Generating class comments in Pharo automatically

Lino Hess
Bachelor Thesis 1st presentation
Supervised by Pooja Rani
17.November 2020
Pharo class comment

I represent a message to be scheduled by the WorldState.

For example, you can see me in action with the following example which print 'alarm test' on Transcript one second after evaluating the code:

Transcript open.
MorphicUIManager currentWorld
   addAlarm: #show:
      withArguments: #('alarm test')
      for: Transcript
      at: (Time millisecondClockValue + 1000).

* Note *
Compared to doing:

the alarm system has several distinctions:
- Runs with the step refresh rate resolution.
- Alarms only run for the active world. (Unless a non-standard scheduler is in use)
- Alarms with the same scheduled time are guaranteed to be executed in the order they were added
Pharo comment template

Class: Test

Please comment me using the following template inspired by Class Responsibility Collaborator (CRC) design:

For the Class part: State a one line summary. For example, "I represent a paragraph of text".

For the Responsibility part: Three sentences about my main responsibilities - what I do, what I know.

For the Collaborators Part: State my main collaborators and one line about how I interact with them.

Public API and Key Messages
- message one
- message two
  - (for bonus points) how to create instances.

One simple example is simply gorgeous.

Internal Representation and Key Implementation Points.

Implementation Points
Pharo comment template

- Class, Responsibility and Collaborators
- API and Key Messages
- Examples, implementations
Why do we want to generate comments?

- Possibility to spend less time on writing comments
- Create a uniform format to prevent inconsistent comments
Goal

Create a commenting tool written in Pharo
What are the challenges in generating comments automatically?

- Which approach to use?
- How do we define the heuristics to use?
Related work in Java

- Moreno et Al.
- Focused mostly on the responsibilities of the classes

Automatic Generation of Natural Language Summaries for Java Classes

Laura Moreno1, Jain Aggarwal2, Geoprdard Siddhara2, Andrea Mutje2, Dori Haldar3, K. Vijay-Shanker2

1Waseda University 2University of California at Santa Barbara 3Institute of Defense Technology
Denton, TX, USA Santa Barbara, CA, USA Bangalore, India

Authors, title, affiliations, and contact information are provided.

Focus on summarization techniques for Java classes, discussing the need for comprehensive summaries and the challenges in generating them.

- Moreno et Al.
- Focused mostly on the responsibilities of the classes

Previous work in Java

- Moreno et Al.
- Focused mostly on the responsibilities of the classes

The need for summarization techniques in Java is highlighted, with a focus on the responsibilities of the classes.

- Moreno et Al.
- Focused mostly on the responsibilities of the classes

Summarization techniques for Java classes are discussed, with an emphasis on responsibilities.

- Moreno et Al.
- Focused mostly on the responsibilities of the classes

The discussion on responsibilities continues, with a focus on summarization techniques.

- Moreno et Al.
- Focused mostly on the responsibilities of the classes

The importance of summarization in Java is further elaborated, with a focus on responsibilities.

- Moreno et Al.
- Focused mostly on the responsibilities of the classes

The overall focus remains on summarization techniques and responsibilities in Java.
Heuristie-based process of Moreno et Al.

Class written in Java

Class stereotypes
1. Entity
2. Minimal entity
3. Data provider
4. Commander
5. Boundary
6. Factory
7. Controller
8. Pure controller
9. Large class
10. Lazy class
11. Degenerate
12. Data class
13. Pool

Method stereotypes
1. Accessors
2. Mutators
3. Creational methods
4. Collaborational methods
5. Degenerate methods

Summary
1. General description
2. Description based on class stereotype.
3. Description of behaviour based on method stereotype.
4. List of inner classes
Our approach

- Heuristic based
- Corresponding to the template format
Related work in Pharo

- Analyzed Class comments
- Found various information types embedded in class comments
- Many comments were written and formatted in a non-uniform way
Heuristic-based process in Pharo

Information types

1. Intent
2. Responsibility
3. Collaborator
4. Public API
5. Example
6. Implementation Points
7. Instance Variables
8. Class references
9. Warnings

Class written in Pharo

Filled in template.
Pipeline

Understanding the project
- End of October

Define heuristics
- Ongoing

Developing the tool
- End of February

Evaluate approach by Pharo developers
- End of February

Comparison with Moreno et Al.
- End of March

Writing paper
- April