STATIC TEST TECHNIQUES

TEST

PBA SOFTWAREUDVIKLING/
BSC SOFTWARE DEVELOPMENT

Christian Nielsen cnls@cphbusiness.dk

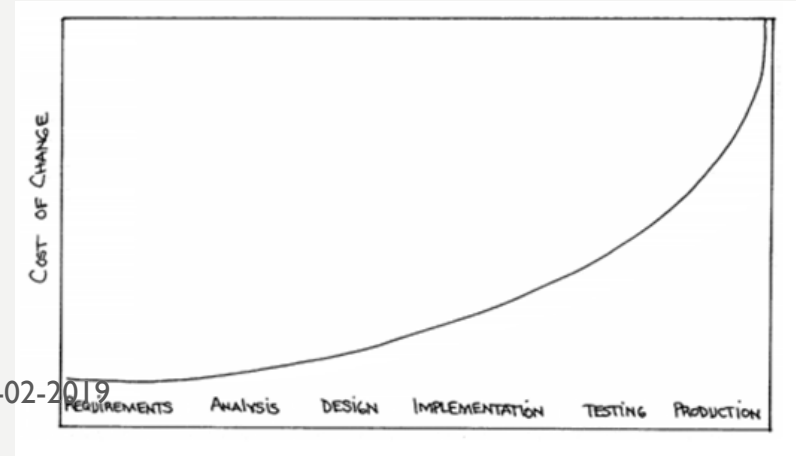
Tine Marbjerg tm@cphbusiness.dk

SPRING 2019

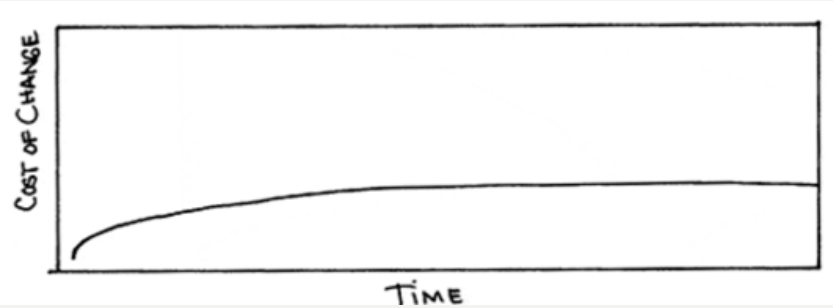
ECONOMY AND TEST

- Cost/benefit
  - We want to find as **many faults** as possible with as **little effort** as possible, i.e. cheapest ☺
- Static testing is often done before dynamic testing
  - keeps rework costs low as faults are detected at early stage
- General assumptions of cost of change in development:

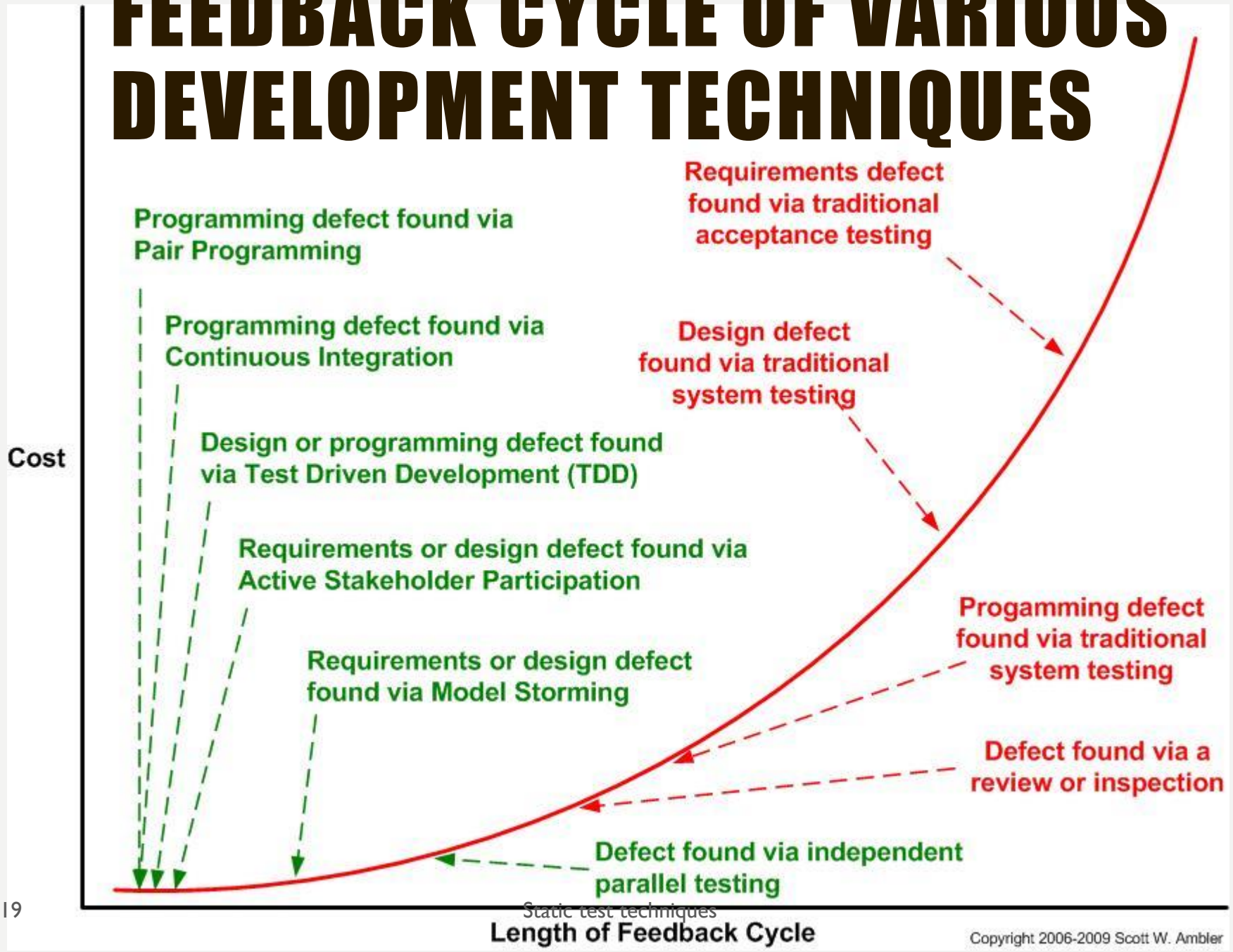
Traditional view



Agile view



FEEDBACK CYCLE OF VARIOUS DEVELOPMENT TECHNIQUES



REVIEWS

Formal reviews
Well structured
Regulated

Informal reviews
No documented procedure
More casual

- The formality is related to factors such as
  - Maturity of the development process
  - Legal or regulatory requirements

- Reviews often present milestones

TYPES OF REVIEWS

- Walkthrough
- Technical review
- Inspection

PULL REQUESTS

- Create a pull request to **propose** and **collaborate** on changes to a repository. These changes are proposed in a *branch*, which ensures that the master branch only contains finished and approved work.
- Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub.
- Once a pull request is opened, you can **discuss** and **review** the potential changes with collaborators and add follow-up commits before your changes are merged into the base branch.

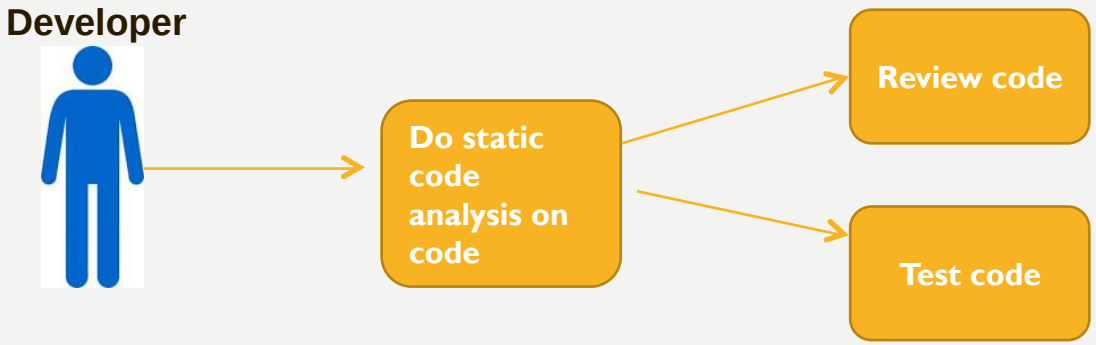
https://help.github.com/articles/about-pull-requests/

STATIC CODE ANALYSIS

- Coding standards
- Code metrics

Static analysis can be performed on requirements and design artifacts, but most tools focus on software code, so **our focus** will be on **static code analysis**

- **Static code analysis is preferably done by the developers *before* reviews and test activities**
- **Can also be done automatically as part of deployment pipeline**



WHAT'S WRONG WITH THIS CODE?

```
...

public class DemoCodingStandards {

   private void findAll() {
        DB db = new DB();
        ArrayList list = db.retrieve();
        int size = list.size();
        System.out.println("Number of records retrieved " + size);
    }

    public static void main(String[] args) {
        DemoCodingStandards demo = new DemoCodingStandards();
        demo.findAll();
    }
}
```

CODING STANDARDS

Tool can check for adherence to coding standards

Examples of coding standards document

1. https://google.github.io/styleguide/javaguide.html

2.
   1. Use the Sun code conventions by default: http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html
   2. Never catch exceptions without logging the stack trace or rethrowing.
   3. Use dependency injection to decouple classes from each other
   4. Avoid abbreviations unless well-known e.g. DTO
   5. Methods that return Collections or arrays should not return null. Return empty collections and arrays instead of null
   6. …

CHECKSTYLE – CODING STANDARD TOOL

Checkstyle

- is highly configurable tool that can be made to support almost any coding standard
- Example of report: http://maven.apache.org/plugins/maven-checkstyle-plugin/checkstyle.html

Apache Maven Checkstyle Plugin

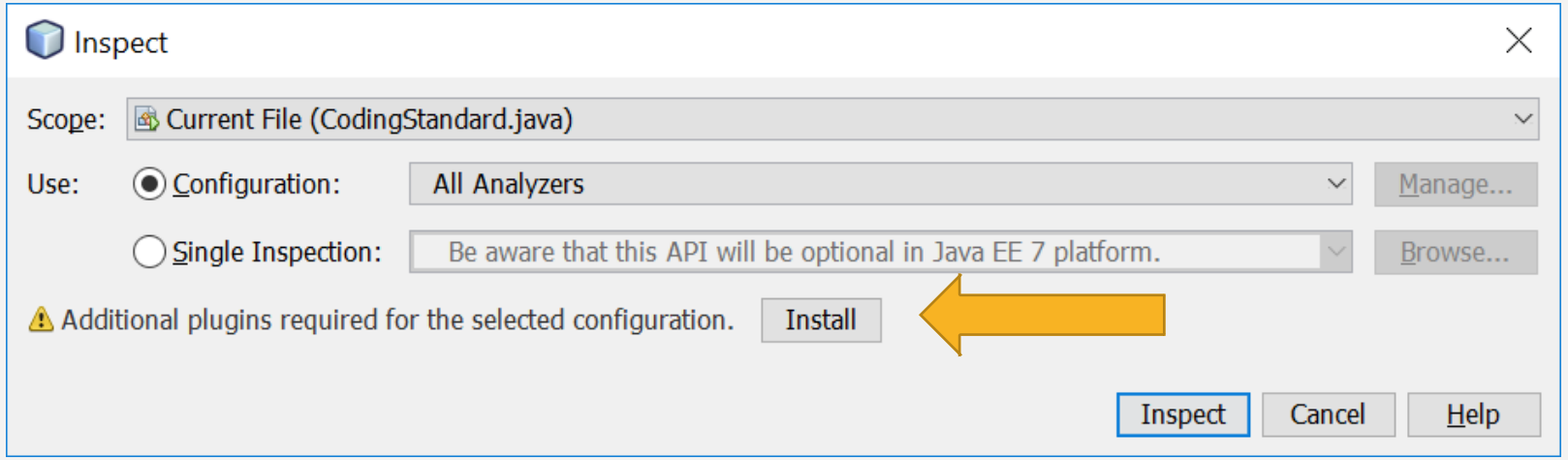
- generates Checkstyle Report:

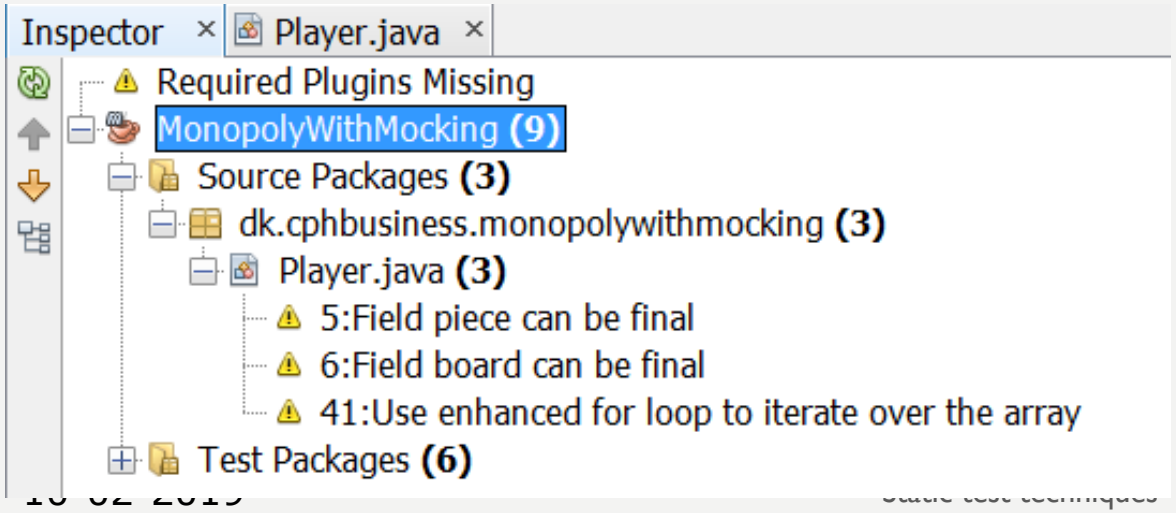
```
mvn checkstyle:checkstyle
```

- Result is in the target/site directory

CODE INSPECTION IN NETBEANS

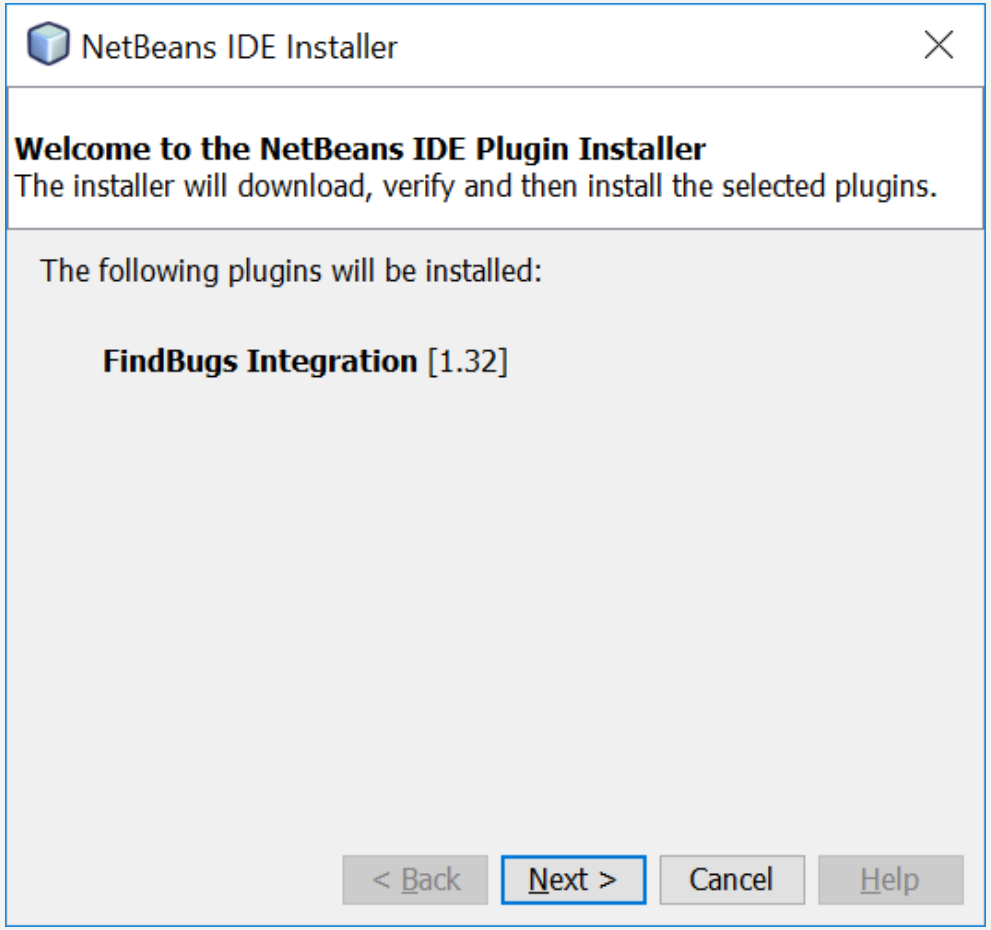
NetBeans: *Choose project/file → Source → Inspect → Configuration → All Analyzers*



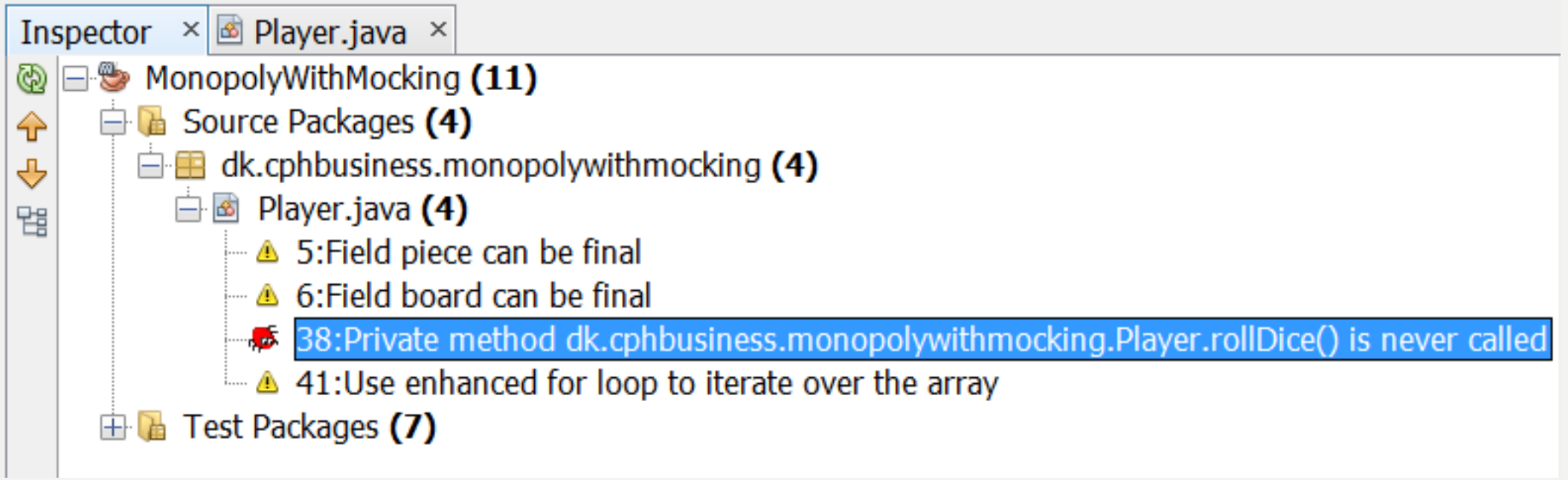


PLUGIN: FIREBUGS

Adds extra advice /warnings



WITH FIREBUGS ONBOARD



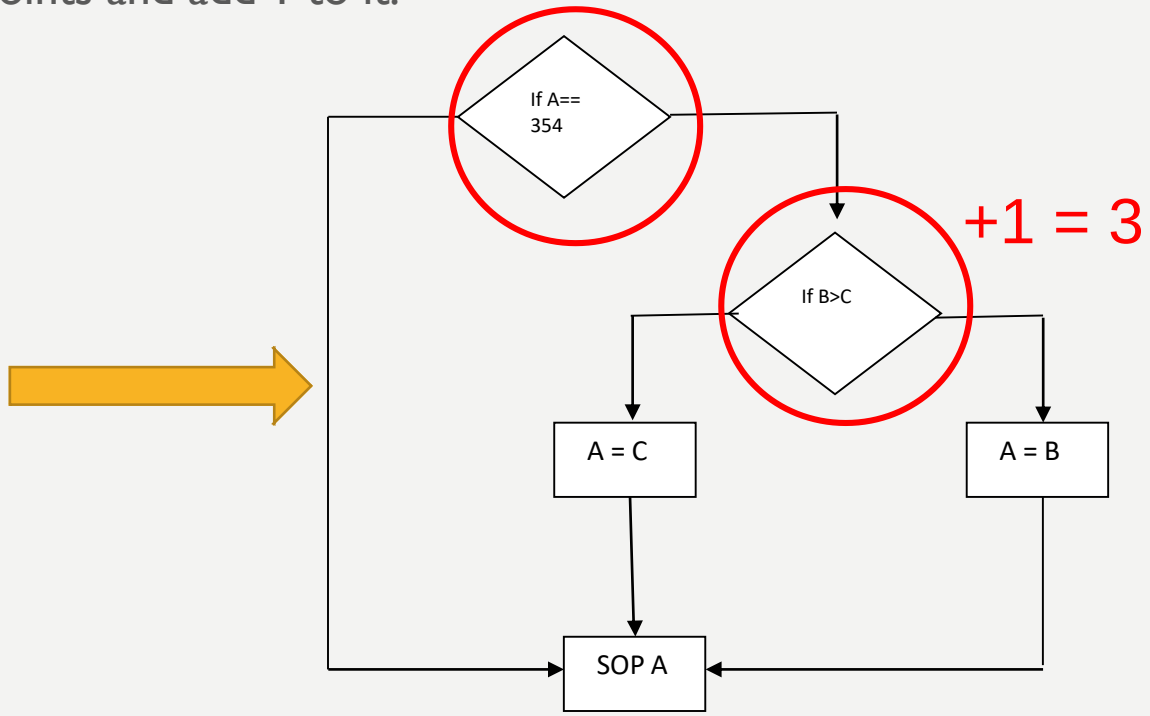
CODE METRICS

CYCLOMATIC COMPLEXITY (CC)

The number of decisions in a program

- A simple way to calculate CC:
  - sum the number of decision points and add 1 to it.

```
if (A == 354)
{
   if (B > C)
      A = B;
   else A = C;
}
System.out.println(A);
```



CYCLOMATIC COMPLEXITY LEVELS

CC comes in different variations (CC, C2, C3):

Metric	Name	Boolean operators	Select Case	Alternative name
CC	Cyclomatic complexity	Not counted	+1 for each Case branch	Regular cyclomatic complexity
CC2	Cyclomatic complexity with Booleans	+1 for each Boolean	+1 for each Case branch	Extended or strict cyclomatic complexity
CC3	Cyclomatic complexity without Cases	Not counted	+1 for an entire Select Case	Modified cyclomatic complexity

Source: http://www.aivosto.com/project/help/pm-complexity.html

WHAT CAN WE USE CC FOR?

- Refactor code
- Identify test cases

- Demo: Let's look at metrics for Palindrome project

THE DEPLOYMENT PIPELINE

Source: Continuous Delivery by Jez Humble & David Farley

EXERCISE 3

Your turn!

Fire up **coding standard tool** for one of your project (again just technical proof of concept):
Checkstyle & Firebugs (or similar in your IDE)

Calculate **Cyclomatic Complexity:**
(e.g. with JaCoCo or Source Code Metrics)
Make sure you understand how your tool calculates the CC metric.