

Configuration Management and Refactorings in Eclipse

Refactoring Scenario: Rename operation

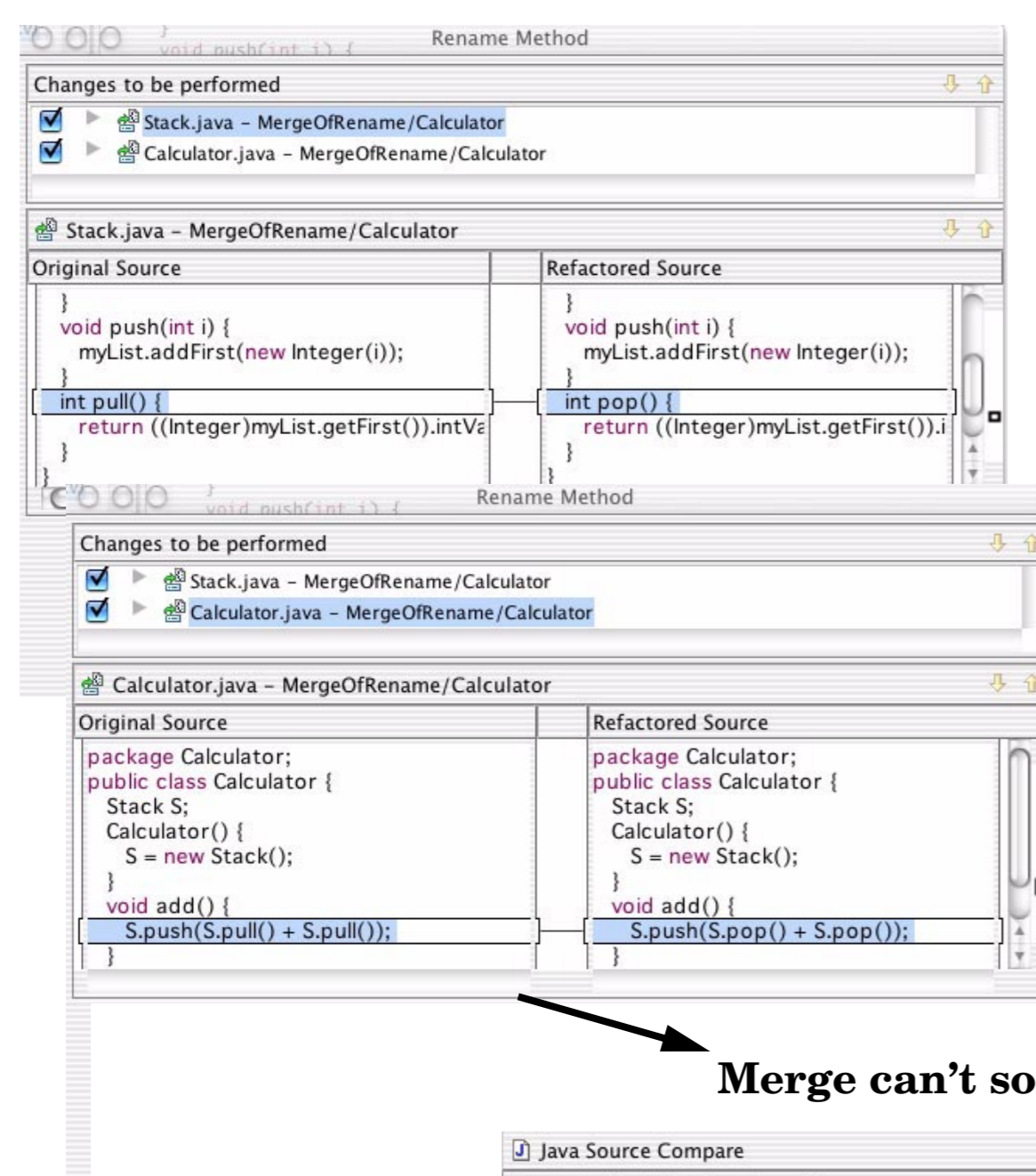
Original

```

Stack.java
package Calculator;
import java.util.*;
public class Stack {
    LinkedList myList;
    Stack() {
        myList = new LinkedList();
    }
    void push(int i) {
        myList.addFirst(new Integer(i));
    }
    int pull() {
        return ((Integer)myList.getFirst()).intValue();
    }
}

Calculator.java
package Calculator;
public class Calculator {
    Stack S;
    Calculator() {
        S = new Stack();
    }
    void add() {
        S.push(S.pull() + S.pull());
    }
}
    
```

User A
Renames Pull to Pop

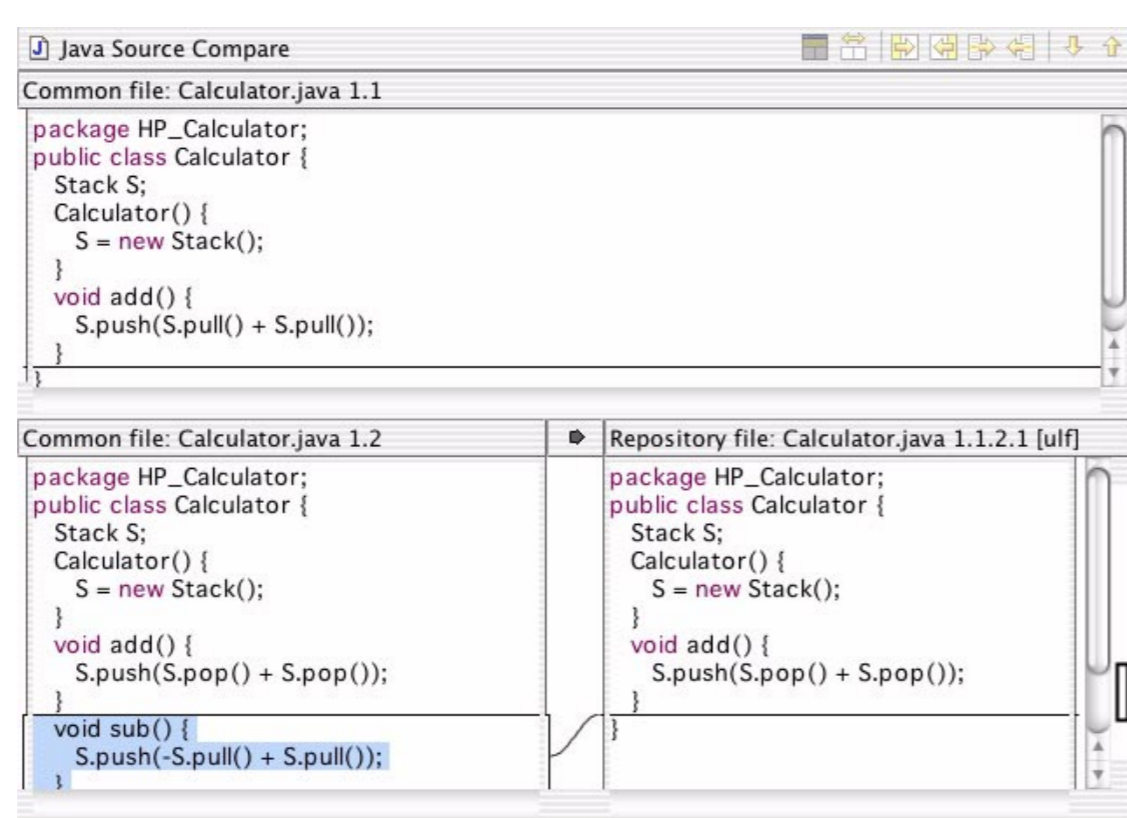


User B
Adds an operation "sub"

```

Calculator.java
package Calculator;
public class Calculator {
    Stack S;
    Calculator() {
        S = new Stack();
    }
    void add() {
        S.push(S.pull() + S.pull());
    }
    void sub() {
        S.push(-S.pull() + S.pull());
    }
}
    
```

Merge can't sort it out



Result: Refactoring
not applied to added
code.

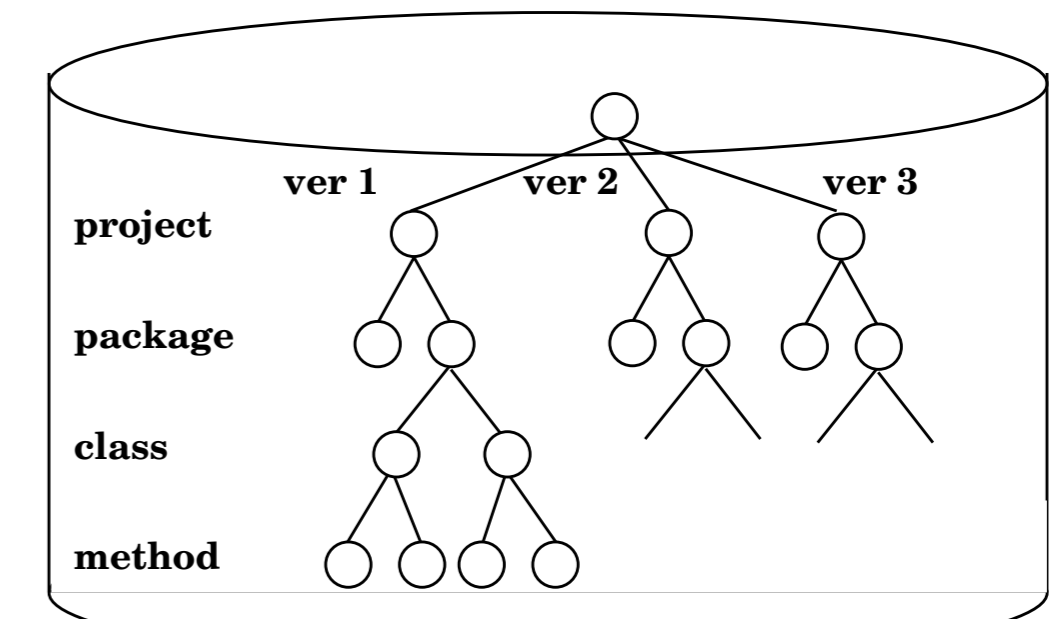
Other refactorings
with similar
merge problems:

- rename of class
- move
- extract method
- ...

Architecture

Use COOP/Orm as the
repository provider

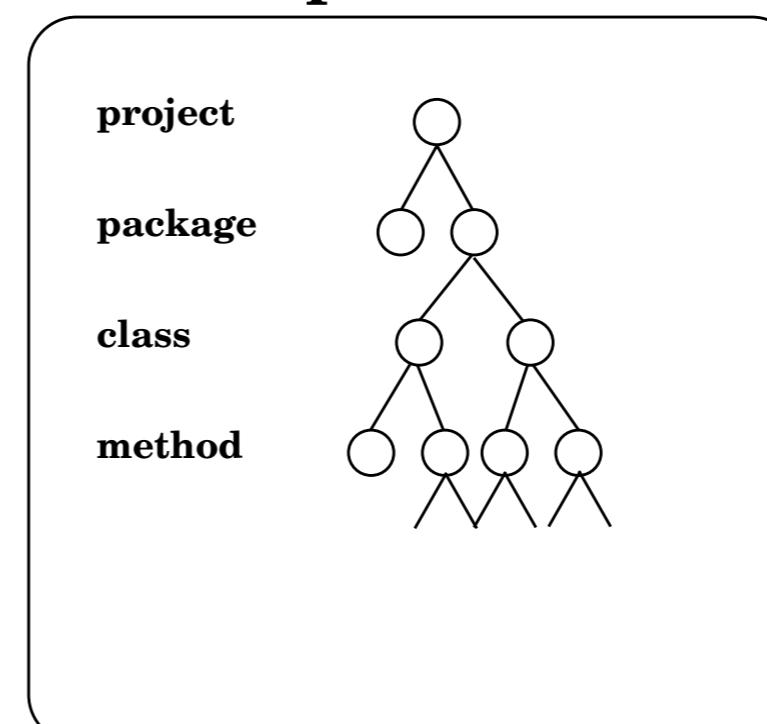
COOP/Orm (Server)



- Hierarchical storage structure
- Store full projects and deltas created by client
- Retrieves all deltas needed to recreate the required version
- Version aware server

Read more about COOP/Orm
<http://www.lucas.lth.se/cm/>

Eclipse (client)



- Create deltas as editor operations (e.g. a refactoring op.)
- Restore old versions from deltas
- Create merge from full version and deltas

How this solves the problem:

- Eclipse constructs a delta with one refactoring (rename pull to pop)
- At merge the refactoring is applied also to the added code

