Reference and Perspectivation in Java and Kotlin

2nd status update
Agenda

- Demo of recent changes
- No feedback from friendly user yet
- Evaluation ideas
- Next steps
Demo of recent changes
Evaluation ideas: Contributions

When can anaphors be useful for reading / writing code?

- IntelliJ plug-in for Java and Kotlin
- Show that anaphors can speed up reading and writing
- Understand when/which anaphors are easy to understand
- (Adapt presentation of source code based on individual knowledge)
Evaluation ideas

- Video analysis
- Corpus study
- Telemetry
- Experiment
- Questionnaire
- Plug-in downloads
Evaluation ideas: (PP) video analysis

- When can anaphors be useful for reading / writing code?
  - PP videos: Situations when variables are added/changed
- Show that anaphors can speed up reading and writing
- Understand when/which anaphors are easy to understand
  - Videos with plug-in
Evaluation ideas: Corpus study

- When can anaphors be useful for reading / writing code?
  - Percentage of anaphors that can be resolved/perspectivized
- Show that anaphors can speed up reading and writing
  - Amount of redundant information that can be folded away
- Understand when/which anaphors are easy to understand
  - Activation: frequency of concepts/relations in corpus
  - Distance between referring expression and related element
Evaluation ideas: Telemetry

- When can anaphors be useful for reading / writing code?
  - Writing (auto-realization, completion and quick fixes)
    - Used for relevant number/fraction of references
    - Written without interruption of flow
    - No compiler errors introduced
    - Not undone/replaced immediately
  - Reading (perspectivation/folding and highlighting)
    - Not: user highlights anaphor and related element (like regressions in eye tracking)
    - Not: user expands collapsed fold (like regressions in eye tracking)

- Show that anaphors can speed up reading and writing
  - Amount of redundant information that can be folded away

- Understand when/which anaphors are easy to understand
  - Activation: git changelog, clicks and edits of words/concepts/relations
  - Distance between referring expression and related element
Evaluation ideas: Experiment

- When can anaphors be useful for reading / writing code?
  - Only few situations in experiment
- Show that anaphors can speed up reading and writing
  - Control condition without anaphors
- Understand when/which anaphors are easy to understand
  - Only few situations in experiment
Evaluation ideas: Questionnaire

- When can anaphors be useful for reading / writing code?
  - Is plugin useful and why + what users miss
- Show that anaphors can speed up reading and writing
- Understand when/which anaphors are easy to understand
  - More in-situation: examples in GitHub issues and on Slack (examples of difficulties)
Evaluation ideas: Plug-in downloads

- When can anaphors be useful for reading / writing code?
- Show that anaphors can speed up reading and writing
- Understand when/which anaphors are easy to understand
Next steps

- Plugin features
  - Copying names along qualified expressions (Metonymy/Prepositions)
  - Click/Select referent
- Performance analysis/optimization
- Optimize completion
- Additional friendly users
- Corpus analysis
  - What can (not) be resolved?
  - What can (not) be perspectivized?
- Plug-in in JetBrains marketplace (postponed)
- Further features like cataphors, indirect anaphors, deixis
Next steps

- April  
  Next status update
- May 22  
  KotlinConf
- Jul 17  
  WeAreDevelopers World Congress?