DevOps & Continuous Testing - intro

G. Molines

2017-2018

Université Nice Sophia Antipolis

POLYTECH NICE-SOPHIA

INTRODUCTION

Logistics

- Class starts at on time
- Cell phone switched off
- **One** conversation at a time

Teaching Team

Guilhem Molines

guilhem.molines@unice.fr

Philippe Collet

Philippe.Collet@unice.fr

Communication

- Discussion: Email and Slack
- Instructions and deliveries: Github + Bitbucket

DELIVERING SOFTWARE

Delivering software…

- In 1995, MS released Windows 95
  - Major breakthrough for consumers
  - Real multi-tasking with background apps
- How many service packs were released?

Delivering software…

- In 1995, MS released Windows 95
  - Major breakthrough for consumers
  - Real multi-tasking with background apps

- How many service packs were released?

→ 2 (yes, two)

(6 months and 1 year after release)

Delivering software…

- In 2015, MS released Windows 10
  - 4 editions
  - Continuous, forced updates

- How many service packs were released?

Delivering software…

- In 2015, MS released Windows 10
  - 4 editions
  - Continuous, forced updates

- How many service packs were released?

→ couldn’t count…

(1 GB on first day)

What is the most important thing you need to deliver software continuously?

Trust

How do you get trust in your software delivery?

- Good architecture
- Development guidelines
- Project management
- Controlled processes
- Traceability of requirements
- Automation
- Repeatable **pipeline**
- Testing
- Testing
- And…

Testing

How do you build a software delivery pipeline?

How do you write good tests?

SDLC IN A NUTSHELL

Single student, simple project

- Ant
- Maven?
- SCM?
- Code and click

Group of students, simple project

- Ant / Maven
- Github
- Code and click
- Build on local machine
- Ticket management ?

Industrial complex project

- 20 – 200 contributors
- One release / year
- One patch / month
- > 20M LOCs
- Deployment complexity

- → What do you need?

Industrial complex project

- Componentization
- Independence of builds
- Each component tested
- Clear quality indicators
- Requirement traceability
- Fast builds
- Each contributor only builds what they code
- Deployment automation

SDLC

Software Delivery Life Cycle

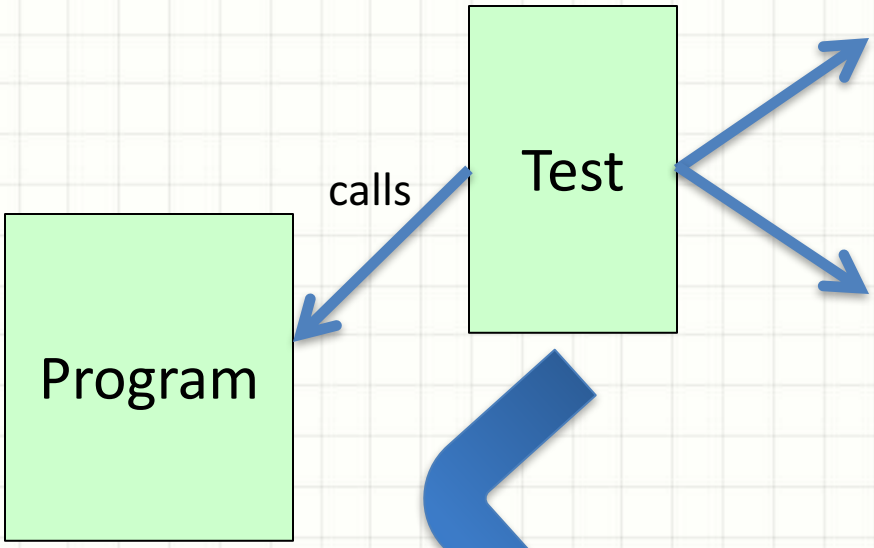
- All the tools, processes, ways of collaborating to build, deliver, deploy and maintain software matching users' expectations

TD part 2

- This is what we will learn and build together

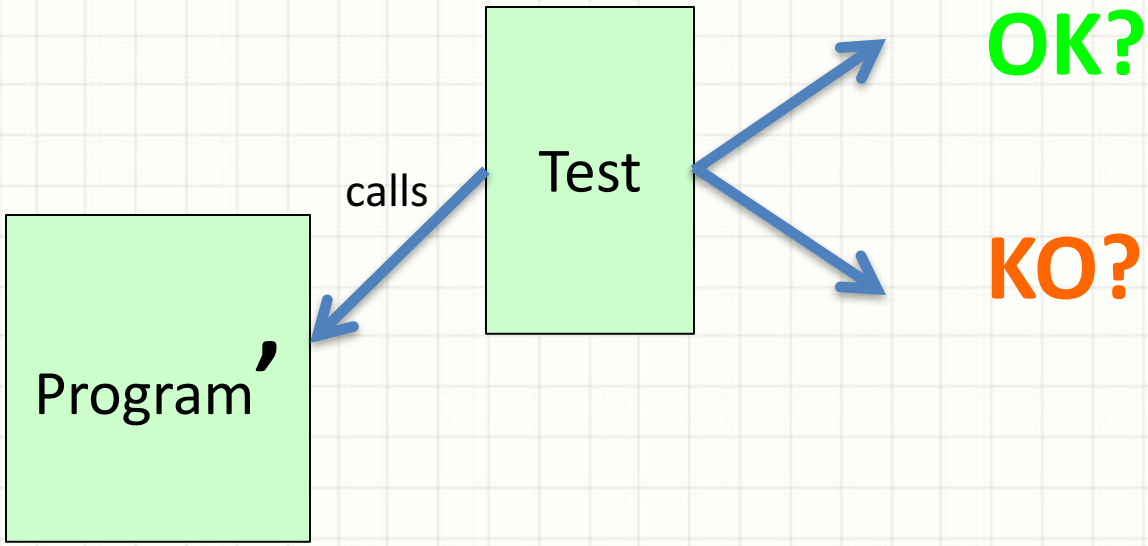
MUTANTS

How do you test a program?



Call Program function
Assert (returned value == expected value)
Result

How do you know your test is good?



Changing a program?

- What types of changes:
  - Variable set to another value
  - Condition meaning:
    - == → =
    - > → <
  - For loop: change range
  - Inject randomness
  - Etc.

For each change

- Program may not compile
- Test can crash
- Test can lead to same result
- Or different result

- → find “good” mutants

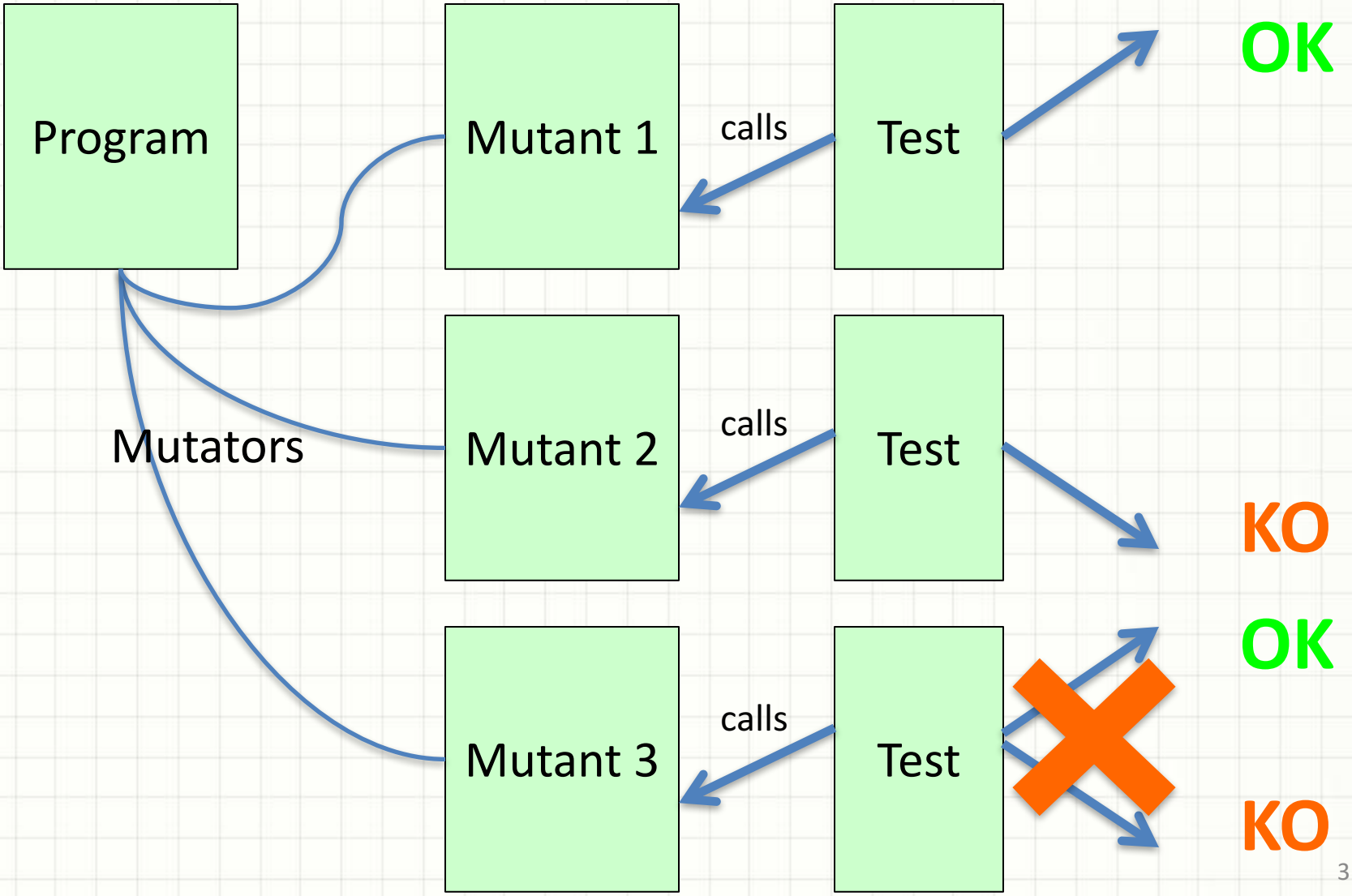
Automate changes

- Reflection
- Utility library: Spoon
- Define directory structures
- Automation script to launch mutants

Assess results

- Generate test report for each mutant
- Compare test reports across mutants

General view



TD part 1

- Starts now!

- Groups of 4
- Code on github
- Deliverable on Feb. 28th

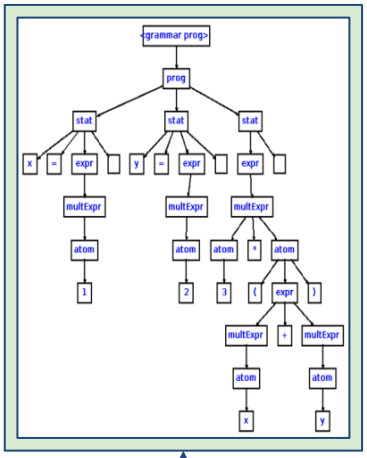
SPOON

What do we need?

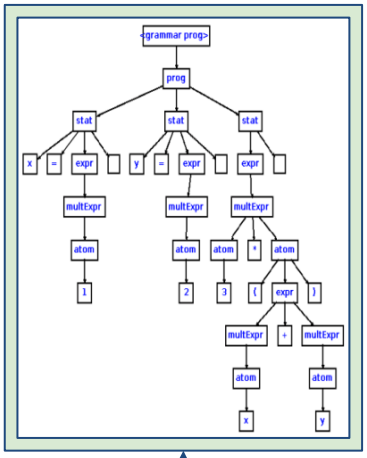
Query code
To potentially
change it
analyse it
by **adding** / **removing** / **updating** elements


SPOON
a library to analyze, rewrite, transform, transpile
Java source code


Fichier
java
:in

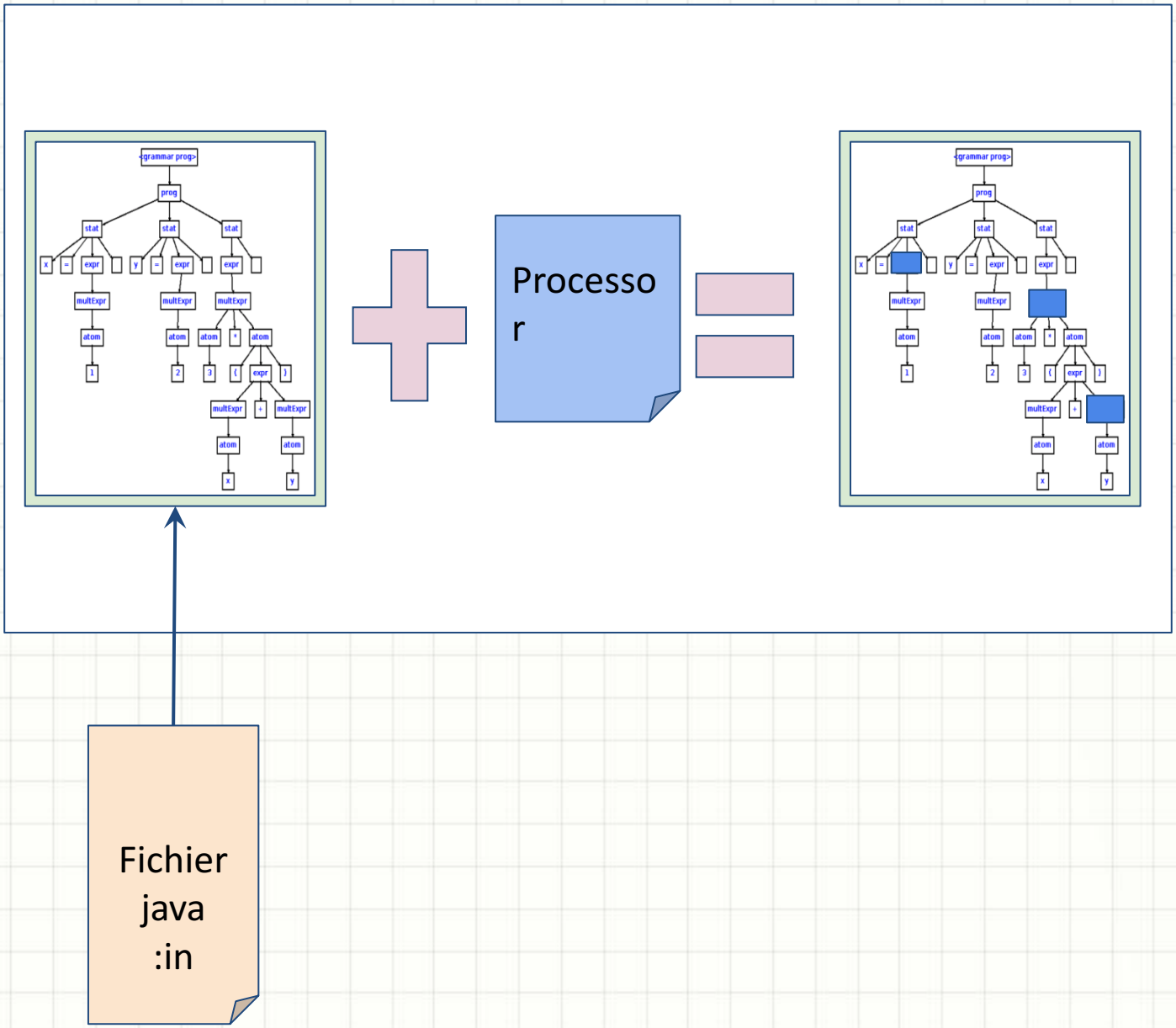


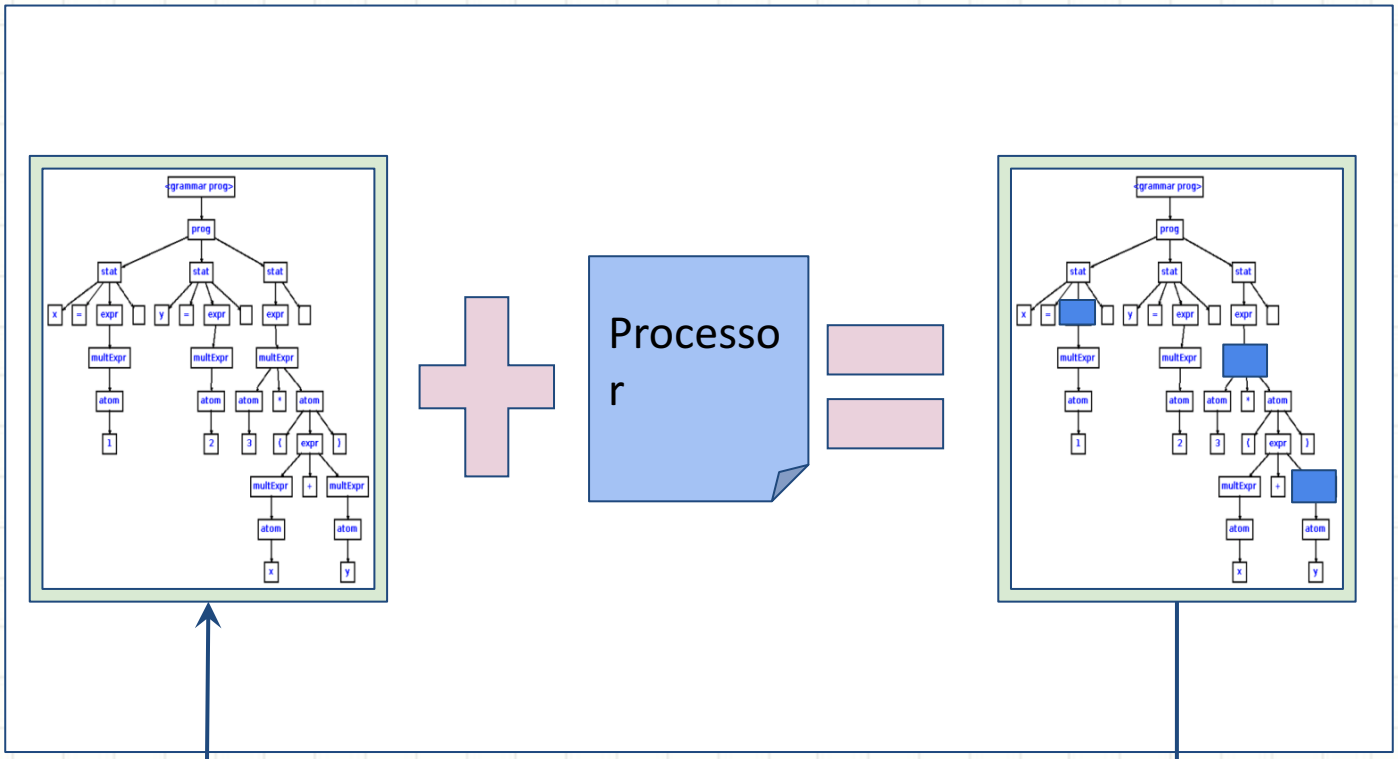
Fichier
java
:in



Processor

Fichier java :in


Processo
r
Fichier
java
:in



Fichier
java
:in

Fichier
java
:out

Out?

How and where Spoon generates result?

Maven lifecycle :

- `validate`
- `compile`
- `test`
- `package`
- `verify`
- `install`
- `deploy`

real lifecycle more detailed here :
https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

Maven lifecycle :

- `validate`
- `compile`
- `test`
- `package`
- `verify`
- `install`
- `deploy`

Spoon triggered here (*generate-sources*)

real lifecycle more detailed here : https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

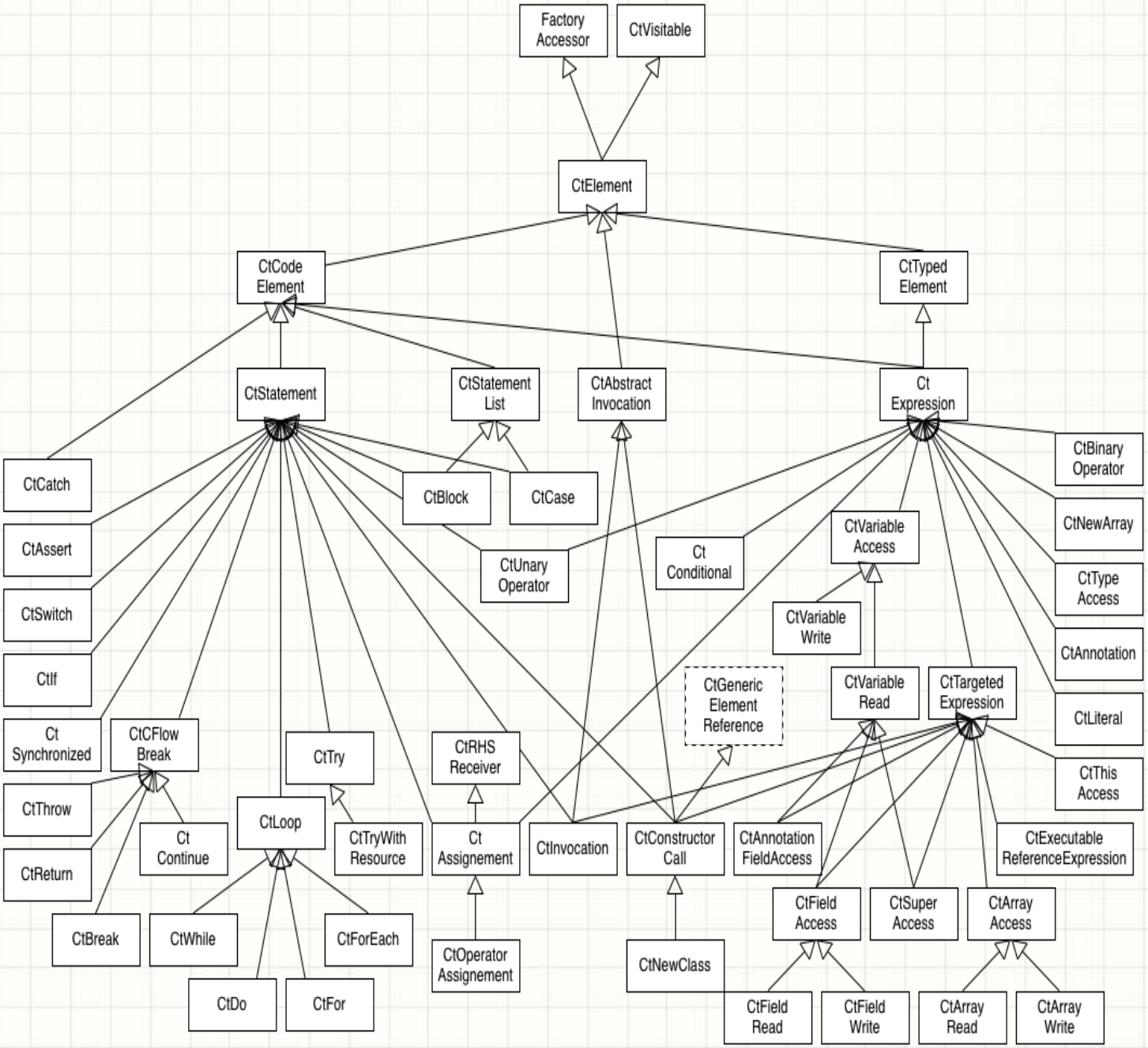
How to write processor?

```
public class AProcessor extends
AbstractProcessor<TARGET_OF_PROCESSING> {
  @Override
  public void process(TARGET_OF_PROCESSING object) {
            // ...
  }
}
```

TARGET_OF_PROCESSING ?

How to write processor?

```
public class CatchProcessor extends AbstractProcessor<CtCatch> {
  public void process(CtCatch element) {
     if (element.getBody().getStatements().size() == 0) {
        getFactory().getEnvironment().report(this, Level.WARN, element, "empty
catch clause");
     }
  }
}
```


Factory Accessor
CtVisitable
CtElement
CtCode Element
CtTyped Element
CtStatement
CtStatement List
CtAbstract Invocation
Ct Expression
CtCatch
CtAssert
CtSwitch
CtIf
Ct Synchronized
CtCFlow Break
CtThrow
CtReturn
CtBreak
Ct Continue
CtWhile
CtLoop
CtDo
CtFor
CtForEach
CtTry
CtTryWith Resource
CtBlock
CtCase
CtUnary Operator
CtRHS Receiver
Ct Assignement
CtOperator Assignement
CtInvocation
CtConstructor Call
CtNewClass
Ct Conditional
CtGeneric Element Reference
CtAnnotation FieldAccess
CtVariable Access
CtVariable Write
CtVariable Read
CtTargeted Expression
CtFieldAccess
CtSuper Access
CtArray Access
CtField Read
CtField Write
CtArray Read
CtArray Write
CtExecutable ReferenceExpression
CtThis Access
CtLiteral
CtAnnotation
CtType Access
CtNewArray
CtBinary Operator

Plugin

```
<plugin>
  <groupId>fr.inria.gforge.spoon</groupId>
  <artifactId>spoon-maven-plugin</artifactId>
  <version>2.2</version>
  <executions>
    <execution>
      <phase>generate-sources</phase>
      <goals>
        <goal>generate</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <processors>
      <processor>fr.inria.gforge.spoon.processors.CatchProcessor</processor>
            </processors>
  </configuration>
</plugin>
```

Dependency

```
<dependency>
<groupId>fr.inria.gforge.spoon</groupId>
    <artifactId>spoon-core</artifactId>
    <version>5.5.0</version>
</dependency>
```

http://spoon.gforge.inria.fr/index.html

https://github.com/INRIA/spoon

QUESTIONS?

Next

- This week:
  - TD: Mutants 1
- Testing
  - TD: Mutants 2

APPENDIX