation sets.

**TIP:** always use an annotation set to create a new annotation! Do not use the constructor.

# Annotation API

## Main Annotation methods:

```
1 public String getType();  
2 public Node getStartNode();  
3 public Node getEndNode();  
4 public FeatureMap getFeatures();
```

## gate.Node

```
1 public Long getOffset();
```

## JAPE With Java RHS Template

```
1 Imports: { import static gate.Utils.*; }
2
3 Phase: Example
4 Input: Token // and any other input annotation types
5 Options: control = appelt
6
7 Rule: Example1
8 (
9   // Normal JAPE LHS goes here
10 ):label
11 -->
12 {
13   // Java code goes in here
14 }
```

## Java Block Variables

The variables available to Java RHS blocks are:

- doc** The document currently being processed.
- inputAS** The `AnnotationSet` specified by the `inputASName` runtime parameter to the JAPE transducer PR. Read or delete annotations from here.
- outputAS** The `AnnotationSet` specified by the `outputASName` runtime parameter to the JAPE transducer PR. Create new annotations in here.
- ontology** The ontology (if any) provided as a runtime parameter to the JAPE transducer PR.
- bindings** The bindings map. . .

## Bindings

- `bindings` is a Map from string to `AnnotationSet`
- Keys are labels from the LHS.
- Values are the annotations matched by the label.

```
1 (
2   {Token.string == "University"}
3   {Token.string == "of"}
4   ({Lookup.minorType == city}):uniTown
5 ):orgName
```

- `bindings.get("uniTown")` contains one annotation (the `Lookup`)
- `bindings.get("orgName")` contains three annotations (two `Tokens` plus the `Lookup`)

## A Simple Example

This is a simple example of a Java RHS that prints the type and features of each annotation it matches. Give it a try!

```
1 Rule: ListEntities
2 ({Person}|{Organization}|{Location}):ent
3 -->
4 {
5     // get the annotations that matched
6     AnnotationSet ents = bindings.get("ent");
7
8     for(Annotation e : ents) {
9         // display the type and features of each
10        System.out.println("Type: " + e.getType());
11        System.out.println("Features: " + e.getFeatures());
12    }
13 }
```

- Application: hands-on/jape/example1.xgapp
- JAPE file: hands-on/jape/resources/simple.iabe

## Named Java Blocks

```
1 -->
2 :uniTown{
3     uniTownAnnots.iterator().next().getFeatures()
4     .put("hasUniversity", Boolean.TRUE);
5 }
```

- You can label a Java block with a label from the LHS
- The block will only be called if there is at least one annotation bound to the label
- Within the Java block there is a variable `labelAnnots` referring to the `AnnotationSet` bound to the label
  - i.e. `AnnotationSet xyAnnots = bindings.get("xy")`
- you can have any number of `:bind.Type = {}` assignment expressions and blocks of Java code, separated by commas.

## Common Idioms for Java RHS

### Setting a new feature on one of the matched annotations

```
1 Rule: LcString
2 ({Token}):tok
3 -->
4 :tok {
5     for (Annotation a : tokAnnots) {
6         // get the FeatureMap for the annotation
7         FeatureMap fm = a.getFeatures();
8         // get the "string" feature
9         String str = (String)fm.get("string");
10        // convert it to lower case and store
11        fm.put("lcString", str.toLowerCase());
12    }
13 }
```

## Exercise 2: Modifying Existing Annotations

- Load `hands-on/jape/exercise2.xgapp`
- As before, this is ANNIE plus an extra transducer, this time loading  
`hands-on/jape/resources/general-pos.jape`.
- Modify the Java RHS block to add a `generalCategory` feature to the matched `Token` annotation holding the first two characters of the POS tag (the `category` feature).
  - `String.substring(startIndex, endIndex)`
- Remember to reinitialize the “Exercise 2 Transducer” after editing the JAPE file.
- Test it by running the “Exercise 2” application.

## Common Idioms for Java RHS

### Removing matched annotations from the input

```
1 Rule: Location
2 ({Lookup.majorType = "location"}):loc
3 -->
4 :loc.Location = { kind = :loc.Lookup.minorType,
5     rule = "Location"},
6 :loc {
7     inputAS.removeAll(locAnnots);
8 }
```

This can be useful to stop later phases matching the same annotations again.

## Common Idioms for Java RHS

### Accessing the string covered by a match

```
1 Rule: Location
2 ({Lookup.majorType = "location"}):loc
3 -->
4 :loc {
5     String str = stringFor(doc,locAnnots);
6 }
```

## Example: Contained Annotations

To get annotations contained within the span of the match

```
1 Rule: NPtokens
2 ({NounPhrase}):np
3 -->
4 :np {
5     List<String> posTags = new ArrayList<String>();
6     for(Annotation tok :
7         getContainedAnnotations(inputAS,
8                                 npAnnots, "Token")) {
9         posTags.add(
10            (String)tok.getFeatures().get("category"));
11     }
12     FeatureMap fm =
13         npAnnots.iterator().next().getFeatures();
14     fm.put("posTags", posTags);
15     fm.put("numTokens", (long)posTags.size());
16 }
```

## Exercise 3: Working with Contained Annotations

- Load `hands-on/jape/exercise3.xgapp`
- As before, this is ANNIE plus an extra transducer, this time loading `hands-on/jape/resources/exercise3-main.jape`.
- This is a multiphase grammar containing the `general-pos.jape` from exercise 2 plus `num-nouns.jape`.
- Modify the Java RHS block in `num-nouns.jape` to count the number of nouns in the matched `Sentence` and add this count as a feature on the sentence annotation.
- Remember to reinitialize the “Exercise 3 Transducer” after editing the JAPE file.
- Test it by running the “Exercise 3” application.

## Passing state between rules

To pass state between rules, use document features:

```
1 Rule: Section
2 ({SectionHeading}):sect
3 -->
4 :sect {
5     doc.getFeatures().put("currentSection",
6         stringFor(doc, sectAnnots));
7 }
8
9 Rule: Entity
10 ({Entity}):ent
11 -->
12 :ent {
13     entAnnots.iterator().next().getFeatures()
14         .put("inSection",
15             doc.getFeatures().get("currentSection"));
16 }
```

## Returning from RHS blocks

- You can **return** from a Java RHS block, which prevents any later blocks or assignments for that rule from running, e.g.

```
1 -->
2 :uniTown{
3   String townString = stringFor(doc, uniTownAnnots);
4   // don't add an annotation if this town has been seen before. If we
5   // return, the UniversityTown annotation will not be created.
6   if (!( (Set) doc.getFeatures().get("knownTowns") )
7         .add(townString)) return;
8 },
9 :uniTown.UniversityTown = {}
```

## Annotation Sets and Ordering

- An AnnotationSet is a set, so it is not ordered

```
10 Rule: SimpleNPRule1
11 (
12   {{Token.generalCategory=="DT"}}?
13   {{Token.generalCategory=="JJ"}} [0, 4]
14   {{Token.generalCategory=="NN"}}+
15 ):nnp
16 -->
17 :nnp {
18   System.out.println("_____");
19   System.out.println(stringFor(doc, nnpAnnots));
20   System.out.println("The individual tokens:");
21
22   for(Annotation tok : nnpAnnots) {
23     System.out.println(stringFor(doc, tok));
24   }
25 }
```

- The grammar for this example is in `hands-on/jape/resources/match-nps.jape`. To run the example yourself, load `exercise2.xgapp` in GATE Developer, load an extra JAPE Transducer PR, and give it as a parameter this grammar file. Finally, add the resulting new PR at the end of the Exercise 2 application and re-run it.

## Annotation Sets and Ordering (Continued)

- Here is a sample output, if you execute this rule on our test document

---

```
waste management businesses
Now printing the matched individual tokens:
businesses
waste
management
```

---

- Instead use `inDocumentOrder(AnnotationSet as)` which returns a list containing the annotations in the given annotation set, in document order

# Exceptions

- Any `JapeException` or `RuntimeException` thrown by a Java RHS block will cause the JAPE Transducer PR to fail with an `ExecutionException`
- For non-fatal errors in a RHS block you can throw a `gate.jape.NonFatalJapeException`
- This will print debugging information (phase name, rule name, file and line number) but will not abort the transducer execution.
  - However it will interrupt this rule, i.e. if there is more than one block or assignment on the RHS, the ones after the **throw** will not run.