Mining Change Histories for Systematic Changes

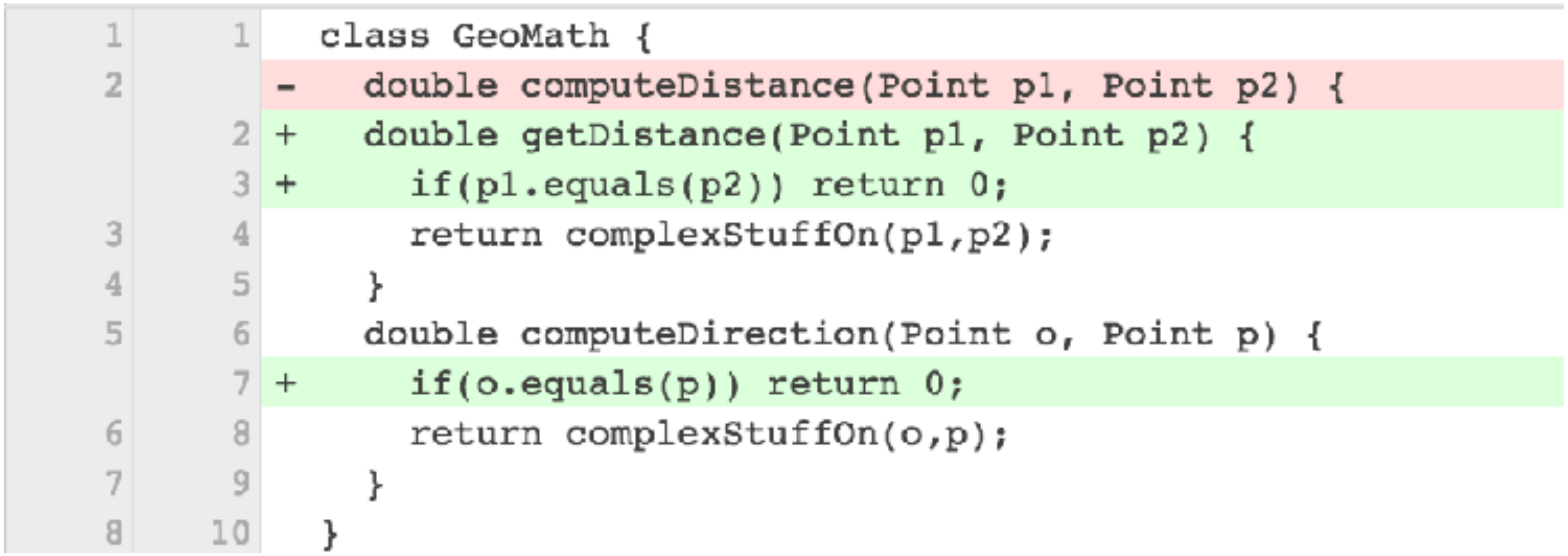
Presenter: Reinout Stevens

Arvid De Meyer (2015). **Mining Change Histories for Unknown Change Patterns.**
Master's thesis, Vrije Universiteit Brussel

Context

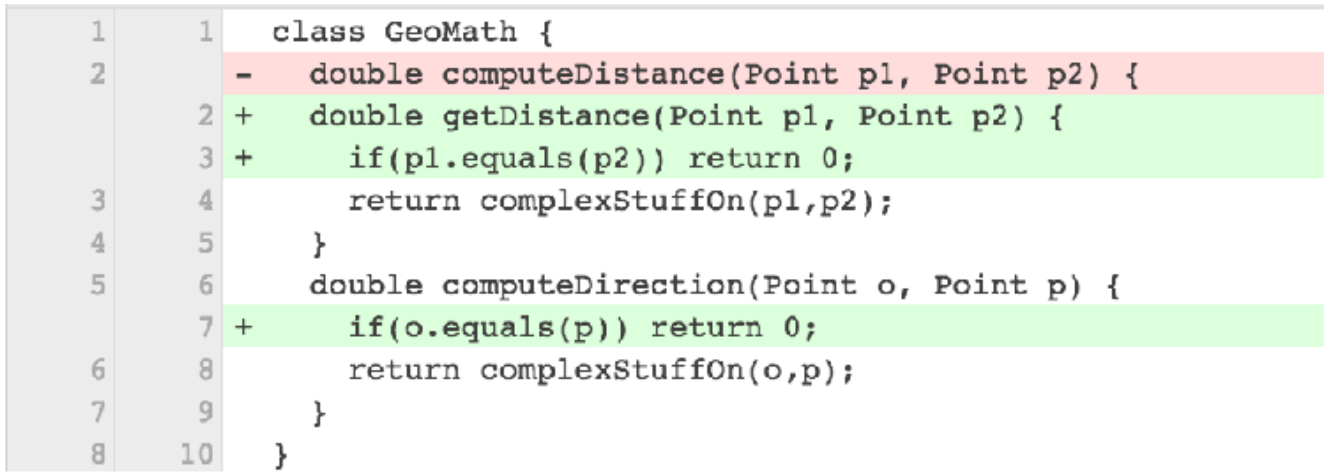
```
class GeoMath {
  double computeDistance(Point p1, Point p2) {
    return complexStuffOn(p1,p2);
  }
  double computeDirection(Point o, Point p) {
    return complexStuffOn(o,p);
  }
}
```

```
class GeoMath {
  double getDistance(Point p1, Point p2) {
    if(p1.equals(p2)) return 0;
    return complexStuffOn(p1,p2);
  }
  double computeDirection(Point o, Point p) {
    if(o.equals(p)) return 0;
    return complexStuffOn(o,p);
  }
}
```



```
  class GeoMath {
-   double computeDistance(Point p1, Point p2) {
+   double getDistance(Point p1, Point p2) {
+     if(p1.equals(p2)) return 0;
      return complexStuffOn(p1,p2);
    }
    double computeDirection(Point o, Point p) {
+     if(o.equals(p)) return 0;
      return complexStuffOn(o,p);
    }
  }
```

Goal



```
  class GeoMath {
-   double computeDistance(Point p1, Point p2) {
+   double getDistance(Point p1, Point p2) {
+     if(p1.equals(p2)) return 0;
      return complexStuffOn(p1,p2);
    }
    double computeDirection(Point o, Point p) {
+     if(o.equals(p)) return 0;
      return complexStuffOn(o,p);
    }
  }
```

```
double <method-name>(Point <param1>, Point <param2>) {
   <method-body>
}
```

↓

```
double <method-name>(Point <param1>, Point <param2>) {
   if(<param1>.equals(<param2>)) return 0;
   <method-body>
}
```

Change Distilling

```
class GeoMath {
  double computeDistance(Point p1, Point p2) {
    return complexStuffOn(p1,p2);
  }


  double computeDirection (Point o, Point p) {
    return complexStuffOn(o,p);
  }
}
```

```
class GeoMath {
  double getDistance(Point p1, Point p2) {
    if(p1.equals(p2)) return 0;
    return complexStuffOn(p1,p2);
  }

  double computeDirection(Point o, Point p)
    if(o.equals(p)) return 0;
    return complexStuffOn(o,p);
  }
}
```

Insert   Update

1. update(computeDistance, getDistance)
2. insert(:body, getDistance(), if( ){})
3. insert(:test, if( ){}, equals( ))
4. insert(:receiver, equals(), p1)
5. insert(:arguments, equals(), p2)
6. insert(:consequent, if( ){ }, return)
7. insert(:arguments, return, 0)
8. …

Frequent Itemset Mining

Transaction	Items
1	{Bread, Butter}
2	{Beer, Milk, Butter}
3	{Bread, Milk, Butter}
4	{Beer, Butter}
5	{Bread, Milk, Butter}

Frequent pattern (support 3)
{Bread, Butter}

~~**Frequent pattern (support 3)**~~
~~{Bread}~~

…

Negara, S. et al. (2014). **Mining Fine-grained Code Changes to Detect Unknown Change Patterns.** In Proceedings of the 36th International Conference on Software Engineering (ICSE) (pp. 803–813). May 31 - June 07, 2014, Hyderabad, IN.

Applied to Changes

Transaction	Items
1	{Bread, Butter}
2	{Beer, Milk, Butter}
3	{Bread, Milk, Butter}
4	{Beer, Butter}
5	{Bread, Milk, Butter}

Q1. How to group changes in transactions?

Q2. When are two changes considered equal?

Grouping Analogy

Transaction	Items
1	{Bread, Butter}
2	{Beer, Milk, Butter}
3	{Bread, Beer, Butter}

Transaction	Items
1	{Saw, Gloves, Nails}
2	{Hammer, Nails, Wood}
3	{Nails, Gloves, Hammer}

Transaction	Items
1	{Coughing Sirop}
2	{Band Aid, Painkillers, Rubbing Alcohol}
3	{Band Aid, Painkillers}

Transaction	Items
1	{Bread, Butter, Saw, Gloves, Nails, Sirop}
2	{Beer, Milk, Butter, Hammer, Nails, Wood, Band Aid, Painkillers, Rubbing Alcohol}
3	{Bread, Milk, Butter, Nails, Gloves, Hammer, Band Aid, Painkillers}

Grouping

Group	Changes
getDistance()	(update, <SimpleName, getDistance>) (insert, <SimpleName, p1>) (insert, <SimpleName, equals>) (insert, <SimpleName, p2>) (insert, <MethodInv, p1.equals(p2)>) (insert, <NumberLiteral, 0>) (insert, <ReturnStatement, return 0>) (insert, <IfStatement, if (…) …>)
computeDirection()	(insert, <SimpleName, o>) (insert, <SimpleName, equals>) (insert, <SimpleName, p>) (insert, <MethodInv, o.equals(p)>) (insert, <NumberLiteral, 0>) (insert, <ReturnStatement, return 0>) (insert, <IfStatement, if (…) …>)

Method File Package Commit

Equality Analogy

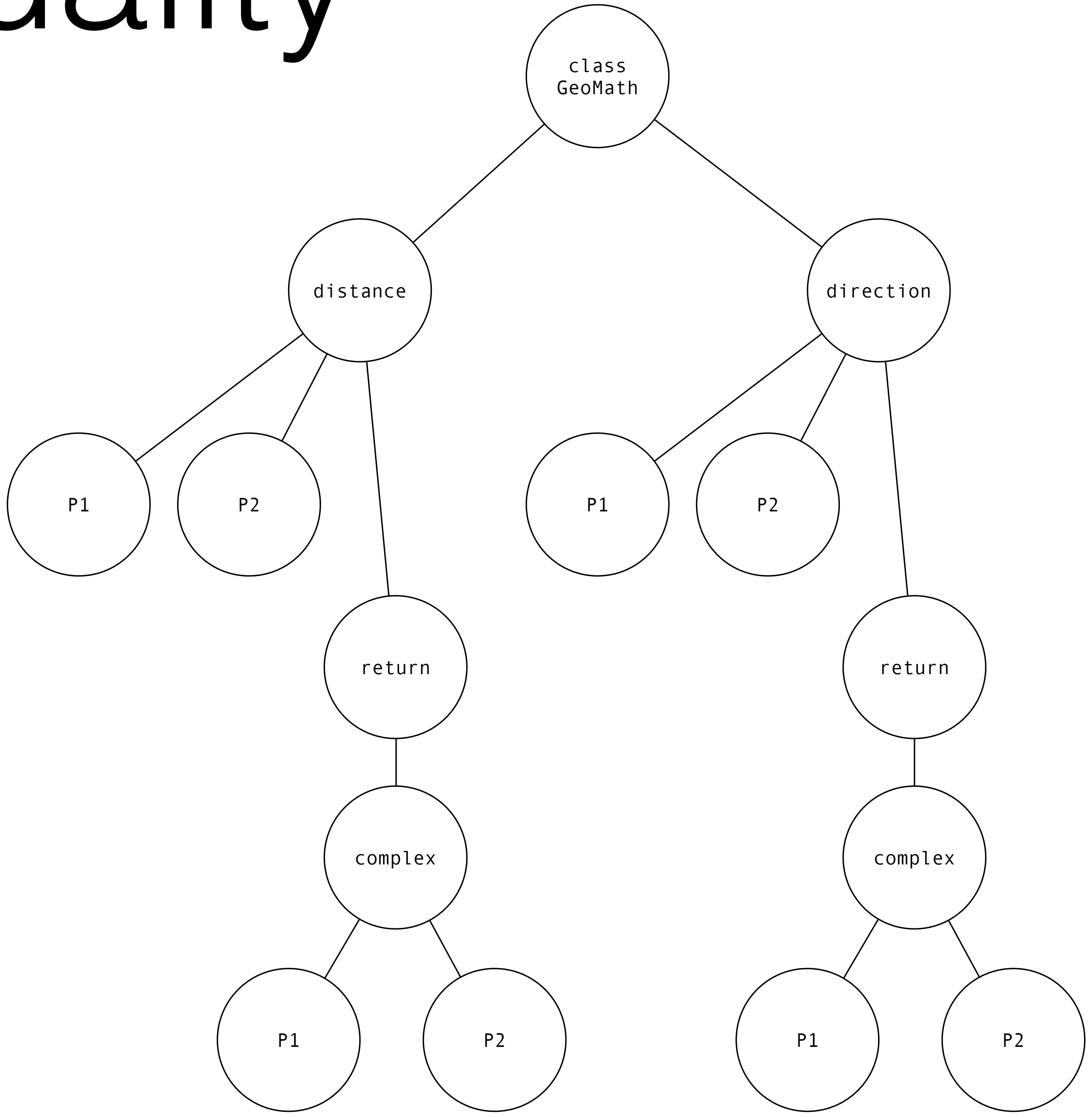
Transaction	Items
1	{White Bread, Butter}
2	{Alcohol-Free Beer, Soy Milk, Margarine}
3	{Brown Bread, Whole Milk, Butter}
4	{Trappist, Chocolate Paste}
5	{French Bread, Skimmed Milk, Butter}

$\downarrow$

Frequent pattern (support 3)
{Bread, Butter}

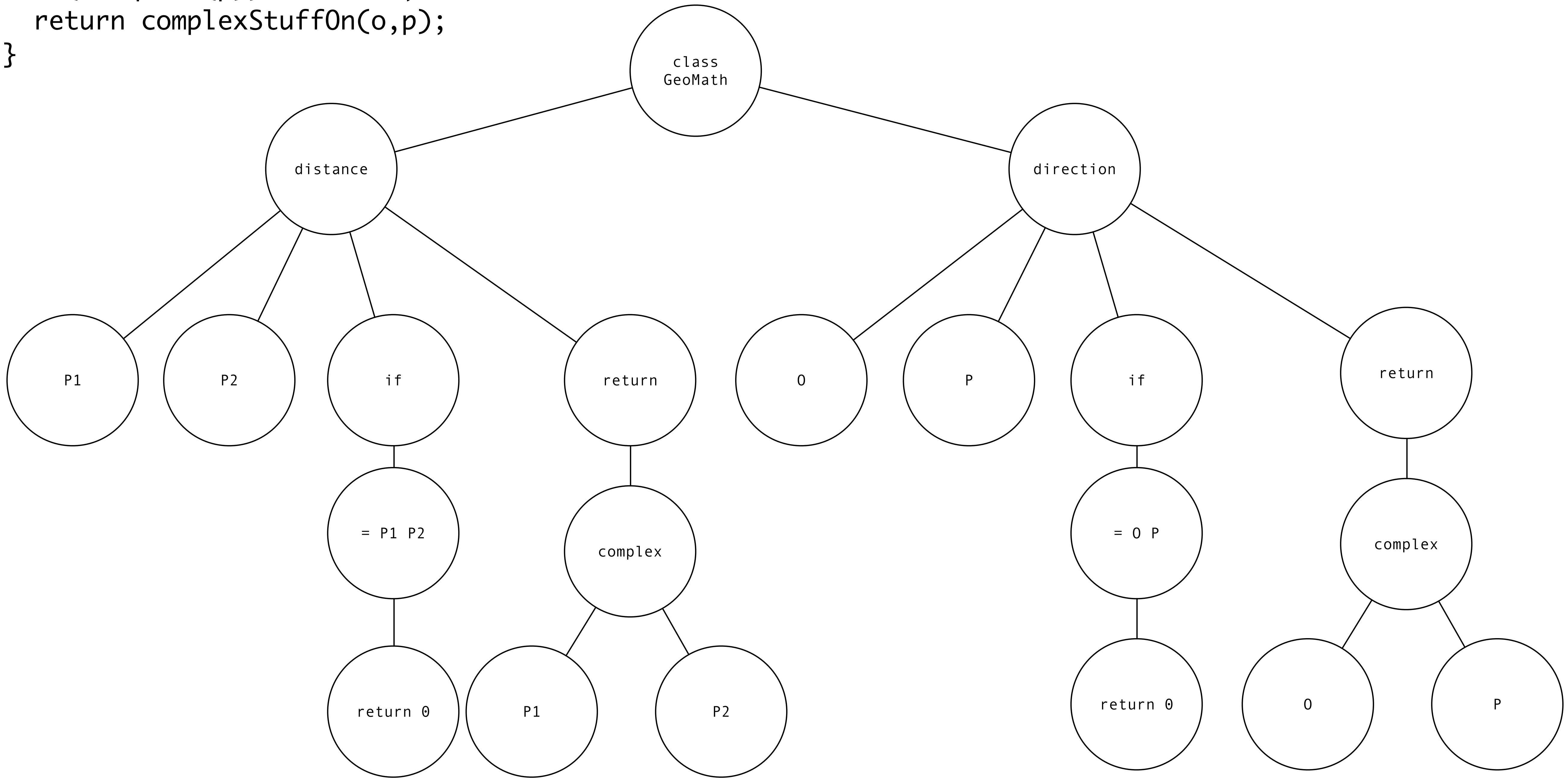
Equality

```
class GeoMath {
  double computeDistance(Point p1, Point p2) {
    return complexStuffOn(p1,p2);
  }
  double computeDirection(Point o, Point p) {
    return complexStuffOn(o,p);
  }
}
```



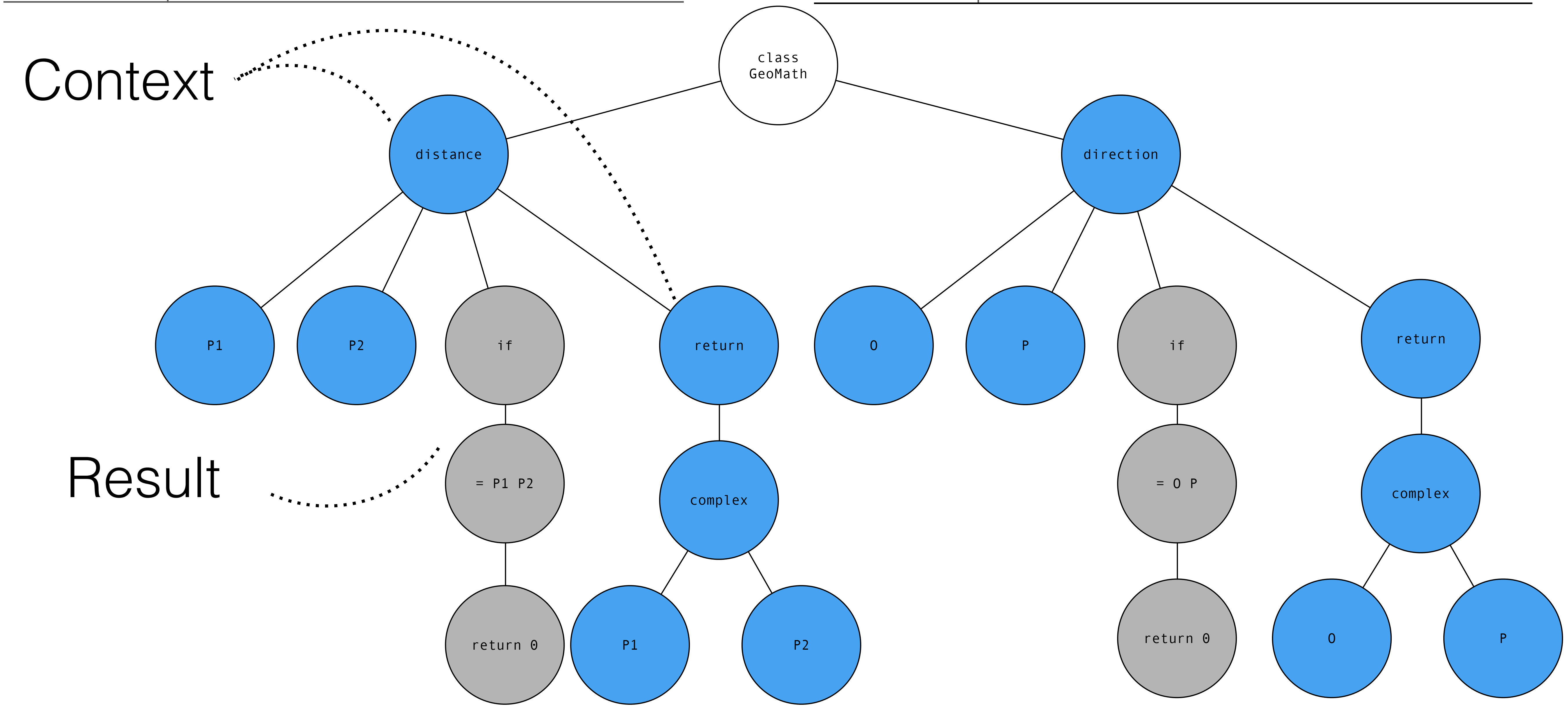
Equality

```
class GeoMath {
  double getDistance(Point p1, Point p2) {
    if(p1.equals(p2)) return 0;
    return complexStuffOn(p1,p2);
  }
  double computeDirection(Point o, Point p) {
    if(o.equals(p)) return 0;
    return complexStuffOn(o,p);
  }
}
```



Group	Changes
distance()	(insert, <SimpleName, p1>) (insert, <SimpleName, equals>) (insert, <SimpleName, p2>) (insert, <MethodInv, p1.equals(p2)>) (insert, <NumberLiteral, 0>) (insert, <ReturnStatement, return 0>) (insert, <IfStatement, if (…) …>)

Group	Changes
direction()	(insert, <SimpleName, o>) (insert, <SimpleName, equals>) (insert, <SimpleName, p>) (insert, <MethodInv, o.equals(p)>) (insert, <NumberLiteral, 0>) (insert, <ReturnStatement, return 0>) (insert, <IfStatement, if (…) …>)



Experiment

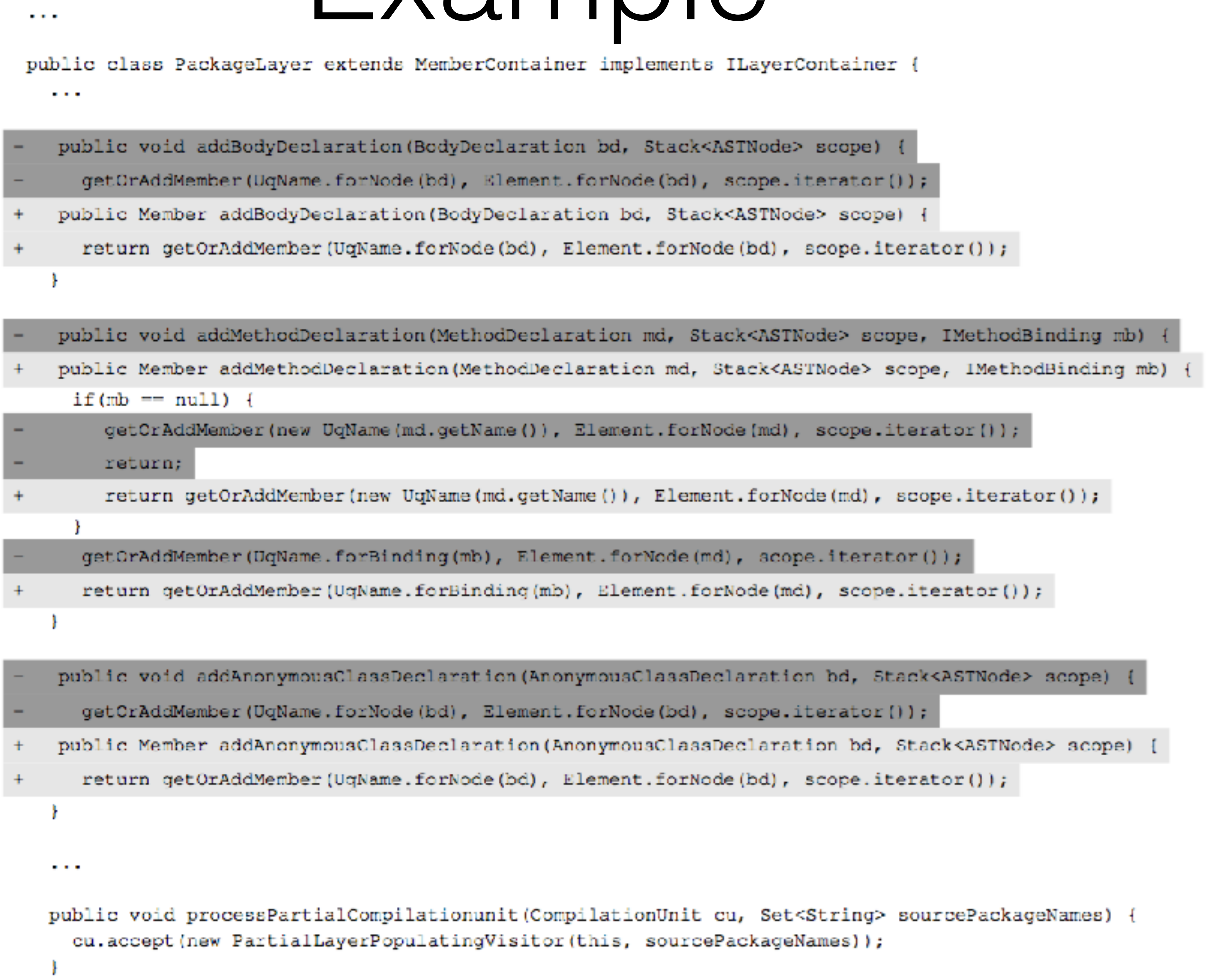
Exapus

- Motivation
  - Familiarity
  - Many systematic-repetitive changes
- Size
  - 263 commits
  - 13198 java LOC over 129 files

Open-Source

- Distilled 22670 changes
- Grouping at Method granularity
- Mined 403 change patterns (< 10 min)

Example



```
...
public class PackageLayer extends MemberContainer implements ILayerContainer {
  ...

-  public void addBodyDeclaration(BodyDeclaration bd, Stack<ASTNode> scope) {
-    getOrAddMember(UqName.forNode(bd), Element.forNode(bd), scope.iterator());
+  public Member addBodyDeclaration(BodyDeclaration bd, Stack<ASTNode> scope) {
+    return getOrAddMember(UqName.forNode(bd), Element.forNode(bd), scope.iterator());
   }

-  public void addMethodDeclaration(MethodDeclaration md, Stack<ASTNode> scope, IMethodBinding mb) {
+  public Member addMethodDeclaration(MethodDeclaration md, Stack<ASTNode> scope, IMethodBinding mb) {
     if(mb == null) {
-      getOrAddMember(new UqName(md.getName()), Element.forNode(md), scope.iterator());
-      return;
+      return getOrAddMember(new UqName(md.getName()), Element.forNode(md), scope.iterator());
     }
-    getOrAddMember(UqName.forBinding(mb), Element.forNode(md), scope.iterator());
+    return getOrAddMember(UqName.forBinding(mb), Element.forNode(md), scope.iterator());
   }

-  public void addAnonymousClassDeclaration(AnonymousClassDeclaration bd, Stack<ASTNode> scope) {
-    getOrAddMember(UqName.forNode(bd), Element.forNode(bd), scope.iterator());
+  public Member addAnonymousClassDeclaration(AnonymousClassDeclaration bd, Stack<ASTNode> scope) {
+    return getOrAddMember(UqName.forNode(bd), Element.forNode(bd), scope.iterator());
   }

  ...

  public void processPartialCompilationunit(CompilationUnit cu, Set<String> sourcePackageNames) {
    cu.accept(new PartialLayerPopulatingVisitor(this, sourcePackageNames));
  }
```

Future Work

- Experiment with different equalities and groupings
- Apply on different code bases