UNIVERSITÄT OSNABRÜCK

Live Code Smell Detection of Data Clumps in an Integrated Development Environment

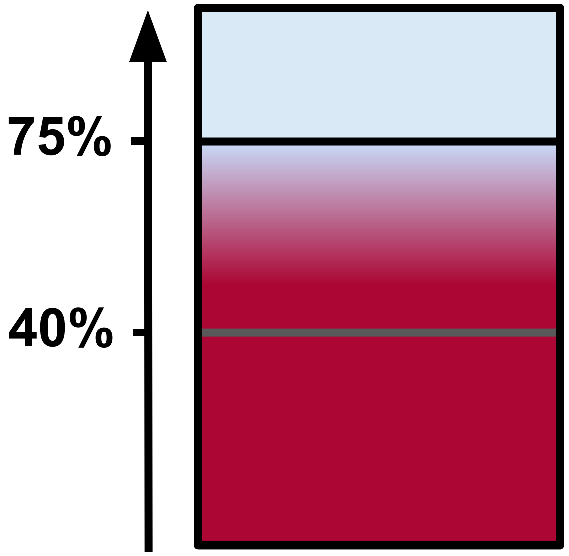
ENASE 2023

Nils Baumgartner, Firas Adleh, Elke Pulvermüller

Outline

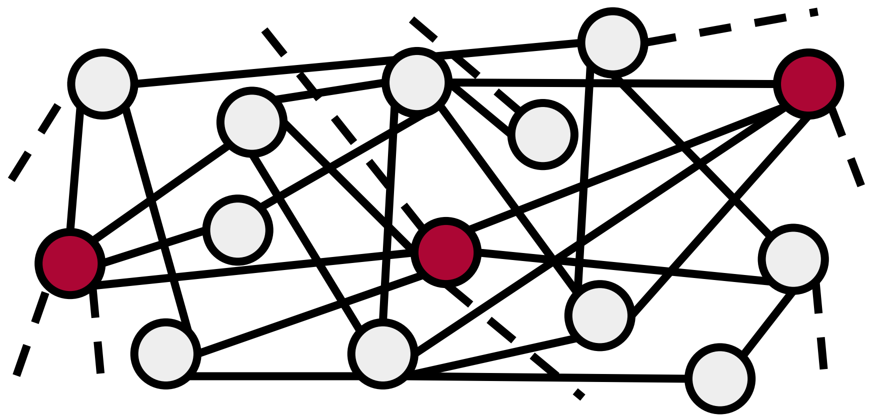
1. Introduction

- High Maintaining cost for software (Brown et al., 1998)
- Maintenance through refactoring (Becker et al., 1999)
- IDEs useful tools for developers



- Maintenance cost

Data clumps spread across project



1. Introduction

- **1-second response time maintains user focus**

  (Miller, 1968) and (Nielsen, 1993)
- **Code smells**
  - **indicator for deeper problems** (Fowler,1999)
  - **can negative impact**
  - **Subjective definition** (Mäntylä and Lassenius, 2006)

1. Introduction

- **Data clumps improved definition** (Zhang et al., 2008)

- **fields**

```
public class MyClass{
     private int foo;
     private int bar;
     private int baz;
     public void myMethod() {}
}

public class MyOtherClass{
     private int bar;
     private int foo;
     public void myOtherMethod(int c) {}
     private int baz;
}
```

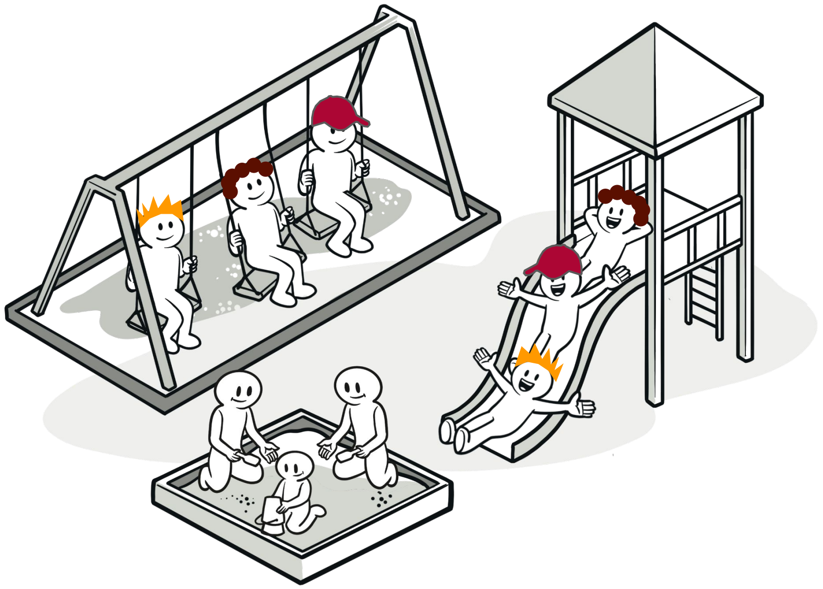
- **parameters**

```
public class MyClass{
     public void myMethod(
          String s, int foo,
          int bar, int baz
     ) {}
}

public class MyOtherClass{
     public void myOtherMethod(
          int bar, int x,
          int foo, int baz
     ) {}
}
```

- **Refactoring Steps** (Fowler, 1999): ***Extract Class, Introduce Parameter Object*** **and** ***Preserve Whole Object***

1. Introduction

- Data clumps "*tend to be like children: They enjoy hanging around together*" (Fowler, 1999)
- Distribution of data clumps across a software project, like children scattering on a playground
- Live Detection LIVE
- Among Top 10 code smells (Lacerda et al., 2020)



2. Related Work

Data clumps detection:

- Stench Blossom - An Interactive Ambient Visualization for Code Smells (Murphy-Hill and Black, 2010)
- CBSD - Some Code Smells Have a Significant but Small Effect on Faults (Hall et al., 2014)

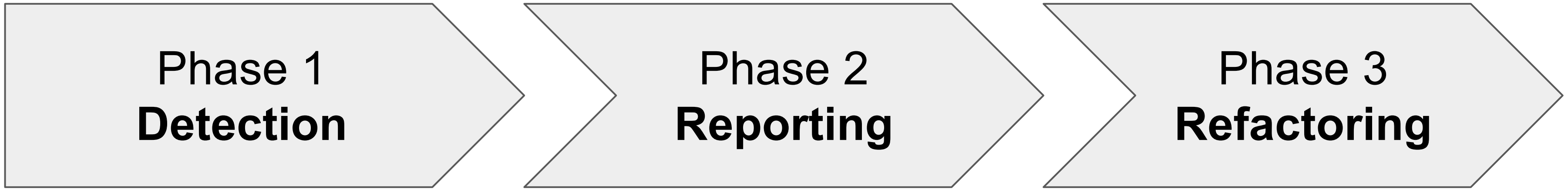
Refactoring suggestions:

- JDeodorant: Clone Refactoring (Mazinanian et al., 2016)
- cASpER: A Plug-in for Automated Code Smell Detection and Refactoring (De Stefano et al., 2020)

- Code smells and refactoring: A tertiary systematic review of challenges and observations (Lacerda et al., 2020)

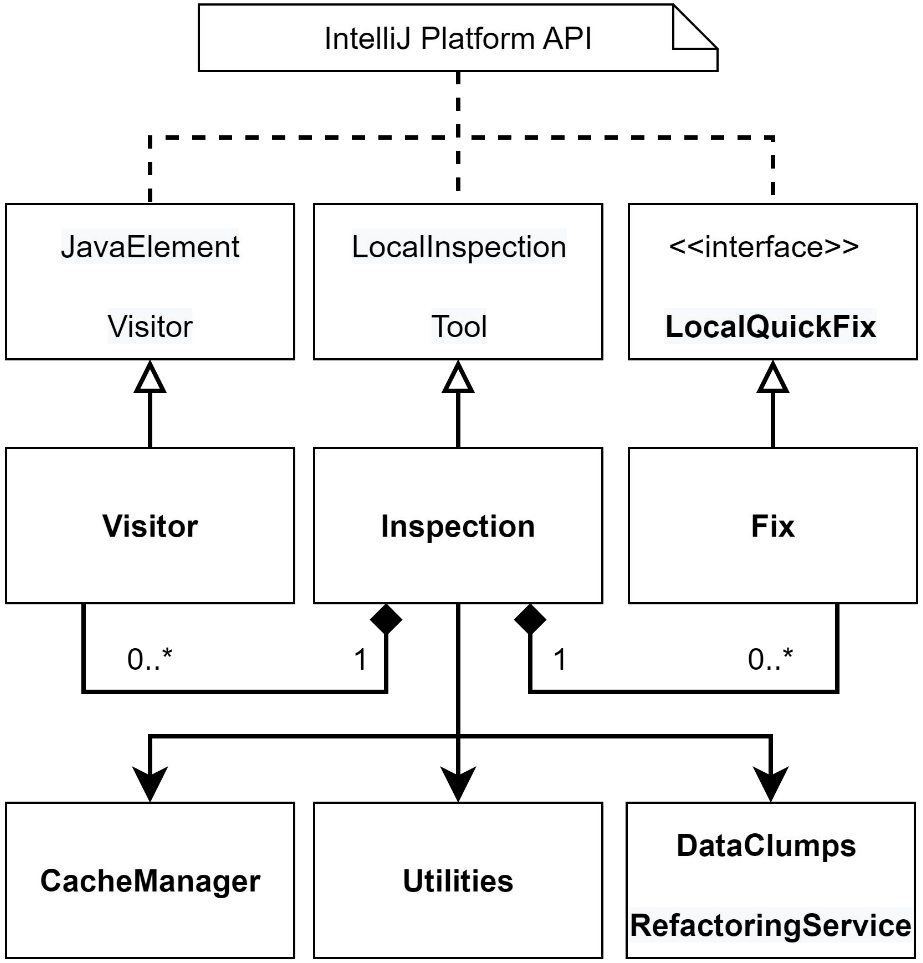
3. Approach

- **Live Code Smell Detection (LCSD) - Java-based plugin for IntelliJ**
- **1. Detection <u>during project changes</u>**
- **2. Reporting of data clumps and refactoring can be initiated**
- **3. Refactoring is applied using provided name of the new class**



3. Approach

- *CacheManager* collects information about the project
- *Visitor* class called after change to source code
- *Fix* classes responsible for refactoring *fields* and *parameters* data clumps
- *Inspection* class manages refactoring and parameters



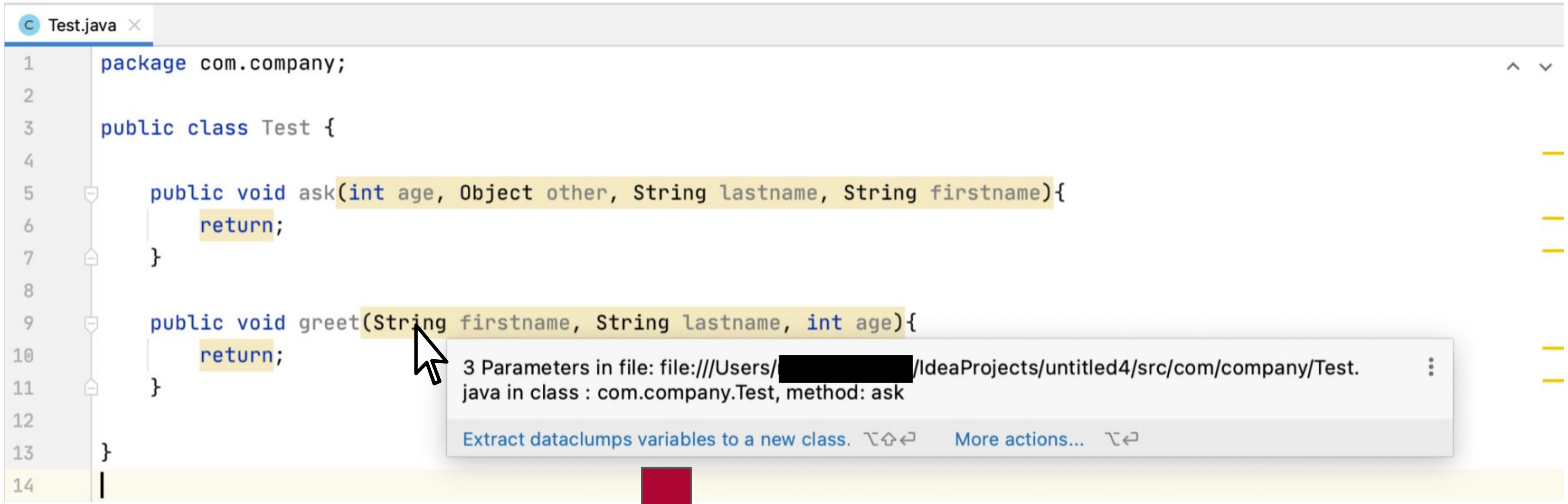
3. Approach: Detection

- Configuration of data clumps definition

3	≥ data fields shared in 2 or more classes
✓	If the hierarchy for data fields should be considered
3	≥ parameters shared in 2 or more methods
✓	If the hierarchy for parameters should be considered
WARN	Level of severity

- Support of live scan and full scan

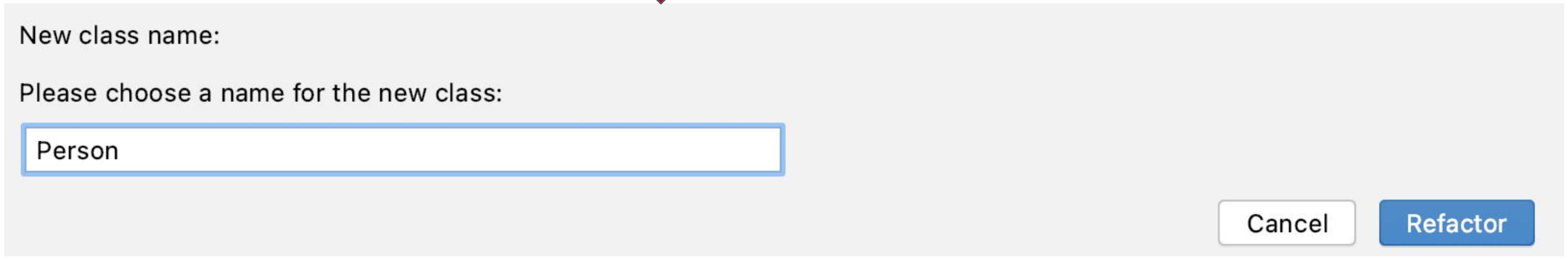
3. Approach: Reporting


Test.java

```
package com.company;

public class Test {

    public void ask(int age, Object other, String lastname, String firstname){
        return;
    }

    public void greet(String firstname, String lastname, int age){
        return;
    }

}
```

3 Parameters in file: file:///Users/[redacted]/IdeaProjects/untitled4/src/com/company/Test.java in class : com.company.Test, method: ask

Extract dataclumps variables to a new class. ⌥⇧↩ More actions... ⌥↩

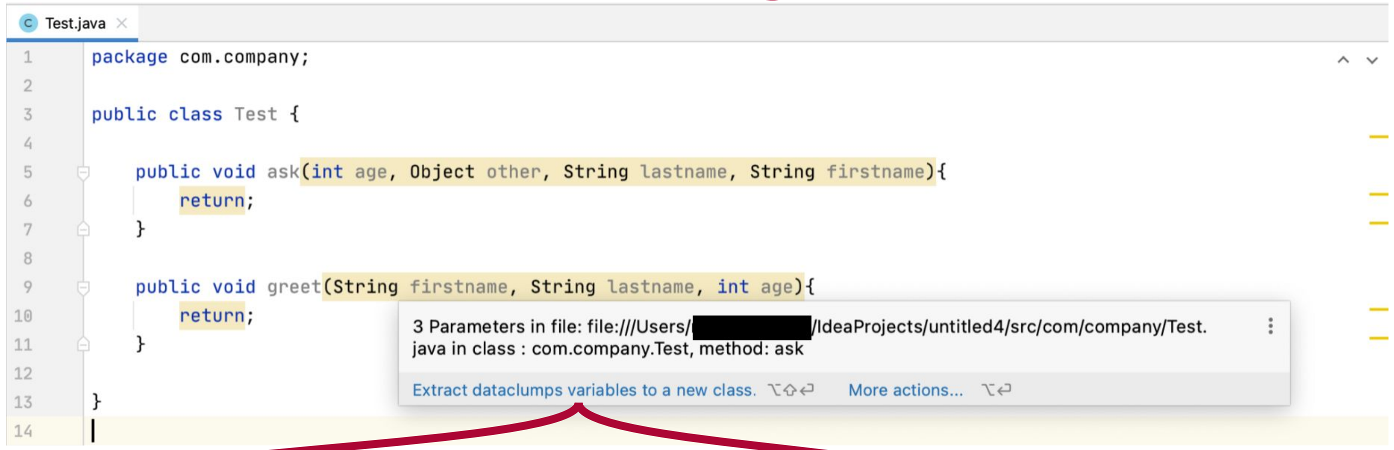

New class name:

Please choose a name for the new class:

Person

Cancel Refactor

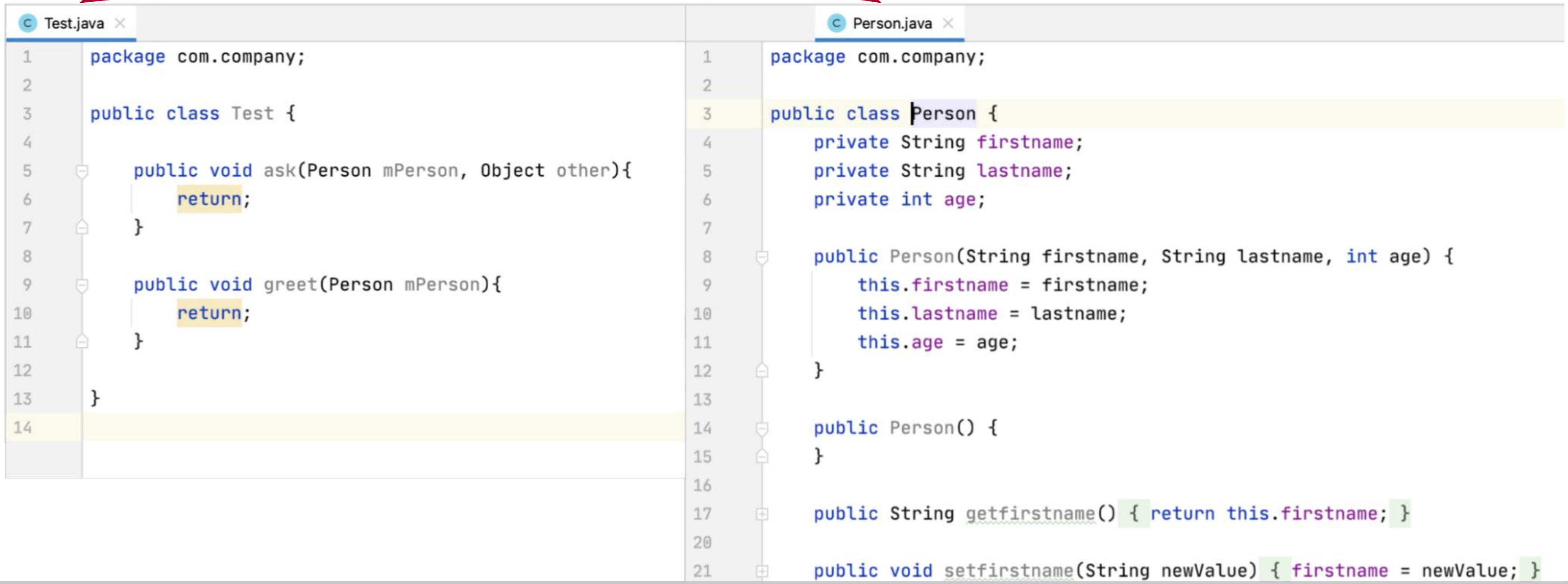
3. Approach: Refactoring



Test.java

```
package com.company;

public class Test {

    public void ask(int age, Object other, String lastname, String firstname){
        return;
    }

    public void greet(String firstname, String lastname, int age){
        return;
    }

}
```

3 Parameters in file: file:///Users/[redacted]/IdeaProjects/untitled4/src/com/company/Test.java in class : com.company.Test, method: ask



Test.java

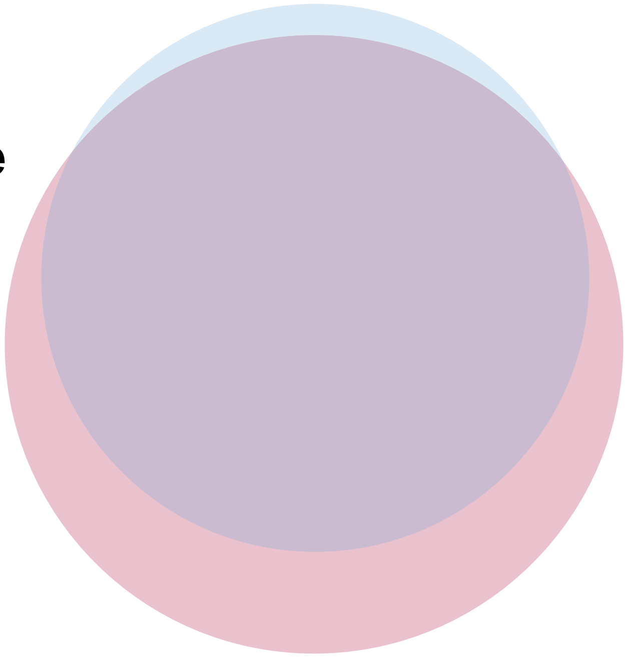
```
package com.company;

public class Test {

    public void ask(Person mPerson, Object other){
        return;
    }

    public void greet(Person mPerson){
        return;
    }

}
```

Person.java

```
package com.company;

public class Person {
    private String firstname;
    private String lastname;
    private int age;

    public Person(String firstname, String lastname, int age) {
        this.firstname = firstname;
        this.lastname = lastname;
        this.age = age;
    }

    public Person() {
    }

    public String getfirstname() { return this.firstname; }

    public void setfirstname(String newValue) { firstname = newValue; }
```

4. Evaluation: Accuracy

- ArgoUML with more than 1500 source files
- *Data clumps* (detected by CBSD tool) in the Unified Bug Data Set (Ferenc et al., 2020) for ArgoUML (v. 0.26 Beta)
  - 97 files containing *data clumps* (after removing non-existing files entries)
- We found 125 files with *data clumps*
- 92 files were the same

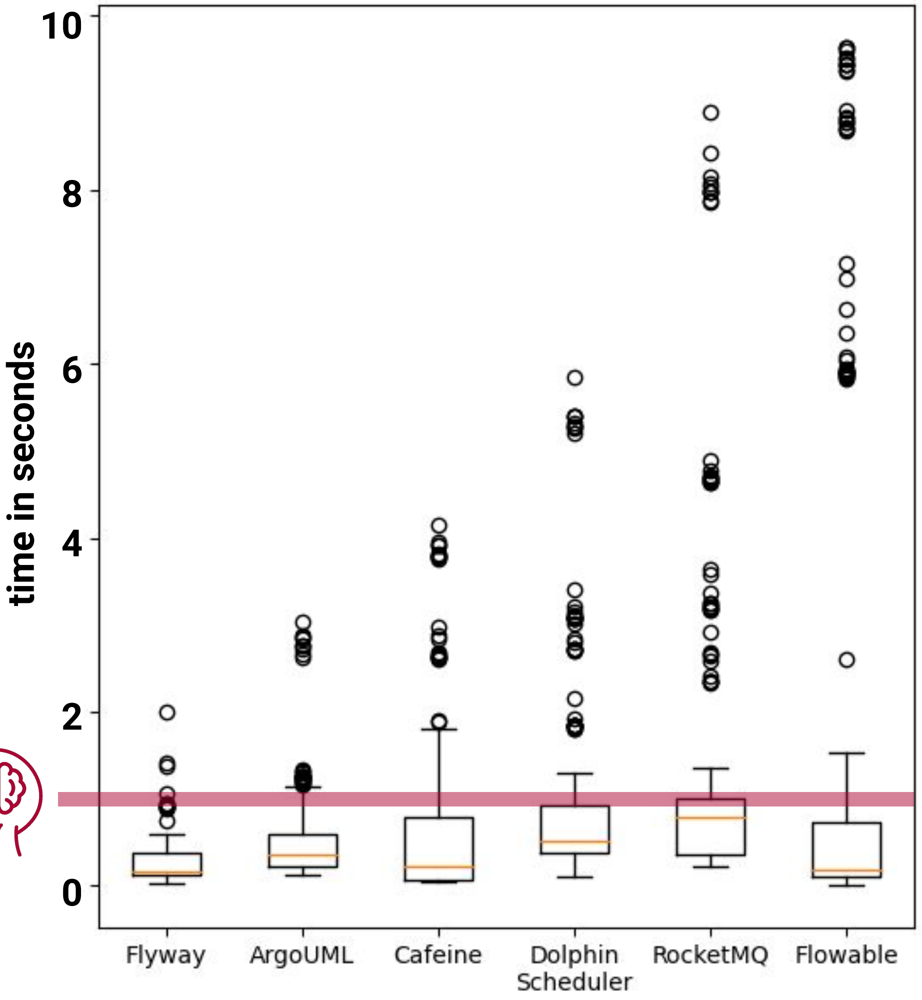


- 🟦 CBSD (97)
- 🟥 LCSD (125)

4. Evaluation: Speed

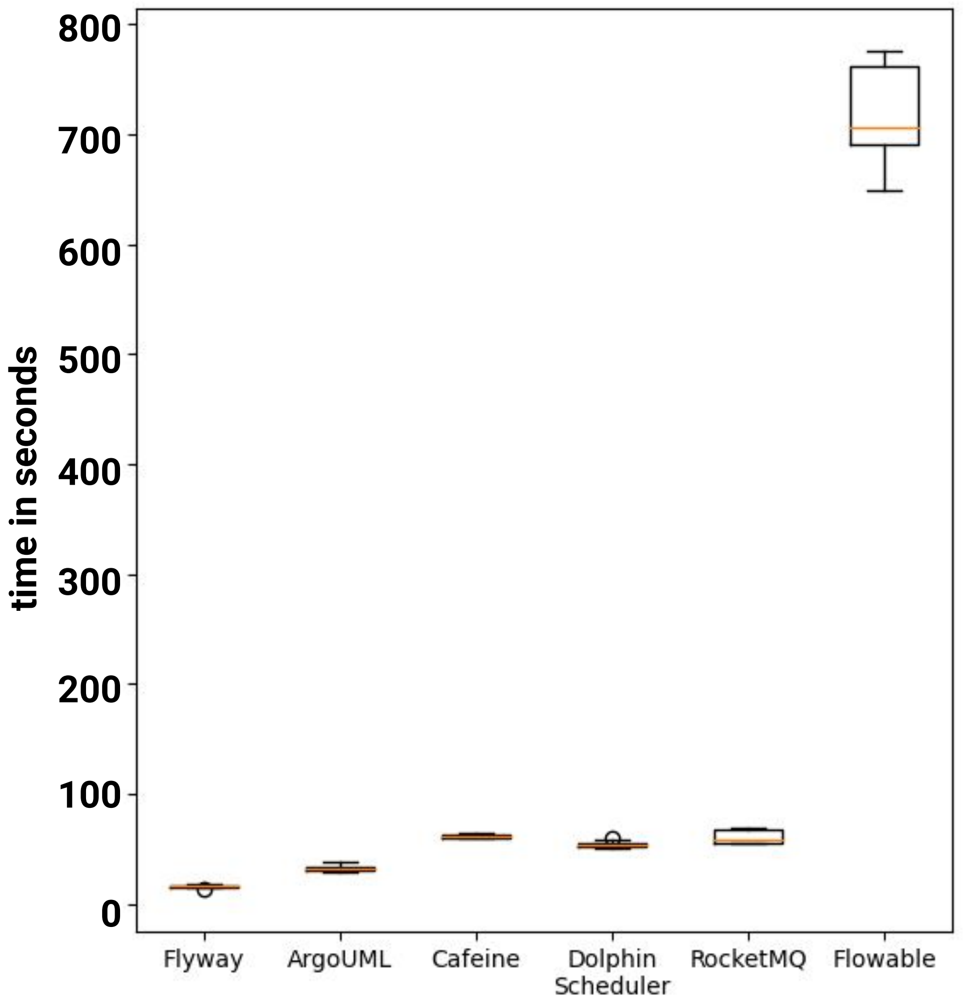
LIVE

- Projects with varying sizes (26-680 KLOC)
- Testing the 20 largest files from each project
- Repeated measurements 10 times
- Initial setup time not included



4. Evaluation: Speed

- Full scan all project files
  - Flyway 26 KLOC
  - RocketMQ 99 KLOC
  - Flowable 680 KLOC
- Repeated measurements 10 times
- Initial setup time not included



5. Conclusion and Future Work

- Conclusion
  - live detection of data clumps
  - Configurable data clumps definition
  - Semi-automatic refactoring for data clumps
- Future Work
  - Data clumps over time in repositories
  - Code smell profiles
  - Semantic name suggestion with AI

- Median below 1 second
- Accuracy ≥ 90 %

References

- Becker, P., Fowler, M., Beck, K., Brant, J., Opdyke, W., and Roberts, D. (1999). Refactoring - Improving the Design of Existing Code. Addison-Wesley Professional, Boston.
- Brown, W. H., Malveau, R. C., McCormick, H. W., and Mowbray, T. J. (1998). AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis.
- Delchev, M. and Harun, M. F. (2015). Investigation of Code Smells in Different Software Domains. Full- scale Software Engineering, page 31.
- De Stefano, M., Gambardella, M. S., Pecorelli, F., Palomba, F., and De Lucia, A. (2020). cASpER: A Plug-in for Automated Code Smell Detection and Refactoring. In Proceedings of the International Conference on Advanced Visual Interfaces, AVI '20, New York, NY, USA. Association for Computing Machinery.
- Ferenc, R., Toth, Z., Ladányi, G., Siket, I., and Gyimóthy, T. (2020). A public unified bug dataset for java and its assessment regarding metrics and bug prediction. Software Quality Journal, 28.
- Fowler, M. (2019). Refactoring - Improving the Design of Existing Code. Addison-Wesley, Amsterdam.
- Mäntylä, M. V. and Lassenius, C. (2006). Subjective Evaluation of Software Evolvability Using Code Smells: An Empirical Study. Empirical Softw. Engg., 11(3):395–431.

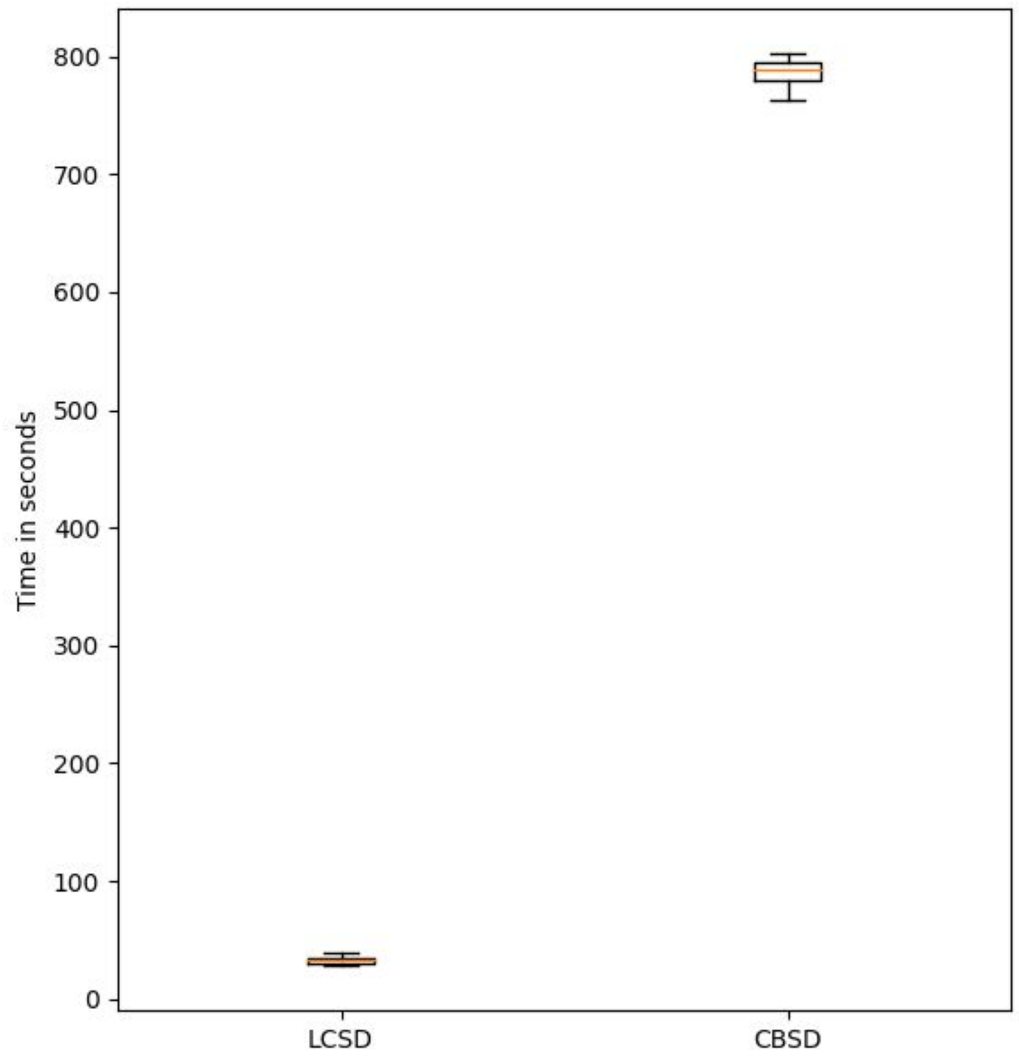
References

- Miller, R. B. (1968). Response Time in Man-Computer Conversational Transactions. In Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I, AFIPS ’68 (Fall, part I), page 267–277, New York, NY, USA. Association for Computing Machinery.
- Nielsen, J. (1993). Chapter 5 – Usability Heuristics.
- Lacerda, G., Petrillo, F., Pimenta, M., and Guéhéneuc, Y. G. (2020). Code smells and refactoring: A tertiary systematic review of challenges and observations. Journal of Systems and Software, 167:110610.
- Zhang, M., Baddoo, N., Wernick, P., and Hall, T. (2008). Improving the Precision of Fowler’s Definitions of Bad Smells. pages 161 – 166.
- Hall, T., Zhang, M., Bowes, D., and Sun, Y. (2014). Some Code Smells Have a Significant but Small Effect on Faults. ACM Trans. Softw. Eng. Methodol., 23(4).
- Murphy-Hill, E. and Black, A. P. (2010). An Interactive Ambient Visualization for Code Smells. In Proceedings of the 5th International Symposium on Software Visualization, SOFTVIS ’10, page 5–14, New York, NY, USA. Association for Computing Machinery.
- Mazinanian, D., Tsantalis, N., Stein, R., and Valenta, Z. (2016). JDeodorant: Clone Refactoring. In 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), pages 613– 616.

Questions?

Backup slide

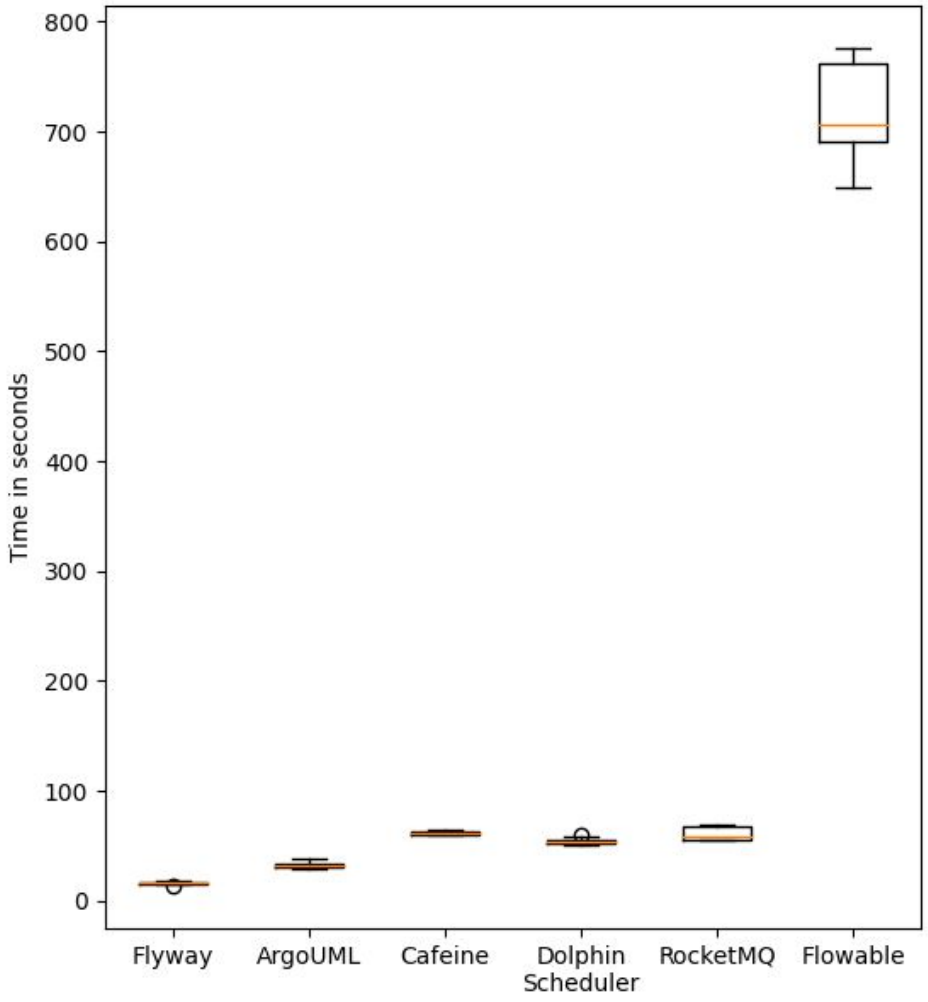
- Full scan of ArgoUML
- LCSD median: 32.5 seconds
- CBSD median: 789 seconds



Backup slide

- Full scan all project files

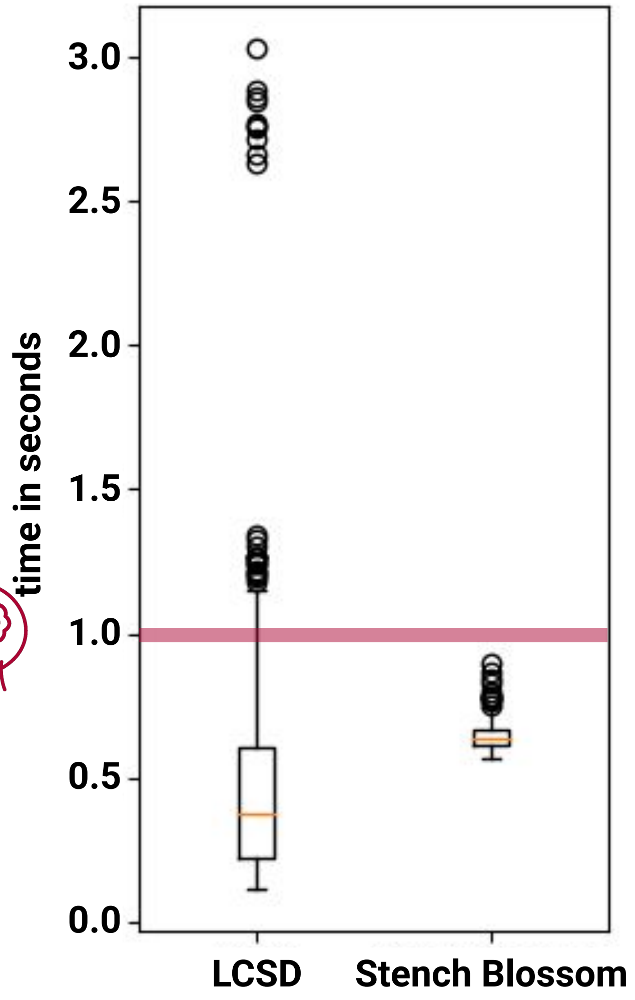
Flyway	26 KLOC
ArgoUML	135 KLOC
Cafeine	60 KLOC
Dolphin Scheduler	92 KLOC
RocketMQ	99 KLOC
Flowable	680 KLOC



Evaluation: Speed: LCSD vs Stench Blossom

LIVE

- Testing the 20 largest files from ArgoUML
- Modified Stench Blossom to only detect data clumps and added a timer
- Initial 5 seconds to open the project in the IDE not included
- Repeated measurements 10 times
  - LCSD median: 0.36 s
  - Stench Blossom median: 0.63 s

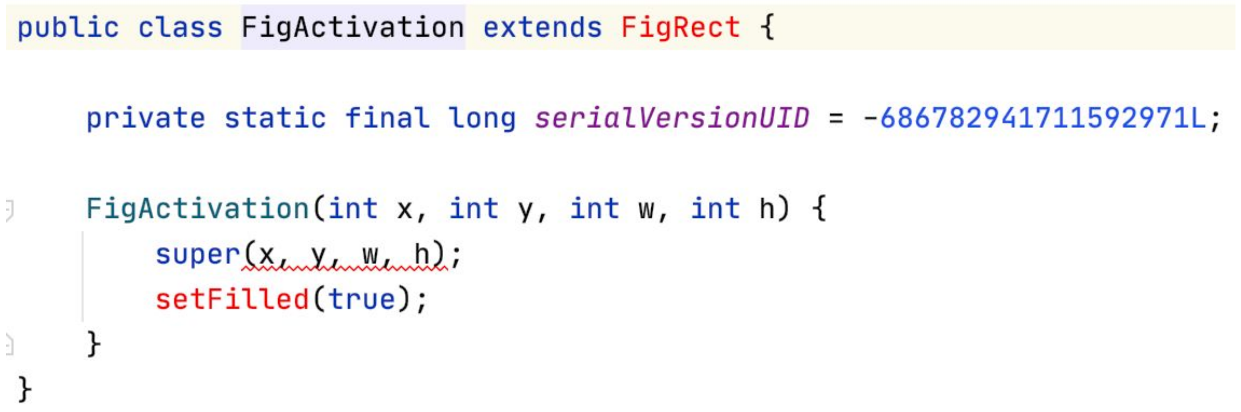


Testing Setup

- All evaluations and tests were performed on the same computer with an Intel Core i7-6700HQ CPU and with 16 GB RAM, running a 64-bit version of Windows 10.

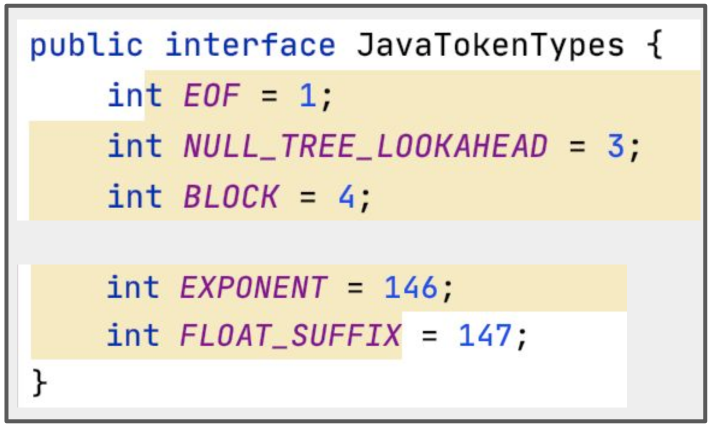
What our tool did not found

- Problem of generics in ADT’s
  - List<String> myStringList;
  - List myStringList;
- Other data clumps to analyse

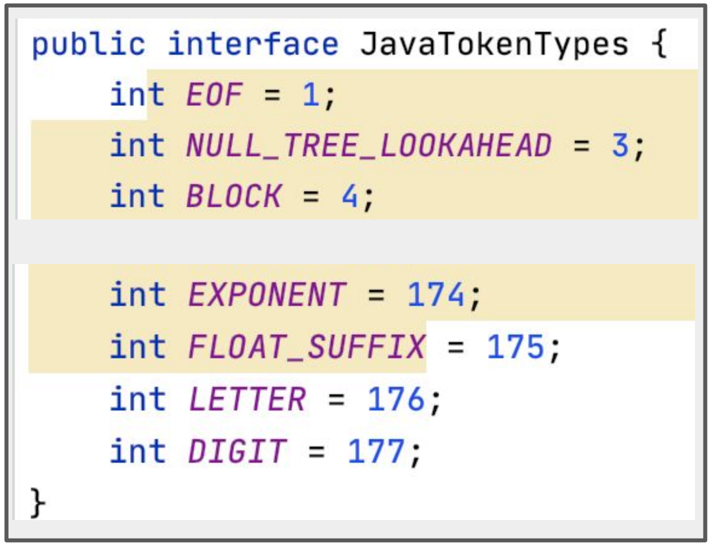


```
public class FigActivation extends FigRect {

    private static final long serialVersionUID = -686782941711592971L;

    FigActivation(int x, int y, int w, int h) {
        super(x, y, w, h);
        setFilled(true);
    }
}
```

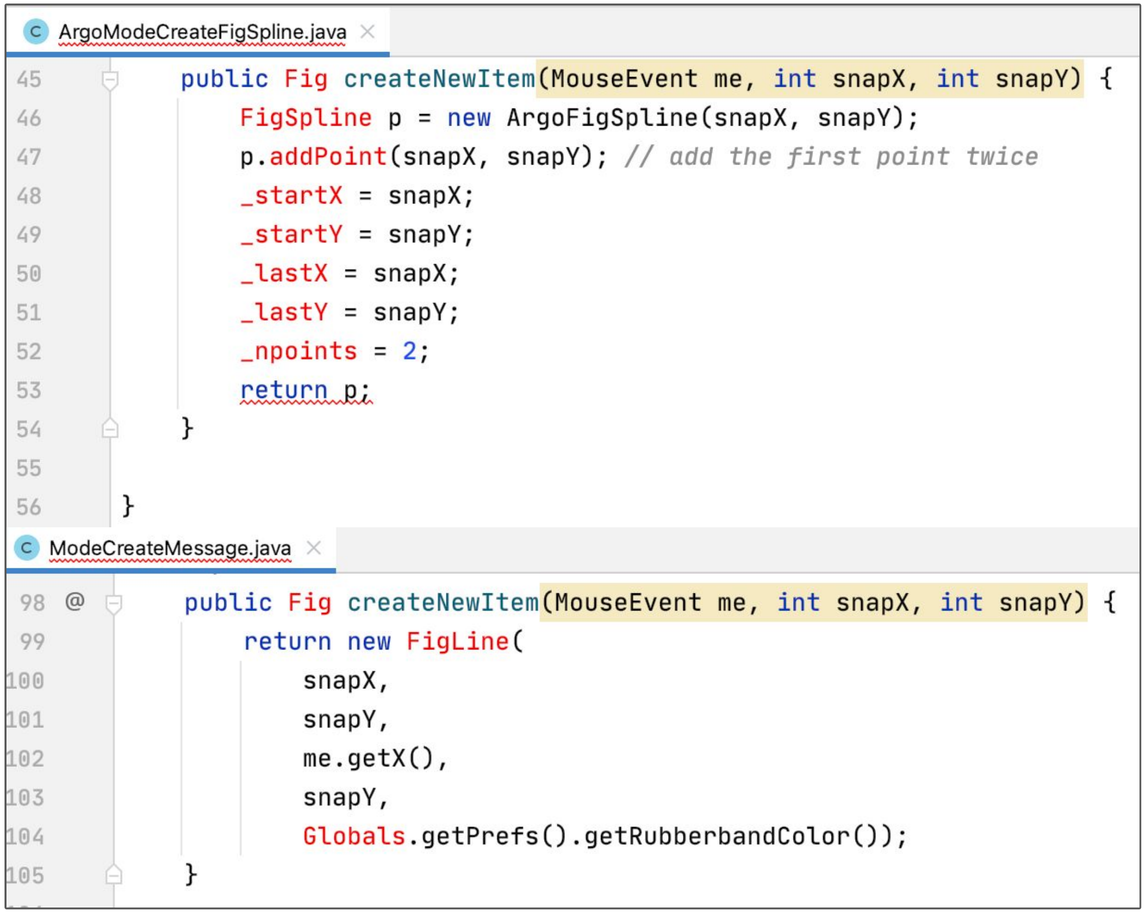
What we found additionally - Example



```java
public interface JavaTokenTypes {
    int EOF = 1;
    int NULL_TREE_LOOKAHEAD = 3;
    int BLOCK = 4;

    int EXPONENT = 146;
    int FLOAT_SUFFIX = 147;
}
```



```java
public interface JavaTokenTypes {
    int EOF = 1;
    int NULL_TREE_LOOKAHEAD = 3;
    int BLOCK = 4;

    int EXPONENT = 174;
    int FLOAT_SUFFIX = 175;
    int LETTER = 176;
    int DIGIT = 177;
}
```



ArgoModeCreateFigSpline.java

```java
    public Fig createNewItem(MouseEvent me, int snapX, int snapY) {
        FigSpline p = new ArgoFigSpline(snapX, snapY);
        p.addPoint(snapX, snapY); // add the first point twice
        _startX = snapX;
        _startY = snapY;
        _lastX = snapX;
        _lastY = snapY;
        _npoints = 2;
        return p;
    }

}
```

ModeCreateMessage.java

```java
    public Fig createNewItem(MouseEvent me, int snapX, int snapY) {
        return new FigLine(
            snapX,
            snapY,
            me.getX(),
            snapY,
            Globals.getPrefs().getRubberbandColor());
    }
```