Assignment 1: Detecting (Anti-)Patterns

Milan Lagae

Institution/University Name: Vrije Universiteit Brussel
Faculty: Sciences and Bioengineering Sciences
Course: Software Quality Analysis

Contents

1 Intro

The assign pattern for this assignment is the Template Method pattern as described in [1] and [2].

All indivual complete methods can be found in the Appendix section §4.

2 Discussion Point 1

2.1 Abstract & Extending

The first step in the logic query is retrieving all abstract & extending classes. The method named get-abstract-and-extending-classes is responsible for this, the complete method can be found in Listing 15.

The first step is retrieving all ast nodes of :TypeDeclaration and define this as the extending class. Retrieve the type of the declaration and check if the type is a class, Listing 1.

```
1 (ast :TypeDeclaration ?extending)
2 (typedeclaration-type ?extending ?type)
3 (type|class ?type)
```

Listing 1: Extending class.

After the type of the class is retrieved, check that the extending class is not defined as an abstract class, Listing 2.

```
1 ; The extending class cannot both be abstract & have a super type
2 (has :modifiers ?extending ?modifiers-extending)
3 (modifier|abstract ?mod-abstract)
4 (fails (contains ?modifiers-extending ?mod-abstract))
```

Listing 2: Modifiers extending class.

2.2 Checking methods

After the list of abstract and extending classes is generated, continue to the method to refine the results by filtering on methods.

Filtering of the on both types of classes is done by the check-methods method, which can be found in full in Listing 16.

The inner of the method is surrounded with a one & all.

Start by iterating the method declerations in the body using the method: typedeclaration-method, which retrieves all the :bodyDeclerations (=methods) in the body of the class, can be found in Listing 17.

2.2.1 Abstract method

The next step is checking if the retrieve method is an abstract method, the full method can be found in Listing 18.

2.2.1.1 Method Body

The first step is checking what the content of the body of the method is, according to the pattern definition, a abstract method should be empty.

The code can be found in Listing 4, since the actual Java code can be two different examples as illustrated in Listing 3.

```
public abstract methodDefinition();
// or
public abstract methodDefinition() {};
```

Listing 3: Java method examples

For the first type the Ekeko body property will have the null type. The second type requires more code, first check the body is not null, by performing a fail check. Than retrieve the statements property value from the body (the actual expressions). Transform the list to its raw value using value-raw. Lastly check the list of statements is empty (count = 0).

```
; Get method body
(has :body ?abstract-method ?abstract-body)
; Method body is null or has no statements
(conde
  [(value|null ?abstract-body)]
  [
   (fails (value|null ?abstract-body))
   (has :statements ?abstract-body ?stmts)
   (value-raw ?stmts ?raw)
   (equals 0 (count ?raw))])

```

Listing 4: Abstract body must be empty.

2.2.1.2 Method Modifiers

Next check the modifiers on the method, in Listing 5. There are two different types of conditions;

- public & abstract;
- protected.

```
; Modifiers
(has :modifiers ?abstract-method ?modifiers)
; Abstract method is public + abstract or protected
(conde
  [
   (modifier|public ?mod-public)
   (contains ?modifiers ?mod-public)
   (modifier|abstract ?mod-abstract)
   (contains ?modifiers ?mod-abstract)]
  [
   (modifier|protected ?mod-protected)
   (contains ?modifiers ?mod-protected)])

```

Listing 5: Abstract method, required modifiers.

That concludes the code required for checking if a method is an abstract method.

2.2.2 Overriding method

The following step is retrieving the overriding method associated with the abstract method, the auxilary method used for this is named: is-overriding-method and can be fully found in Listing 19.

This information can be retrieved by using the build-in Ekeko method: (methoddeclaration-methoddeclaration|overrides ?abstract-method ?overrider-method).

2.2.2.1 Class

After the method is retrieved, retrieve the class associated with the method, reuse the typedecleration-method defined earlier, Listing 6.

```
; Get the overrider method from the abstract-method
(methoddeclaration-methoddeclaration|overrides ?abstract-method ?overrider-method)
; Check if parent class is the same as the extending class of the abstract
(typedeclaration-method ?extending ?overrider-method)
```

Listing 6: Retrieving overrider method from abstract method.

2.2.2.2 Method Body

Next the body of the method must be null, the code for this can be found in Listing 7. Retrieve the body of the method using the :body property value. After the body is returned, check the body is not null by using a fails.

```
; Overriding method body cannot be empty
(ast :MethodDeclaration ?overrider-method)
(has :body ?overrider-method ?overrider-body)
(fails (value|null ?overrider-body))
```

Listing 7: Checking body of overriding method.

2.2.2.3 Method Modifiers

The final check for the overriding method is checking the modifiers defined on the method, Listing 8. Retrieve the list of modifiers from the method using the :modifiers property value. Get the public modifier value from the modifier|public Ekeko built-in method. After that use contains to check for the presence of the public modifier in the list of modifiers.

```
; Overrider is public
(has :modifiers ?overrider-method ?modifiers)
(modifier|public ?mod-public)
(contains ?modifiers ?mod-public)
```

Listing 8: Checking list of modifiers on the overriding method.

2.2.3 Overrider & Template Method

Following step checks if the overrider-method is present in the template-method of the abstract-class. Surround the query with a one for the ?template-method value.

Retrieve a method decleration from the abstract class, first check if the name is not the same as the abstract-method retrieved earlier. This is done by retrieving the :name property on both respective methods. Proceed to check if comparison of name fails.

If that is the case, check if the method is a template method using is-template-method, the full code of the method can be found in Listing 21.

```
; Check if the overrider method is present
; in the template method of the abstract class
(one (fresh [?template-method]
      (typedeclaration-method ?abstract ?template-method)
      ; Template method does not equal abstract-method
      (has :name ?template-method ?template-name)
      (has :name ?abstract-method ?abstract-name)
      (fails (name|simple-name|simple|same ?template-name ?abstract-name))

      ; Is template method
      (is-template-method ?template-method ?template-body)

      (is-algorithm-step ?template-body ?overrider-method)
    ))
```

Listing 9: Validating template method & overrider method.

2.2.3.1 Template Method

The full is-template-method can be found in Listing 21. The method is responsible for checking if the method declaration is a template-method. A template method has to adhere to the following criteria. It must be a public method and cannot contain any other modifiers. The body of the method is also not allowed to be public.

In code this translates to the following, first checking the list of modifiers on the method as show in Listing 10.

```
; Public and non abstract
(has :modifiers ?template-method ?modifiers)

; Public method
(modifier|public ?mod-public)
(contains ?modifiers ?mod-public)
; Must be public method only
(value-raw ?modifiers ?raw)
(equals 1 (count ?raw))
```

Listing 10: Modifier check on the template method.

Retrieve the :modifiers property from the method using the has query. Retrieving a value for the public modifier using modifier|public to than check using contains if the value is present in the list of modifiers. The length of the list of modifiers is also checked to see if it's length is 1, only containing the public modifier.

After the modifiers are checked, checking if the body of the method is not empty is done as illustrated in Listing 11.

```
1  ; Non empty body
2  (has :body ?template-method ?template-body)
3  (fails (value|null ?template-body))
```

Listing 11: Template mehod non-empty body.

Validating if a method is a template method in the sense is not complete without the next method named: is-algorithm-step.

2.2.4 Algorithm Step

The complete is-algorithm-step method can be found in Listing 20. The method starts of with retrieving the name of the overrider-method element, the :statements property is also retrieved from the template-body, as illustrated in Listing 12.

```
1  ; Overrider method name
2  (has :name ?overrider-method ?overrider-name)
3  ; Extracts rhs method invocation from the body of the template
   method
4  (has :statements ?template-body ?stmts)
```

Listing 12: Method name & body statements.

Each statement in the list of statements is than iterated using contains, for each, the name of the invocation expression is compared with the name of the given overrider-name. The list of queries is surrouned with a one, indicating that at least one match for that particular method must be found in the body of the template method.

```
1  ; We need to at least have one match
2  (one (fresh  [?stmt ?expression ?rhs ?inv-name]
3            ; Iterate each stmt
4            (contains ?stmts ?stmt)
5            ; Extract name
6            (has :expression ?stmt ?expression)
7            (has :rightHandSide ?expression ?rhs)
8            (has :name ?rhs ?inv-name)
9
10           ; 'Compare' with overrider method name
11           (name|simple-name|simple|same ?overrider-name ?inv-
   name)
12            )
```

Listing 13: Comparing method name & invocation name in method body.

This completes the list of methods used for querying the Template Method pattern on the DesignPatterns folder.

2.3 Results

Executing the query as defined above, results in the following result defined in Figure 1.

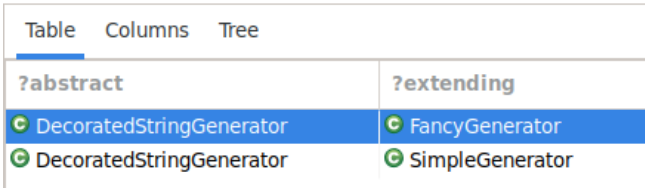

Figure 1: Discussion Point 1 - Result

All the expected results for the Template Method pattern in the DesignPatterns folder are present in the result of the query.

3 Discussion Point 2

When running the resulting query of Discussion Point 1 **exactly**, on the JhotDraw folder, there are no results.

To gather more results, part of the query was removed/ commented out.

3.1 Comparison

First start by removing the comparison query line in the is-algorithm-step method, namely: name|simple-name|simple|same, which can be found in Listing 14.

```
1  ; We need to at least have one match
2  (one (fresh  [?stmt ?expression ?rhs ?inv-name]
3            (...)
4
5            ; 'Compare' with overrider method name
6            (name|simple-name|simple|same ?overrider-name ?inv-
   name))
```

Listing 14: Comparing method name & invocation name in method body.

Executing the remaining query which this line removed results in the following as defined in Figure 2.

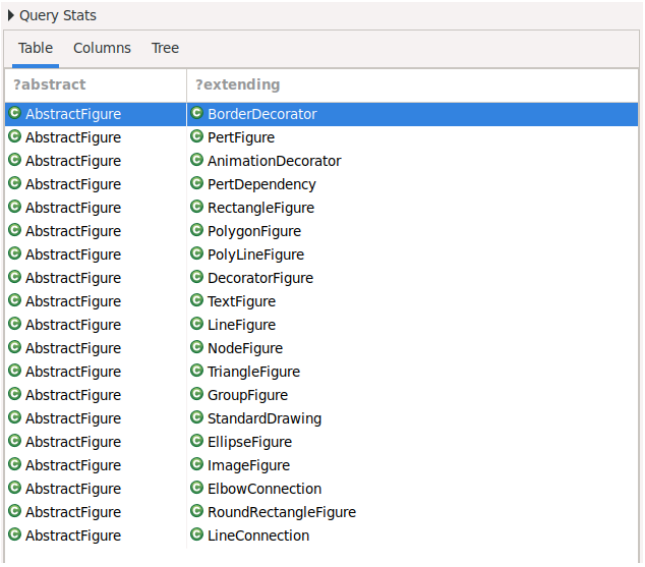

Figure 2: Discussion Point 2 - Comparison Removed Result

The result already includes more of the expected patterns. When comparing with the expected results as defined in the xml file, all results regarding the: AttributeFigure are missing, and for the results regarding AbstractFigure, there is **1** false positive: DecoratorFigure and 3 missing implementations (DiamondFigure, NumberTextFigure, BouncingDrawing).

3.2 Is Algorithm Step

The purpose of the is-algorithm-step method was to check for the template-method in the abstract class of the pattern to check for the overrider methods that are defined as the steps of the algorithm as explained in [2].

When removing the calling of the is-algorithm-step method from the logic query, more results are show, but include more false/positives. The results can be seen in Figure 3 & Figure 4.

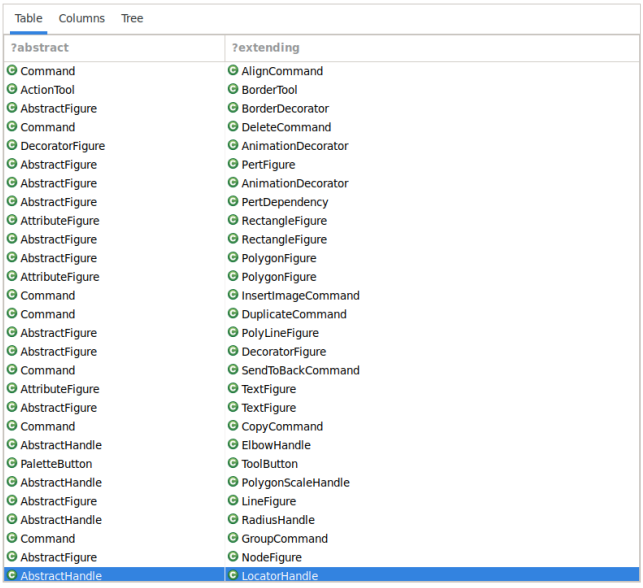

Figure 3: Discussion Point 2 - Result 1

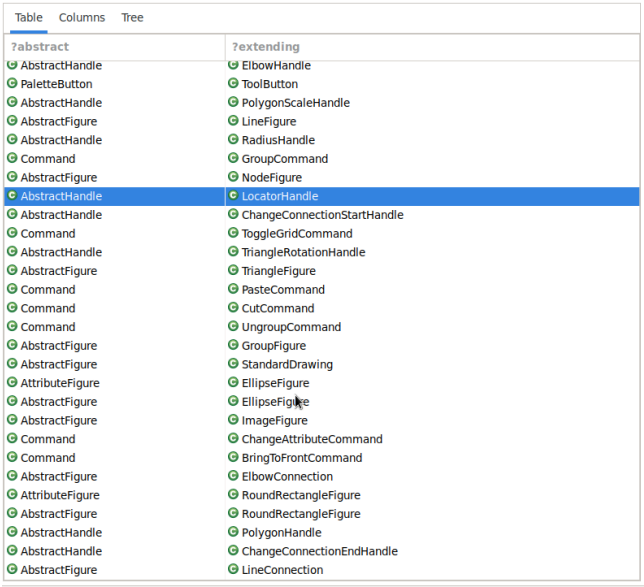

Figure 4: Discussion Point 2 - Result 2

Analysing the results when removing the method, is that more of the expected results are included, but also more false positives are included, mainly: Command, AbstractHandle, PaletteButton as the abstract classes, and their corresponding extending classes.

No further refined of the query was performed for discussion point 2.

4 Appendix

4.1 Discussion Point 1

clj

```clj
; Abstract and extending classess
(defn get-abstract-and-extending-classes [?abstract ?extending]
 (fresh [?type ?supertype ?modifiers-abstract ?mod-abstract ?modifiers-extending]

   ; Intersection of abstract and extending class
   (ast :TypeDeclaration ?extending)
   (typedeclaration-type ?extending ?type)
   (type|class ?type)

   ; The extending class cannot both be abstract & have a super type
   (has :modifiers ?extending ?modifiers-extending)
   (modifier|abstract ?mod-abstract)
   (fails (contains ?modifiers-extending ?mod-abstract))

   ; Get the supertype of the type
   (type-type|super+ ?type ?supertype)

   ; Abstract class
   (type|class ?supertype)
   (typedeclaration-type ?abstract ?supertype)
   (ast :TypeDeclaration ?abstract)

   ; Abstract modifier
   (has :modifiers ?abstract ?modifiers-abstract)
   (contains ?modifiers-abstract ?mod-abstract)))
```

Listing 15: Method that retrieves a list of abstract and extending classes.

```clj
(defn check-methods [?abstract ?extending]
 (fresh [?overrider-body ?overrider-method ?template-method ?abstract-method
       ?template-body ?template-name ?abstract-name]

  (one (all
    ; Gets all the childs nodes of abstract class
    (typedeclaration-method ?abstract ?abstract-method)
    ; Abstract method
    (is-abstract-method ?abstract-method)

    ; Is overrider method?
    (is-overriding-method ?abstract-method ?extending ?overrider-method)

    ; Check if the overrider method is present
    ; in the template method of the abstract class
    (one (fresh [?template-method]
        (typedeclaration-method ?abstract ?template-method)
        ; Template method does not equal abstract-method
        (has :name ?template-method ?template-name)
        (has :name ?abstract-method ?abstract-name)
        (fails (name|simple-name|simple|same ?template-name ?abstract-name))

        ; Is template method
        (is-template-method ?template-method ?template-body)

        (is-algorithm-step ?template-body ?overrider-method)))))))
```

Listing 16: Method responsible for checking method declerations on the pair of abstract and extending class.

```clj
; Method taken from WPO, get all method declerations
(defn typedeclaration-method [?class ?method]
 (child :bodyDeclarations ?class ?method))
```

Listing 17: Retrieves all method declerations from a given class.

```clj
  ; Is abstract method?
(defn is-abstract-method [?abstract-method]
 (fresh [?abstract-body ?modifiers ?mod-abstract ?mod-public ?mod-protected ?stmts ?raw]
   ; Method of the abstract class
   (ast :MethodDeclaration ?abstract-method)

   ; Get method body
   (has :body ?abstract-method ?abstract-body)
   ; Method body is null or has no statements
   (conde
    [(value|null ?abstract-body)]
    [(fails (value|null ?abstract-body))
     (has :statements ?abstract-body ?stmts)
     (value-raw ?stmts ?raw)
     (equals 0 (count ?raw))])


   ; Modifiers
   (has :modifiers ?abstract-method ?modifiers)
   ; Abstract method is public + abstract or protected
   (conde
    [(modifier|public ?mod-public)
     (contains ?modifiers ?mod-public)
     (modifier|abstract ?mod-abstract)
     (contains ?modifiers ?mod-abstract)]
    [(modifier|protected ?mod-protected)
     (contains ?modifiers ?mod-protected)])))
```

Listing 18: Checks if the given method is an abstract method.

```clj
(defn is-overriding-method [?abstract-method ?extending ?overrider-method]
 (fresh [?overrider-body ?abstract-body ?modifiers ?mod-public]
   ; Get the overrider method from the abstract-method
   (methoddeclaration-methoddeclaration|overrides ?abstract-method ?overrider-method)
   ; Check if parent class is the same as the extending class of the abstract
   (typedeclaration-method ?extending ?overrider-method)

   ; Overriding method body cannot be empty
   (ast :MethodDeclaration ?overrider-method)
   (has :body ?overrider-method ?overrider-body)
   (fails (value|null ?overrider-body))

   ; Overrider is public
   (has :modifiers ?overrider-method ?modifiers)
   (modifier|public ?mod-public)
   (contains ?modifiers ?mod-public)))
```

Listing 19: Checks if the given method is an overriding method.

```clj
; Check with algorithm step in template-body
(defn is-algorithm-step [?template-body ?overrider-method]
 (fresh [?stmts ?overrider-name]

  ; Overrider method name
  (has :name ?overrider-method ?overrider-name)
  ; Extracts rhs method invocation from the body of the template method
  (has :statements ?template-body ?stmts)

  ; We need to at least have one match
  (one (fresh  [?stmt ?expression ?rhs ?inv-name]
          ; Iterate each stmt
          (contains ?stmts ?stmt)
          ; Extract name
          (has :expression ?stmt ?expression)
          (has :rightHandSide ?expression ?rhs)
          (has :name ?rhs ?inv-name)

          ; 'Compare' with overrider method name
          (name|simple-name|simple|same ?overrider-name ?inv-name)))))
```

Listing 20: Checks the body of the template-method to see if the overrider method is invoced in the body of the method.

```clj
; Is template method?
(defn is-template-method [?template-method ?template-body]
 (fresh [?mod-abstract ?mod-public ?modifiers ?raw]
  (ast :MethodDeclaration ?template-method)

  ; Public and non abstract
  (has :modifiers ?template-method ?modifiers)

  ; Public method
  (modifier|public ?mod-public)
  (contains ?modifiers ?mod-public)
  ; Must be public method only
  (value-raw ?modifiers ?raw)
  (equals 1 (count ?raw))

  ; Non empty body
  (has :body ?template-method ?template-body)
  (fails (value|null ?template-body))))
```

Listing 21: Checks if the given body is a template method.

Bibliography

[1] [Online]. Available: https://en.wikipedia.org/wiki/Template_method_pattern

[2] R. J. J. V. Erich Gamma Richard Helm, *Design Patterns Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.