METRICS

For

Asset Management

Prepared by

Jobin Joseph (U101116FCS053)

Mamidi Lokesh (U101116FCS063)

PVNSSK Chaitanya (U101116FCS085)

PSK Vamsi (U101116FCS088)

Pradeep Yadav (U101116FCS089)

Software Engineering
NIIT University

13th November 2018

Description

The following metrics were calculated for each class:

- WMC: Weighted methods per class
- DIT: Depth of Inheritance Tree
- NOC: Number of Children
- CBO: Coupling between object classes
- RFC: Response for a Class
- LCOM: Lack of cohesion in methods
- Ca: Afferent coupling (not a C&K metric)
- NPM: Number of Public Methods for a class (not a C&K metric)

The corresponding metrics for each class: WMC, DIT, NOC, CBO, RFC, LCOM, Ce, and NPM.

Metrics (for each class)

Action Class

```
filename
C:\Users\CHAITANYA\workspace\AssetManagement\build\classes\controller\
Action.class
compiled from           Action.java
compiler version        52.0
access flags            33
constant pool           291 entries
ACC_SUPER flag          true
Attribute(s):
        SourceFile(Action.java)
        (Unknown attribute RuntimeVisibleAnnotations: 00 01 01 20 00
01 01 21 5b 00... (truncated))
1 fields:
        private static final long serialVersionUID = 1
3 methods:
        public void <init>()
        protected void doGet(javax.servlet.http.HttpServletRequest
request, javax.servlet.http.HttpServletResponse response)
                throws javax.servlet.ServletException,
java.io.IOException
```

```
        protected void doPost(javax.servlet.http.HttpServletRequest
request, javax.servlet.http.HttpServletResponse response)
                throws javax.servlet.ServletException,
java.io.IOException
```

```
controller.Action 3 0 0 2 30 3 0 1)
```

AdminAction Class

```
filename
C:\Users\CHAITANYA\workspace\AssetManagement\build\classes\controller\
AdminAction.class
compiled from           AdminAction.java
compiler version        52.0
access flags            33
constant pool           234 entries
ACC_SUPER flag          true
Attribute(s):
        SourceFile(AdminAction.java)
6 methods:
        public void <init>(int id, String name, String actor)
        String create(String name, String category)
        String addasset(String name)
        String add(String name, int count)
        String updateStatus(String asset_id, String status)
        String updateLocation(String asset_id, String location, String
room)
```

```
controller.AdminAction 6 0 0 4 32 15 0 1)
```

AuditorActionClass

```
filename
C:\Users\CHAITANYA\workspace\AssetManagement\build\classes\controller\
AuditorAction.class
compiled from           AuditorAction.java
compiler version        52.0
access flags            33
constant pool           248 entries
ACC_SUPER flag          true
Attribute(s):
        SourceFile(AuditorAction.java)
1 fields:
        String function
```

```
6 methods:
        public void <init>(int id, String name, String actor)
        String addasset(String name)
        String add(String name, int count)
        String updateStatus(String asset_id, String status)
        String updateLocation(String asset_id, String location, String
room)
        String remove(String asset_id)
```

```
controller.AuditorAction 6 0 0 4 32 15 0 1)
```

AuditorActionServ Class

```
filename
C:\Users\CHAITANYA\workspace\AssetManagement\build\classes\controller\
AuditorActionServ.class
compiled from           AuditorActionServ.java
compiler version        52.0
access flags            33
constant pool           229 entries
ACC_SUPER flag          true
Attribute(s):
        SourceFile(AuditorActionServ.java)
        (Unknown attribute RuntimeVisibleAnnotations: 00 01 00 e2 00
01 00 e3 5b 00... (truncated))
1 fields:
        private static final long serialVersionUID = 1
3 methods:
        public void <init>()
        protected void doGet(javax.servlet.http.HttpServletRequest
request, javax.servlet.http.HttpServletResponse response)
                throws javax.servlet.ServletException,
java.io.IOException
        protected void doPost(javax.servlet.http.HttpServletRequest
request, javax.servlet.http.HttpServletResponse response)
                throws javax.servlet.ServletException,
java.io.IOException
```

```
controller.AuditorActionServ 3 0 0 2 26 3 0 1)
```

Main Class

```
filename
C:\Users\CHAITANYA\workspace\AssetManagement\build\classes\controller\
main.class
compiled from           main.java
compiler version        52.0
access flags            33
constant pool           199 entries
ACC_SUPER flag          true
```

```
Attribute(s):
        SourceFile(main.java)
        (Unknown attribute RuntimeVisibleAnnotations: 00 01 00 c4 00
01 00 c5 5b 00... (truncated))
4 fields:
        private static final long serialVersionUID = 1
        static final String url = "jdbc:mysql://localhost:3306/asset"
        static final String user = "root"
        static final String pass = "lokesh1999"
3 methods:
        public void <init>()
        protected void doGet(javax.servlet.http.HttpServletRequest
request, javax.servlet.http.HttpServletResponse response)
                throws javax.servlet.ServletException,
java.io.IOException
        protected void doPost(javax.servlet.http.HttpServletRequest
request, javax.servlet.http.HttpServletResponse response)
                throws javax.servlet.ServletException,
java.io.IOException
```

```
controller.main 3 0 0 4 25 3 0 1
```

StaffAction Class

```
filename
C:\Users\CHAITANYA\workspace\AssetManagement\build\classes\controller\
StaffAction.class
compiled from           StaffAction.java
compiler version        52.0
access flags            33
constant pool           203 entries
ACC_SUPER flag          true
Attribute(s):
        SourceFile(StaffAction.java)
1 fields:
        String function
3 methods:
        public void <init>(int id, String name, String actor)
        String updateStatus(String asset_id, String status)
        String updateLocation(String asset_id, String location, String
room)
```

```
controller.StaffAction 3 0 0 4 28 3 0 1)
```

StaffActionServ Class

```
filename
C:\Users\CHAITANYA\workspace\AssetManagement\build\classes\controller\
StaffActionServ.class
compiled from           StaffActionServ.java
compiler version        52.0
access flags            33
constant pool           198 entries
ACC_SUPER flag          true
Attribute(s):
        SourceFile(StaffActionServ.java)
        (Unknown attribute RuntimeVisibleAnnotations: 00 01 00 c3 00
01 00 c4 5b 00... (truncated))
1 fields:
        private static final long serialVersionUID = 1
3 methods:
        public void <init>()
        protected void doGet(javax.servlet.http.HttpServletRequest
request, javax.servlet.http.HttpServletResponse response)
                throws javax.servlet.ServletException,
java.io.IOException
        protected void doPost(javax.servlet.http.HttpServletRequest
request, javax.servlet.http.HttpServletResponse response)
                throws javax.servlet.ServletException,
java.io.IOException
```

```
controller.StaffActionServ 3 0 0 2 24 3 0 1)
```

Time Class

```
controller.time 2 1 0 0 6 1 0 2)
```

View Class

```
controller.View 4 1 0 2 19 6 0 1)
```

User Class

```
model.user 7 1 0 0 8 3 0 7)
```

MainView Class

```
filename
C:\Users\CHAITANYA\workspace\AssetManagement\build\classes\view\mainVi
ew.class
compiled from           mainView.java
compiler version        52.0
access flags            33
constant pool           138 entries
ACC_SUPER flag          true

Attribute(s):
        SourceFile(mainView.java)
        (Unknown attribute RuntimeVisibleAnnotations: 00 01 00 87 00
01 00 88 5b 00... (truncated))
1 fields:
        private static final long serialVersionUID = 1
3 methods:
        public void <init>()
        protected void doGet(javax.servlet.http.HttpServletRequest
request, javax.servlet.http.HttpServletResponse response)
                throws javax.servlet.ServletException,
java.io.IOException
        protected void doPost(javax.servlet.http.HttpServletRequest
request, javax.servlet.http.HttpServletResponse response)
                throws javax.servlet.ServletException,
java.io.IOException

view.mainView 3 0 0 1 16 3 0 1
```

Appendix

Here is the description of each metric calculated,

WMC - Weighted methods per class

A class's *weighted methods per class* WMC metric is simply the sum of the complexities of its methods. As a measure of complexity, we can use the cyclomatic complexity, or we can arbitrarily assign a complexity value of 1 to each method. The value of the WMC is equal to the number of methods in the class.

DIT - Depth of Inheritance Tree

The *depth of inheritance tree* (DIT) metric provides for each class a measure of the inheritance levels from the object hierarchy top. In Java where all classes inherit Object the minimum value of DIT is 1.

NOC - Number of Children

A class's *number of children* (NOC) metric simply measures the number of immediate descendants of the class.

CBO - Coupling between object classes

The *coupling between object classes* (CBO) metric represents the number of classes coupled to a given class (efferent couplings, Ce). This coupling can occur through method calls, field accesses, inheritance, arguments, return types, and exceptions.

RFC - Response for a Class

The metric called the *response for a class* (RFC) measures the number of different methods that can be executed when an object of that class receives a message (when a method is invoked for that object). Ideally, we would want to find for each method of the class, the methods that class will call, and repeat this for each called method, calculating what is called the *transitive closure* of the method's call graph. This process can however be both expensive and quite inaccurate. In *ckjm*, we calculate a rough approximation to the response set by simply inspecting method calls within the class's method bodies. This simplification was also used in the 1994 Chidamber and Kemerer description of the metrics.

LCOM - Lack of cohesion in methods

A class's *lack of cohesion in methods* (LCOM) metric counts the sets of methods in a class that are not related through the sharing of some of the class's fields. The original definition of this metric (which is the one used in *ckjm*) considers all pairs of a class's methods. In some of these pairs both methods access at least one common field of the class, while in other pairs the two methods to not share any common field accesses. The lack of cohesion in methods is then calculated by subtracting from the number of method pairs that don't share a field access the number of method pairs that do. Note that subsequent definitions of this metric used as a measurement basis the number of disjoint graph components of the class's methods. Others modified the definition of connectedness to include calls between the methods of the class. The program *ckjm* follows the original (1994) definition by Chidamber and Kemerer.

Ca - Afferent couplings

A class's afferent couplings is a measure of how many other classes use the specific class. Ca is calculated using the same definition as that used for calculating CBO (Ce).

NPM - Number of Public Methods

The NPM metric simply counts all the methods in a class that are declared as public. It can be used to measure the size of an API provided by a package.