OPENVALU

BETTER SOFTWARE, FASTER.

OPENREWRITE WOR

Automated refactoring made e

PREREQUISITES FOR THIS WO

- the workshop repository
- An IDE, preferably IntelliJ IDEA
- Java 11, 17 and 21
  - sdkman on mac or linux
  - azul zulu or any other openjdk distributio

sdkman

https://sdkman.io/

java

https://www.azul.com/downloads?package=jdk#zulu

wor

https://gith

WHAT'S YOUR NAME AGA

Sebastian Konieczek

- working as software eng
- talking Java and Kotlin
- likes giving workshops a
- occasionally wakeboard

- 19 years of experience
- likes to talk about observability
- DevEx as a hobby
- trainer for cloud infrastructures

WHAT ARE YOUR EXPECTAT

WHAT YOU CAN EXPEC

- what is open rewrite and how does it
- how can I integrate openrewrite into
- how do I configure openrewrite
- how can I create custom openrewrite
- how do I test my custom openrewrite

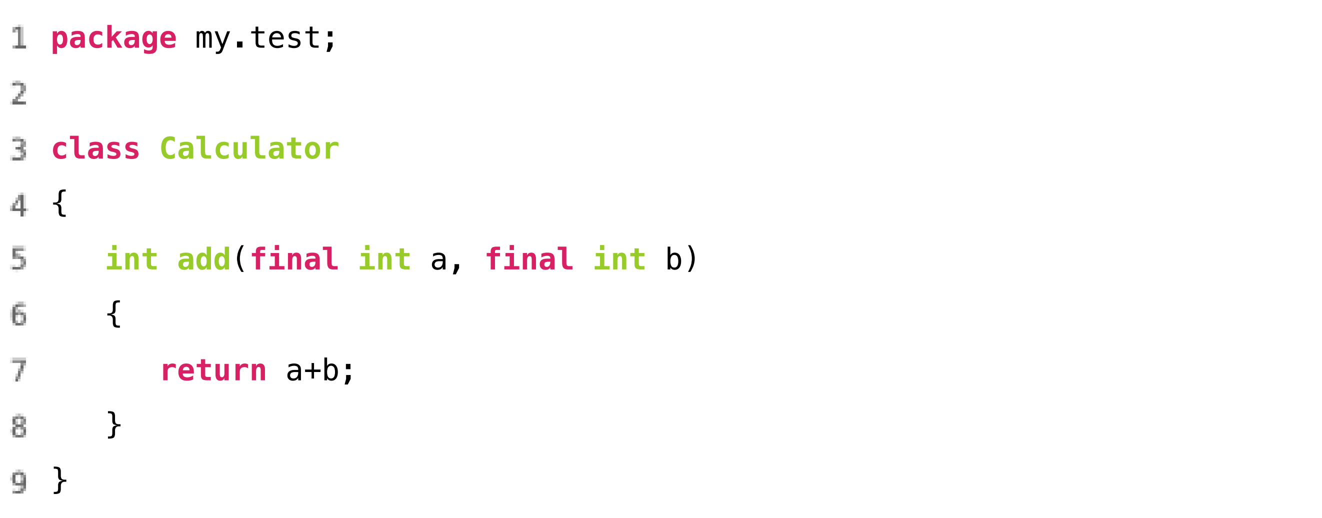
WHAT IS OPENREWR

A SHORT INTRODUCTION

- created and maintained by moderne
- framework and ecosystem for automa refactoring at scale
- prepackaged, open-source refactoring
- moderne platform/cli
    - free tier for open source projects

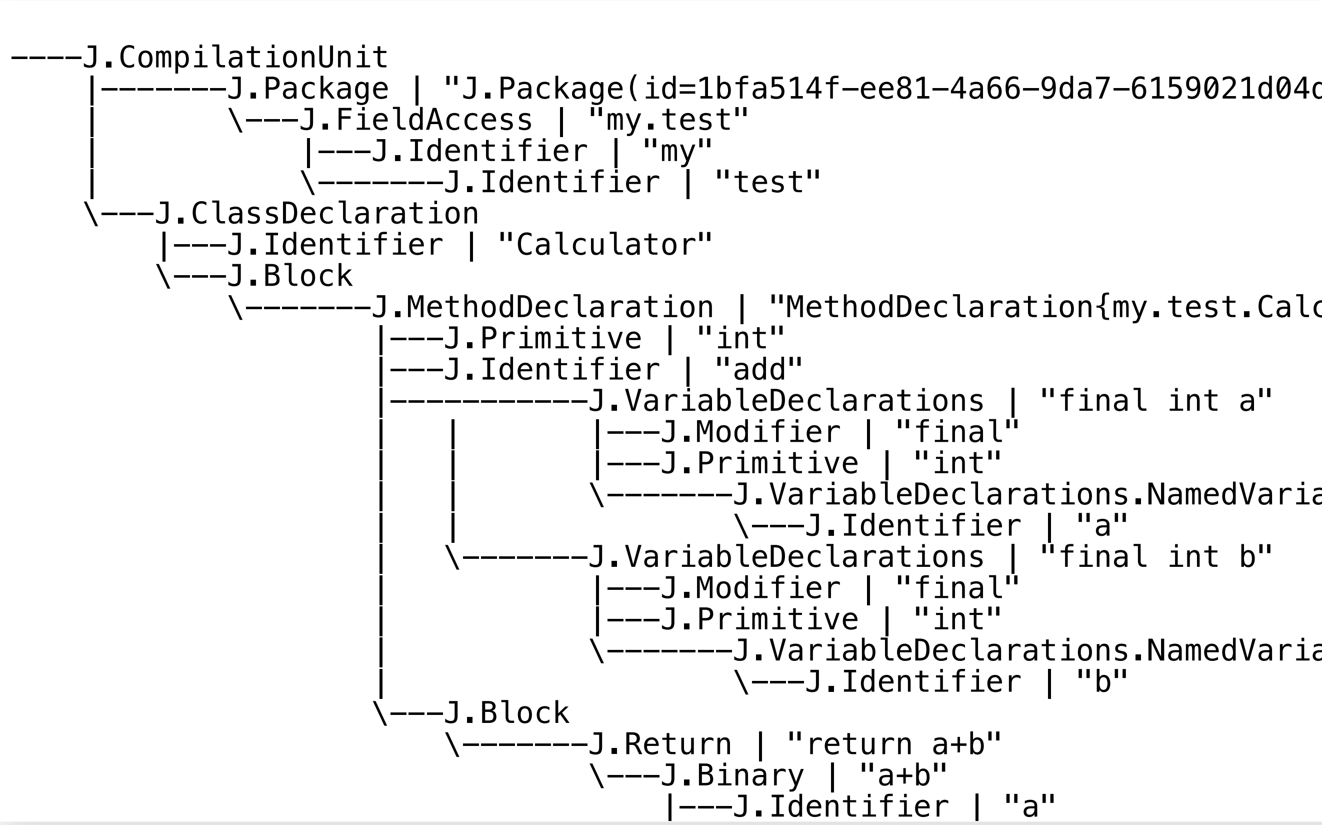
HOW DOES OPENREWRITE

- builds lossless semantic tree - LST
  - java
  - yml
  - xml
  - json
  - ...
- iterates recursively over the LST applyi
  transformations to the LST
- stops when no more changes are appli
- writes transformed LST back to the sou



```java
package my.test;

class Calculator
{
   int add(final int a, final int b)
   {
      return a+b;
   }
}
```



```
----J.CompilationUnit
    |-------J.Package | "J.Package(id=1bfa514f-ee81-4a66-9da7-6159021d04c
    |       \---J.FieldAccess | "my.test"
    |           |---J.Identifier | "my"
    |           \-------J.Identifier | "test"
    \---J.ClassDeclaration
        |---J.Identifier | "Calculator"
        \---J.Block
            \-------J.MethodDeclaration | "MethodDeclaration{my.test.Calc
                    |---J.Primitive | "int"
                    |---J.Identifier | "add"
                    |-----------J.VariableDeclarations | "final int a"
                    |   |       |---J.Modifier | "final"
                    |   |       |---J.Primitive | "int"
                    |   |       \-------J.VariableDeclarations.NamedVaria
                    |   |               \---J.Identifier | "a"
                    |   \-------J.VariableDeclarations | "final int b"
                    |           |---J.Modifier | "final"
                    |           |---J.Primitive | "int"
                    |           \-------J.VariableDeclarations.NamedVaria
                    |                   \---J.Identifier | "b"
                    \---J.Block
                        \-------J.Return | "return a+b"
                                \---J.Binary | "a+b"
                                    |---J.Identifier | "a"
```

WHAT IS A RECIPE

- a set of instructions on how and when
  the LST

HOW TO USE A REC

- build tool plugin
  - gradle
  - maven
- moderne cli tool
  - log in with Github or Bitbucket
  - free for public open source projects
  - for private projects a contract with m required

```
./gradlew rewriteRun       # actually apply the configured recipes
./gradlew rewriteDryRun    # apply the configured recipes and create
./gradlew rewriteDiscover # get a list of available recipes provide
```

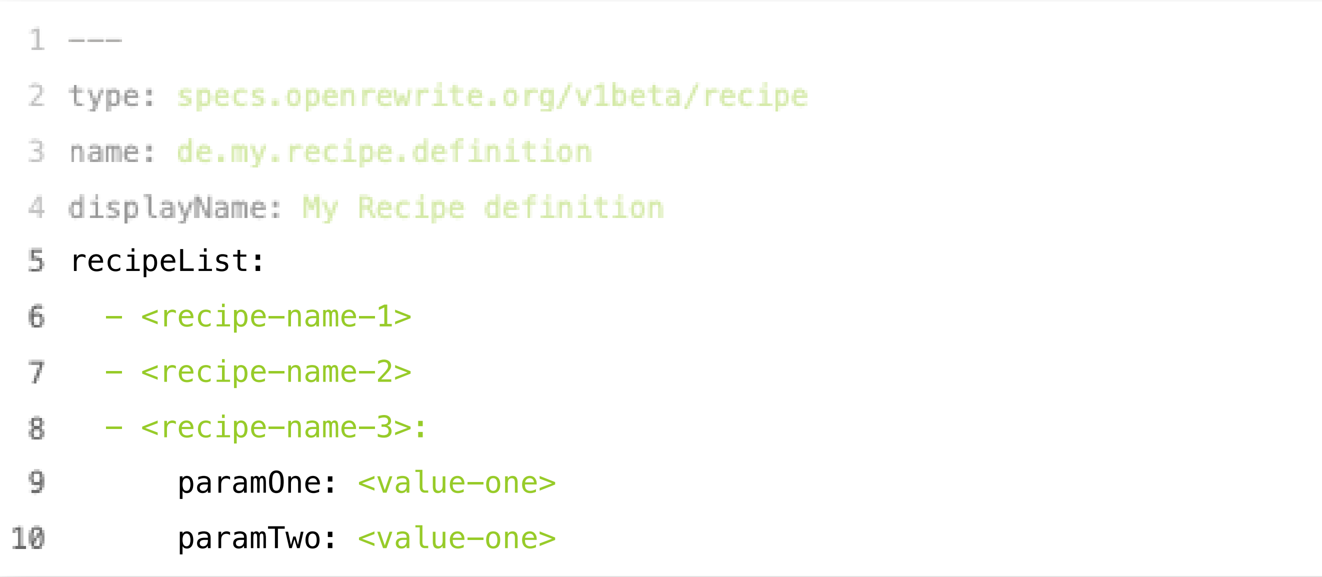
THE GRADLE PLUGIN

```
plugins {
    id("org.openrewrite.rewrite") version "6.11.2"
}

rewrite{
    activeRecipe("<recipe-name-1>")
    activeRecipe("<recipe-name-2>")
}

dependencies {
   rewrite("<rewrite-recipe-lib>")
}
```

THE REWRITE CONFIGURATI



```yaml
---
type: specs.openrewrite.org/v1beta/recipe
name: de.my.recipe.definition
displayName: My Recipe definition
recipeList:
  - <recipe-name-1>
  - <recipe-name-2>
  - <recipe-name-3>:
      paramOne: <value-one>
      paramTwo: <value-one>
```

THE REWRITE CONFIGURATI

- In the root of the project
- rewrite.yml
- activate it with the own name (here de.my.recipe.definition)

REWRITEDISCOVER

Finding recipes fast

```
./gradlew rewriteDiscover
```

ASSIGNMENT: 01_INTR

ASSIGNMENT: 02_SPRING_BOOT_UPGR

DO I NEED TO CONFIGURE A RECIPES SEPARATELY FOR A PROJECTS?

Of course not!

CREATING AN OPENREWRITE LIBRARY

HOW TO PACKAGE AN OWN LIBRARY

- A declarative library uses the same rewrite.yml recipe
- To use a recipe put the rewrite.yml in the root
- To package it put it in "src/main/resources/ME
- a recipe library is packaged as a normal java lib

ASSIGNMENT: 03_WRITE_DECLARATIVE_

WHAT IF THERE IS NO RECIPE A FOR MY REFACTORING

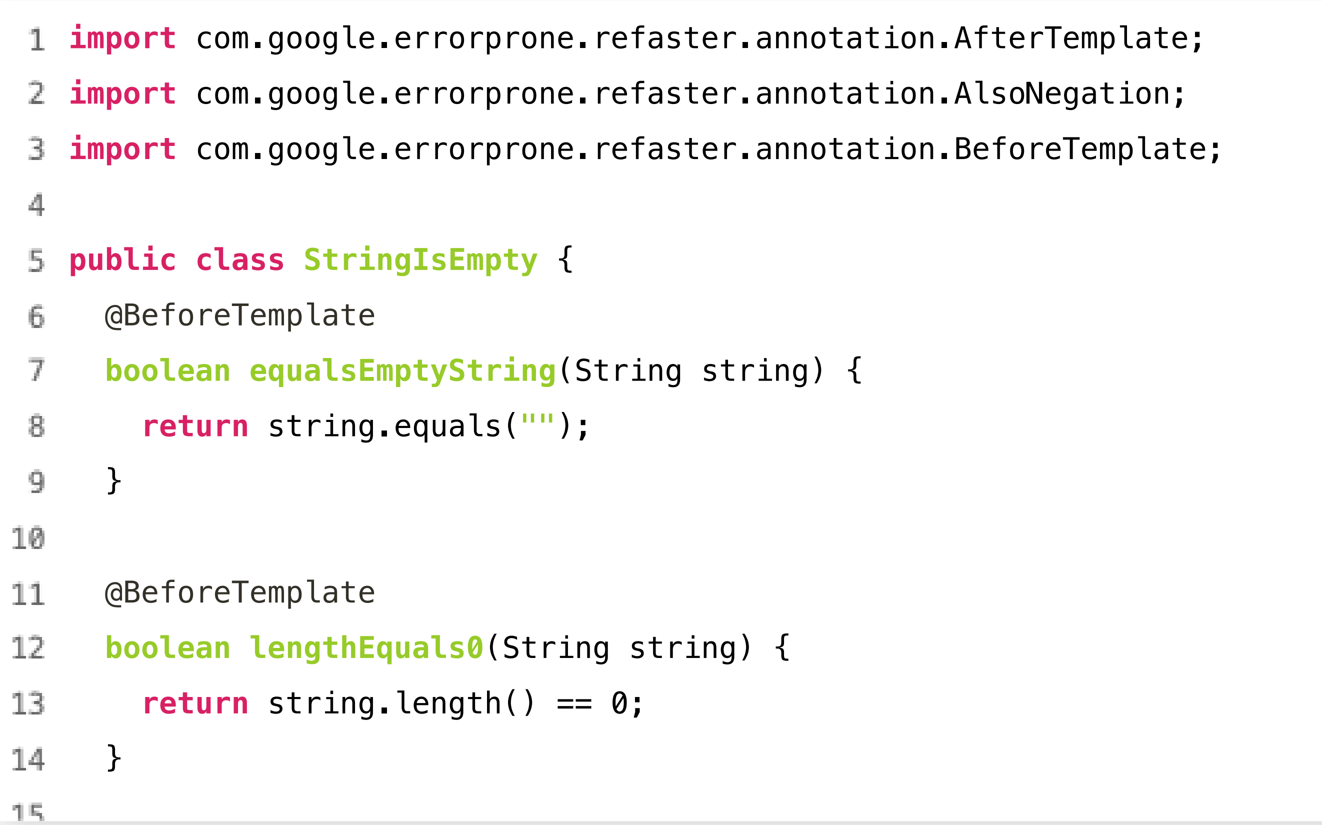
- utilize openrewrite powerful java api
  - requires deep understanding of the
  - a lot of code may be necessary even small refactorings
- alternative: refaster templates

  what is this?

A SMALL EXCURSUS TO GO
REFASTER TEMPLATE

- is part of the google Error Prone tool
- may be used for simple refactorings like
  - migrate uses of method A to method B
  - migrate use of method A where the argumer particular type to method B
  - migrate a particular fluent sequence of meth to some other pattern
  - migrate a sequence of consecutive statemer alternative

Source: Error Prone documentation



```java
import com.google.errorprone.refaster.annotation.AfterTemplate;
import com.google.errorprone.refaster.annotation.AlsoNegation;
import com.google.errorprone.refaster.annotation.BeforeTemplate;

public class StringIsEmpty {
  @BeforeTemplate
  boolean equalsEmptyString(String string) {
    return string.equals("");
  }

  @BeforeTemplate
  boolean lengthEquals0(String string) {
    return string.length() == 0;
  }

```

BEFORE TEMPLATE

- describes the code pattern that shoulc
- the parameter(s) represent any expres specified type
  - the string parameter in the example stands expression of type String

```
boolean equalsEmptyString(String string) {  // string => every e
    return string.equals("");
}
```

- can contain multiple lines to be replac
- for more advanced examples visit refas

AFTER TEMPLATE

- describes the desired pattern
- has the same arguments as the befo
- can contain multiple lines
- for more advanced examples visit re

ALSO NEGATION

- used to signal that the rule can also ma
  logical negation of the @BeforeTempla
- for more advanced examples visit refas

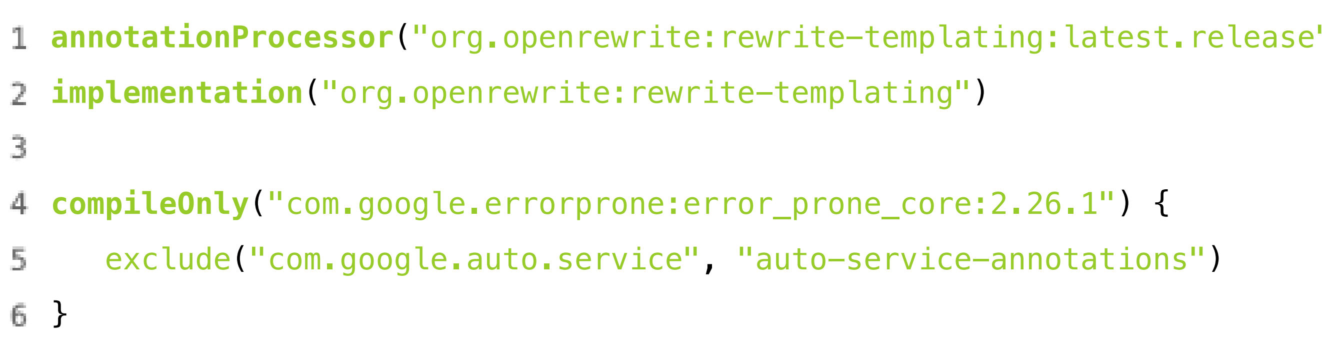
DOES OPENREWRITE USE ERROR PR

No!

A recipe is generated through an annotat by openrewrite

The final class ends with "Recipe" or

REQUIRED DEPENDENCIE
REFASTER



```
annotationProcessor("org.openrewrite:rewrite-templating:latest.release'
implementation("org.openrewrite:rewrite-templating")

compileOnly("com.google.errorprone:error_prone_core:2.26.1") {
    exclude("com.google.auto.service", "auto-service-annotations")
}
```

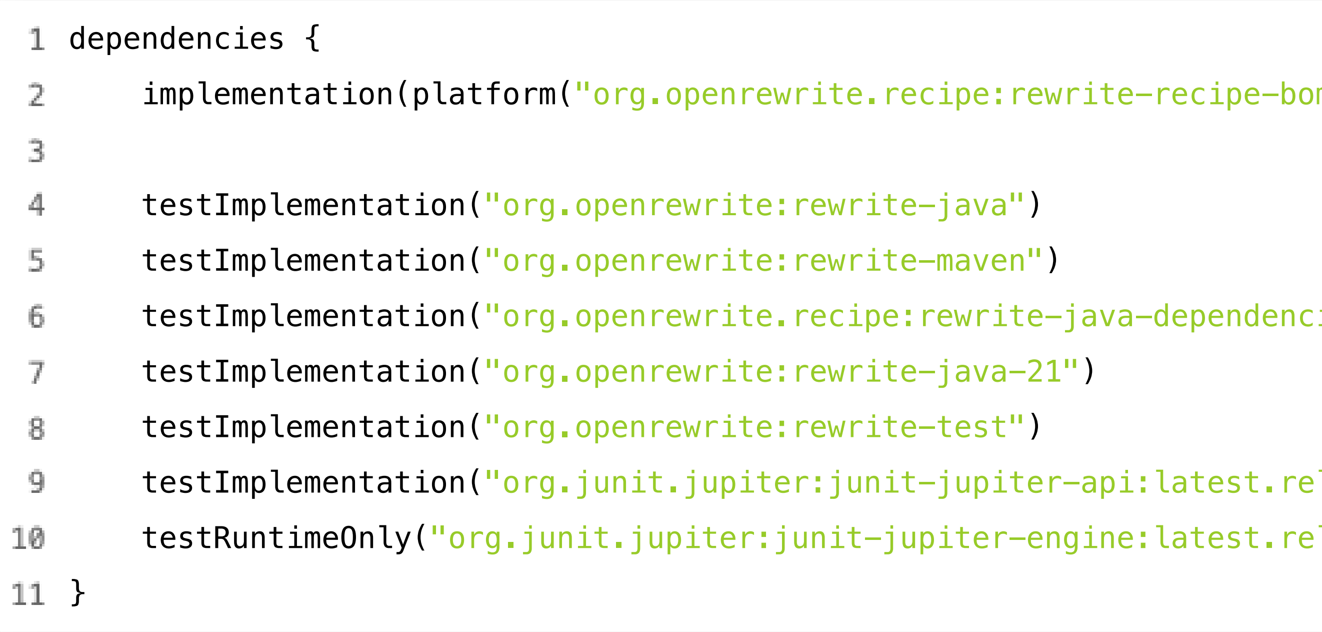
ASSIGNMENT:
04_CUSTOM_REFASTER_R

TESTING

TESTING

- support for writing unit tests
- supports different SourceSpecs (java, y etc...)
- can fine tune the test execution enviro applying a recipe
- tests can use newer version of Java tha (e.g. to make use of multiline strings)

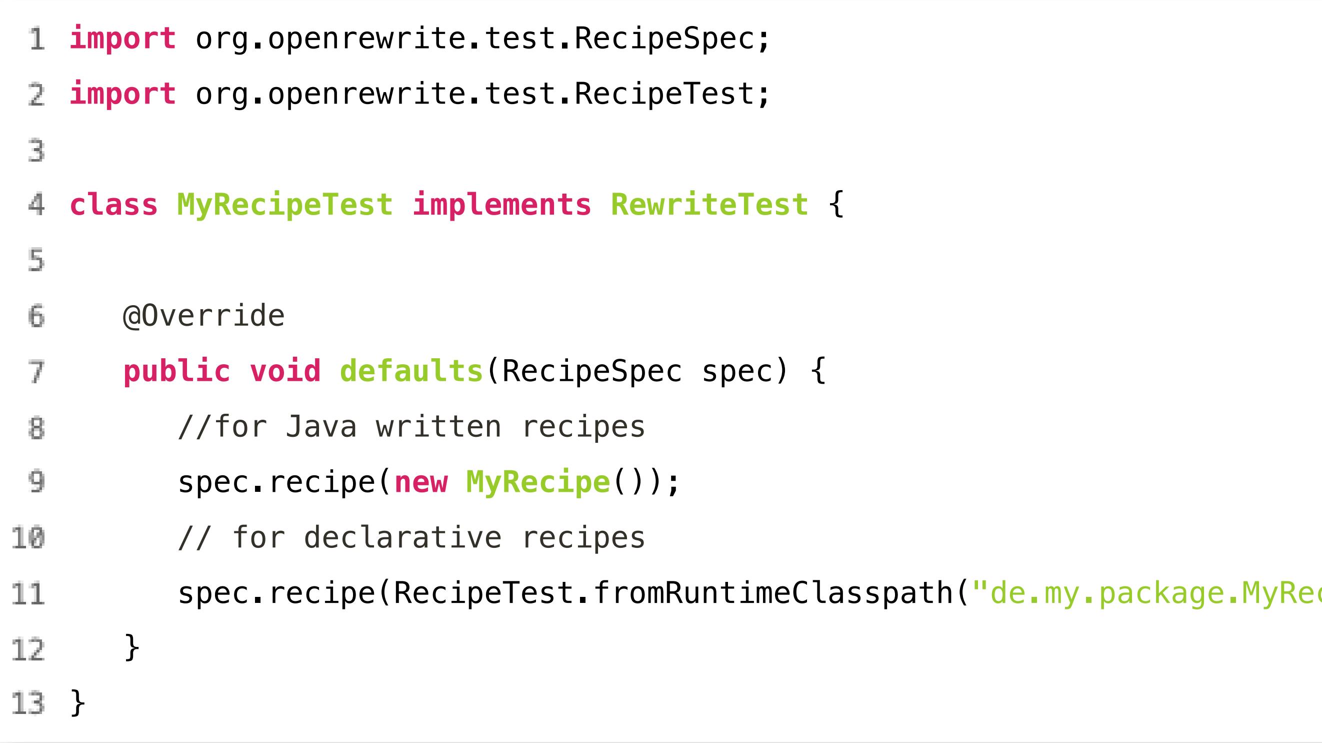
DEPENDENCIES



```
dependencies {
    implementation(platform("org.openrewrite.recipe:rewrite-recipe-bo
    
    testImplementation("org.openrewrite:rewrite-java")
    testImplementation("org.openrewrite:rewrite-maven")
    testImplementation("org.openrewrite.recipe:rewrite-java-dependenci
    testImplementation("org.openrewrite:rewrite-java-21")
    testImplementation("org.openrewrite:rewrite-test")
    testImplementation("org.junit.jupiter:junit-jupiter-api:latest.re
    testRuntimeOnly("org.junit.jupiter:junit-jupiter-engine:latest.re
}
```

}

TEST PREPARATION

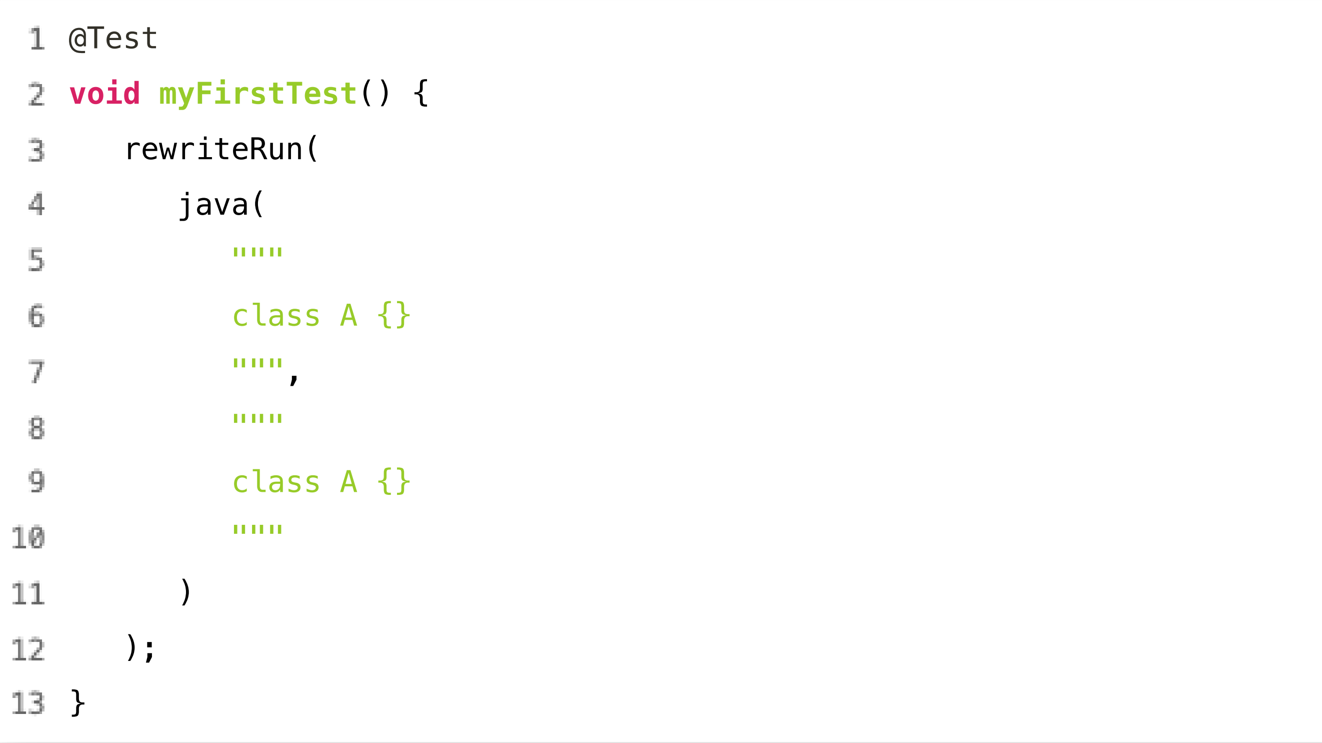


```java
import org.openrewrite.test.RecipeSpec;
import org.openrewrite.test.RecipeTest;

class MyRecipeTest implements RewriteTest {

   @Override
   public void defaults(RecipeSpec spec) {
      //for Java written recipes
      spec.recipe(new MyRecipe());
      // for declarative recipes
      spec.recipe(RecipeTest.fromRuntimeClasspath("de.my.package.MyRec
   }
}
```

FIRST TEST

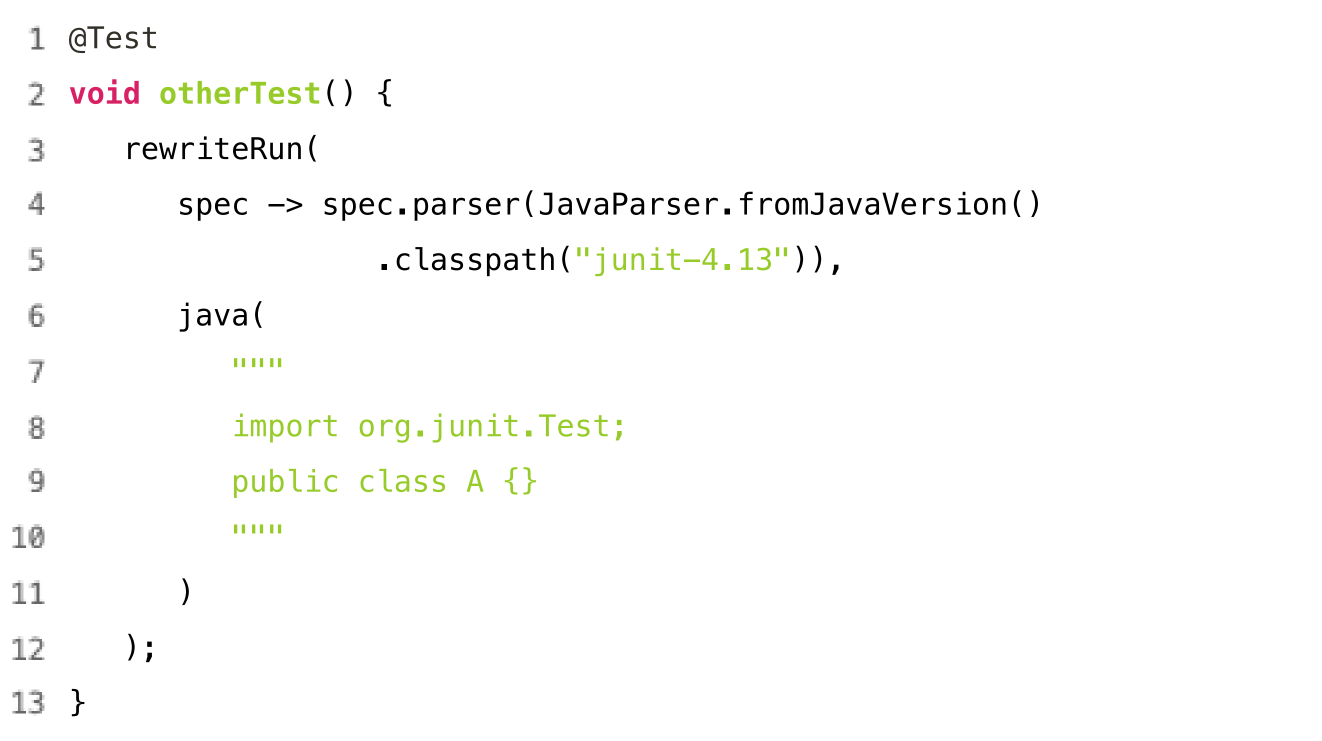


```java
@Test
void myFirstTest() {
   rewriteRun(
      java(
         """
         class A {}
         """,
         """
         class A {}
         """
      )
   );
}
```

REWRITERUN

- expects a list of SourceSpecs (here one
- SourceSpec content must be valid as it like the real source code
- SourceSpec describes a before and aft the recipe was executed
- second String can be omitted if no cha expected
- RecipeSpec can be changed for a test c adding a library to the classpath)

ADAPT RECIPESPEC



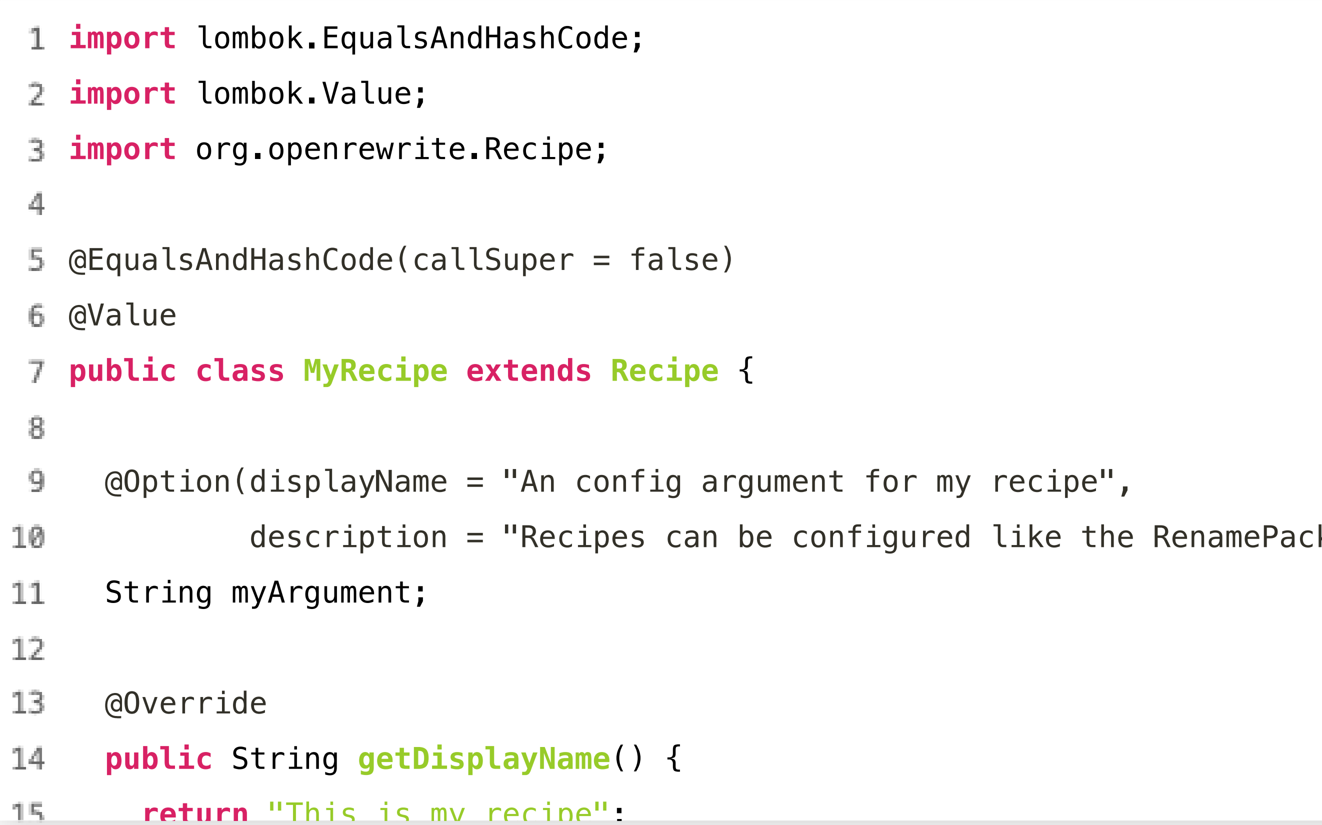
```
@Test
void otherTest() {
   rewriteRun(
      spec -> spec.parser(JavaParser.fromJavaVersion()
                 .classpath("junit-4.13")),
      java(
         """
         import org.junit.Test;
         public class A {}
         """
      )
   );
}
```

ASSIGNMENT: 05_TEST_R

GETTING OUR HANDS DI

Writing our own recipe with the open

RECIPE CLASS



```java
import lombok.EqualsAndHashCode;
import lombok.Value;
import org.openrewrite.Recipe;

@EqualsAndHashCode(callSuper = false)
@Value
public class MyRecipe extends Recipe {

  @Option(displayName = "An config argument for my recipe",
          description = "Recipes can be configured like the RenamePac
  String myArgument;

  @Override
  public String getDisplayName() {
    return "This is my recipe";
```

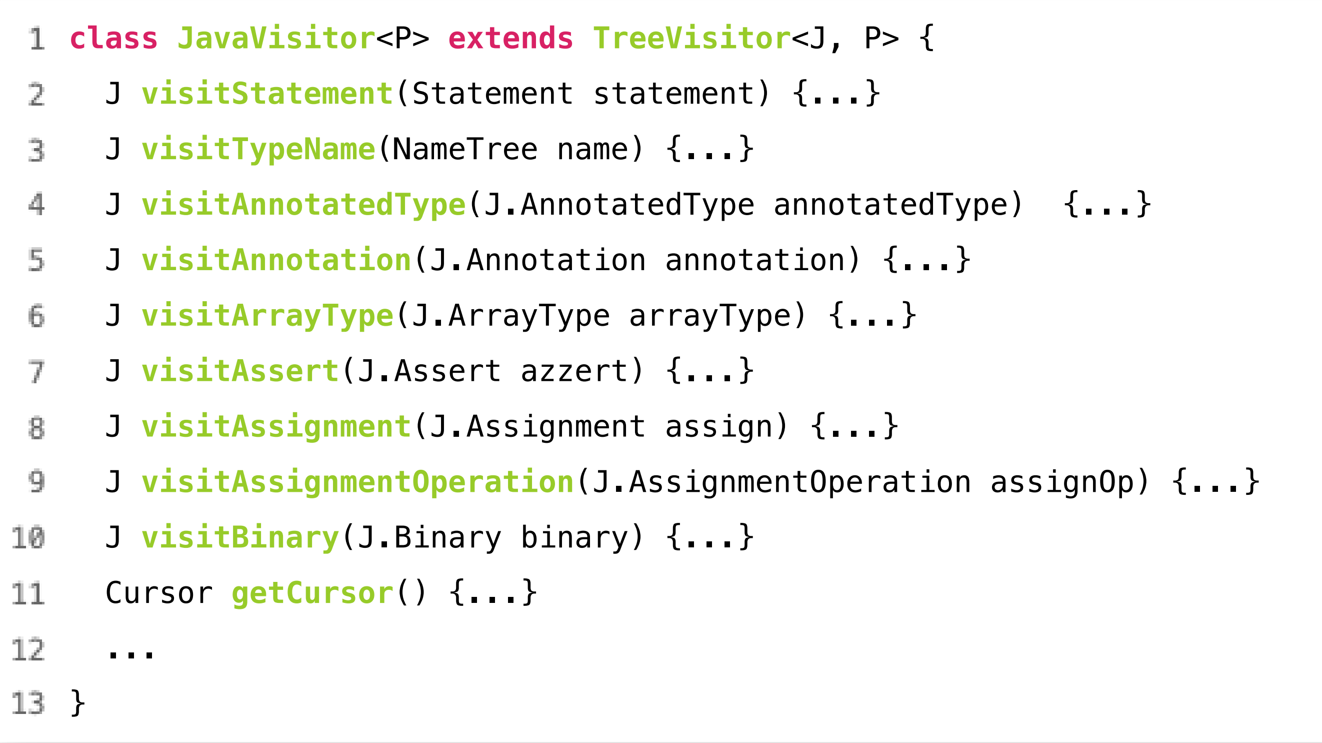
RECIPE

- Defines the configuration of the recipe
- Can have optional arguments
- Defines information that will be displa doing rewriteDiscover
- Defines a visitor to traverse the code ar changes
- Has to be serializable

DIFFERENT VISITORS

- TreeVisitor (abstract base cl
- JavaIsoVisitor
- MavenVisitor
- PlainTextVisitor
- YamlIsoVisitor
- XmlIsoVisitor
- ...

JAVAVISITOR



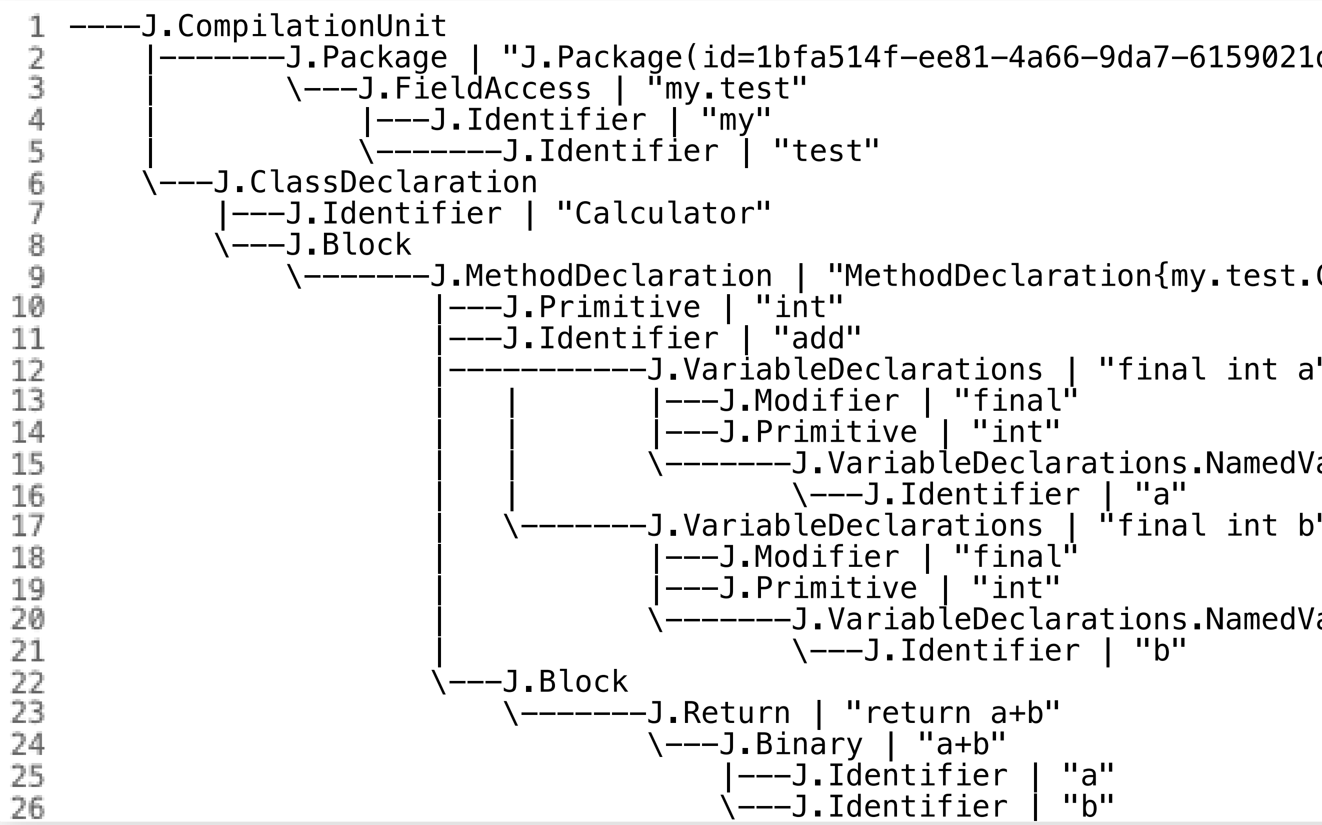
```java
class JavaVisitor<P> extends TreeVisitor<J, P> {
  J visitStatement(Statement statement) {...}
  J visitTypeName(NameTree name) {...}
  J visitAnnotatedType(J.AnnotatedType annotatedType)  {...}
  J visitAnnotation(J.Annotation annotation) {...}
  J visitArrayType(J.ArrayType arrayType) {...}
  J visitAssert(J.Assert azzert) {...}
  J visitAssignment(J.Assignment assign) {...}
  J visitAssignmentOperation(J.AssignmentOperation assignOp) {...}
  J visitBinary(J.Binary binary) {...}
  Cursor getCursor() {...}
  ...
}
```

VISITOR PATTERN

- Visitor will be called when ever travers
  matching block in the LST
- visit methods run independently and w
  by OpenRewrite
- Visit methods available on all level of t
  CompilationUnit to single statement)
- Visit methods return for isomorphic Vis
  same type of LST element as visited

DEBUGGING

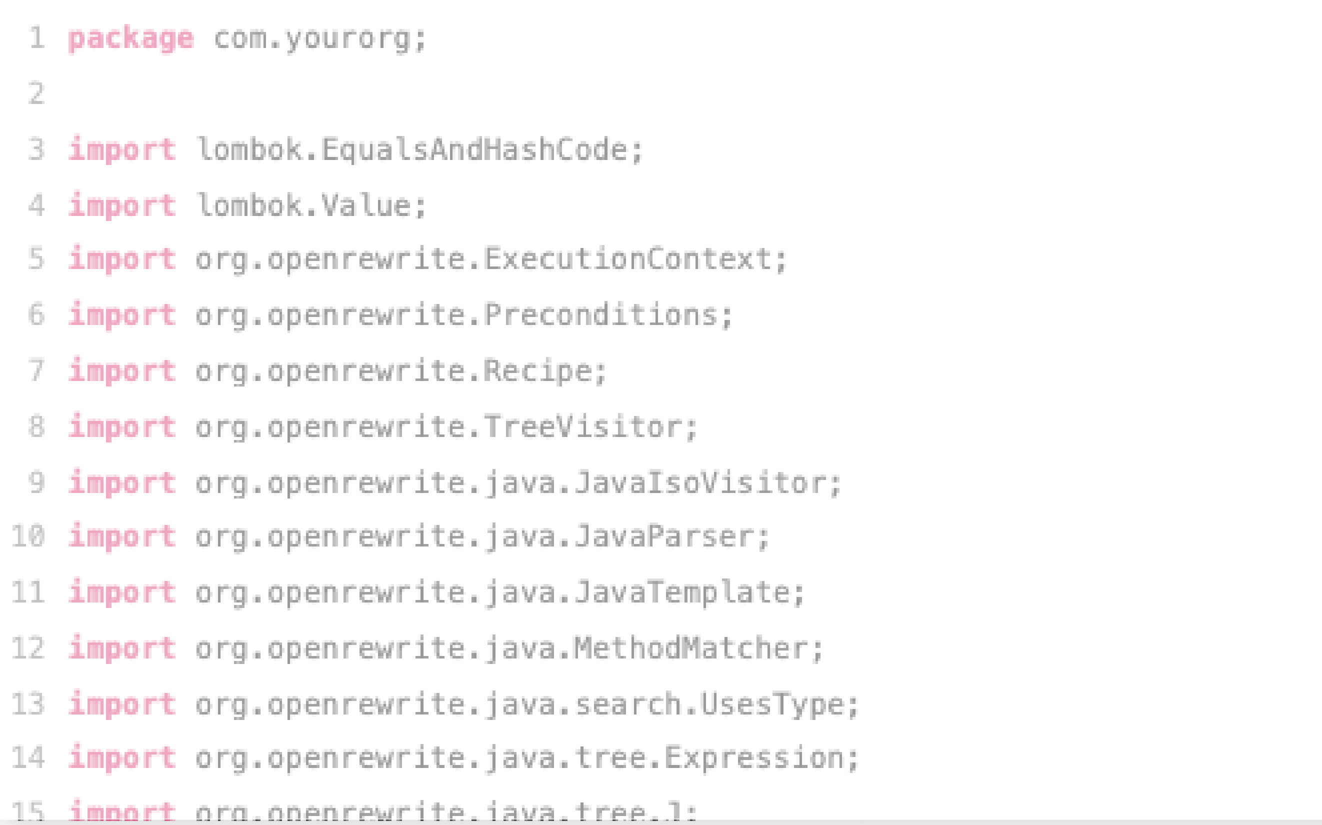
```
System.out.println(TreeVisitingPrinter.printTree(getCursor()));
```



```
----J.CompilationUnit
    |-------J.Package | "J.Package(id=1bfa514f-ee81-4a66-9da7-6159021c
    |       \---J.FieldAccess | "my.test"
    |           |---J.Identifier | "my"
    |           \-------J.Identifier | "test"
    \---J.ClassDeclaration
        |---J.Identifier | "Calculator"
        \---J.Block
            \-------J.MethodDeclaration | "MethodDeclaration{my.test.
                    |---J.Primitive | "int"
                    |---J.Identifier | "add"
                    |-----------J.VariableDeclarations | "final int a
                    |   |       |---J.Modifier | "final"
                    |   |       |---J.Primitive | "int"
                    |   |       \-------J.VariableDeclarations.NamedVa
                    |   |               \---J.Identifier | "a"
                    |   \-------J.VariableDeclarations | "final int b
                    |           |---J.Modifier | "final"
                    |           |---J.Primitive | "int"
                    |           \-------J.VariableDeclarations.NamedVa
                    |                   \---J.Identifier | "b"
                    \---J.Block
                        \-------J.Return | "return a+b"
                                \---J.Binary | "a+b"
                                    |---J.Identifier | "a"
                                    \---J.Identifier | "b"
```

BUT HOW TO START?

Correct! The openrewrite recipe s



```java
package com.yourorg;

import lombok.EqualsAndHashCode;
import lombok.Value;
import org.openrewrite.ExecutionContext;
import org.openrewrite.Preconditions;
import org.openrewrite.Recipe;
import org.openrewrite.TreeVisitor;
import org.openrewrite.java.JavaIsoVisitor;
import org.openrewrite.java.JavaParser;
import org.openrewrite.java.JavaTemplate;
import org.openrewrite.java.MethodMatcher;
import org.openrewrite.java.search.UsesType;
import org.openrewrite.java.tree.Expression;
import org.openrewrite.java.tree.J;
```

Source: https://github.com/moderneinc/rewrite-recipe-starter

THE CURSOR

- keeps track of a visitor's position withi
- used to access parent or sibling LSTs
- discarded if visiting is complete
- contains map to store and share data b methods
- organized as stack

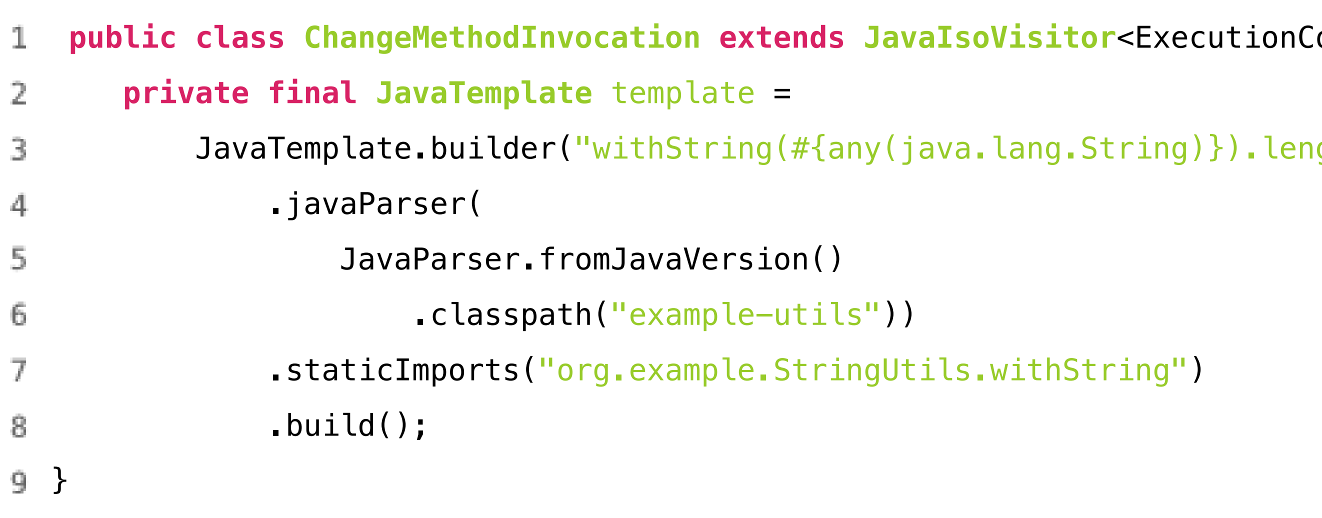
CURSOR EXAMPLES

```
getCursor().putMessageOnFirstEnclosing(J.ClassDeclaration.class, "FOUND_M
...
getCursor().pollMessage("FOUND_METHOD"); // removes message from cursor
...
getCursor().getMessage("FOUND_METHOD");  // leaves message on cursor
...
getCursor().getNearestMessage("FOUND_METHOD");
```

```
getCursor().getParentOrThrow()
```

ASSIGNMENT: 07_APPENDIX_CUSTOM_RECIPE_OPENRE

JAVA TEMPLATES



```
 public class ChangeMethodInvocation extends JavaIsoVisitor<ExecutionC
    private final JavaTemplate template =
        JavaTemplate.builder("withString(#{any(java.lang.String)}).leng
            .javaParser(
                JavaParser.fromJavaVersion()
                    .classpath("example-utils"))
            .staticImports("org.example.StringUtils.withString")
            .build();
}
```

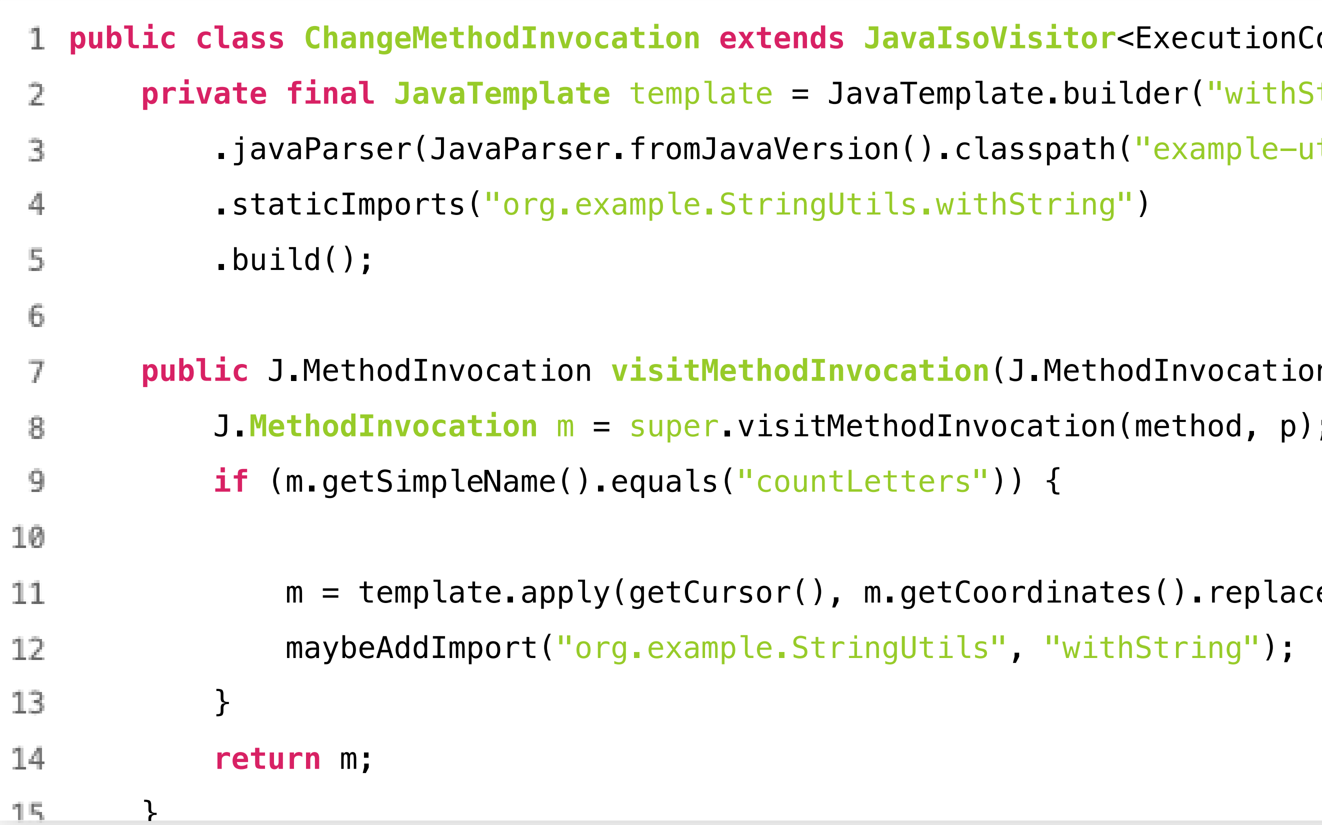
JAVA TEMPLATES

- Generates code (LST elements) based o
  template
- String must be syntical correct
- ensures correct formatting
- can be applied on elements in the visit
- able to reference symbols
- can add needed imports for code snipp

ADD IMPORTS

```
JavaTemplate.builder("new SecureRandom()")
     .imports("java.security.SecureRandom")
     .build();
```

APPLY TO LST



```java
public class ChangeMethodInvocation extends JavaIsoVisitor<ExecutionCo
    private final JavaTemplate template = JavaTemplate.builder("withS
        .javaParser(JavaParser.fromJavaVersion().classpath("example-u
        .staticImports("org.example.StringUtils.withString")
        .build();

    public J.MethodInvocation visitMethodInvocation(J.MethodInvocatio
        J.MethodInvocation m = super.visitMethodInvocation(method, p);
        if (m.getSimpleName().equals("countLetters")) {

            m = template.apply(getCursor(), m.getCoordinates().replace
            maybeAddImport("org.example.StringUtils", "withString");
        }
        return m;
    }
```

WRAP UP AND OUTLOO

when should I use openrewrite

- medium to large code base
- refactoring affects numerous files
- framework and library updates

how should I use openrewrite

- prefer existing recipes
- prefer declarative recipes
- try to achieve what you need with refaster tem
- in other words: try to avoid writing your own re

WRAP UP AND OUTLOO

you need a custom imperative recipe?

- use the starter
- read the docs
- use the moderne you tube channel
- ask for help in the openrewrite slack channel

THANK YOU!