Java for Advanced Programmers

Useful Tools
- Git, Maven and SLF4J

Bernd Kiefer
Jörg Steffen

November 11, 2022

Git & Github

What is Git & Github?

- management tool for your (code) files
- (distributed) version control system
- not connected to any particular language
- GitHub is a hosting service for git
- GitLab is an alternative for on-premises installation

Git & Github

What is Git & Github?

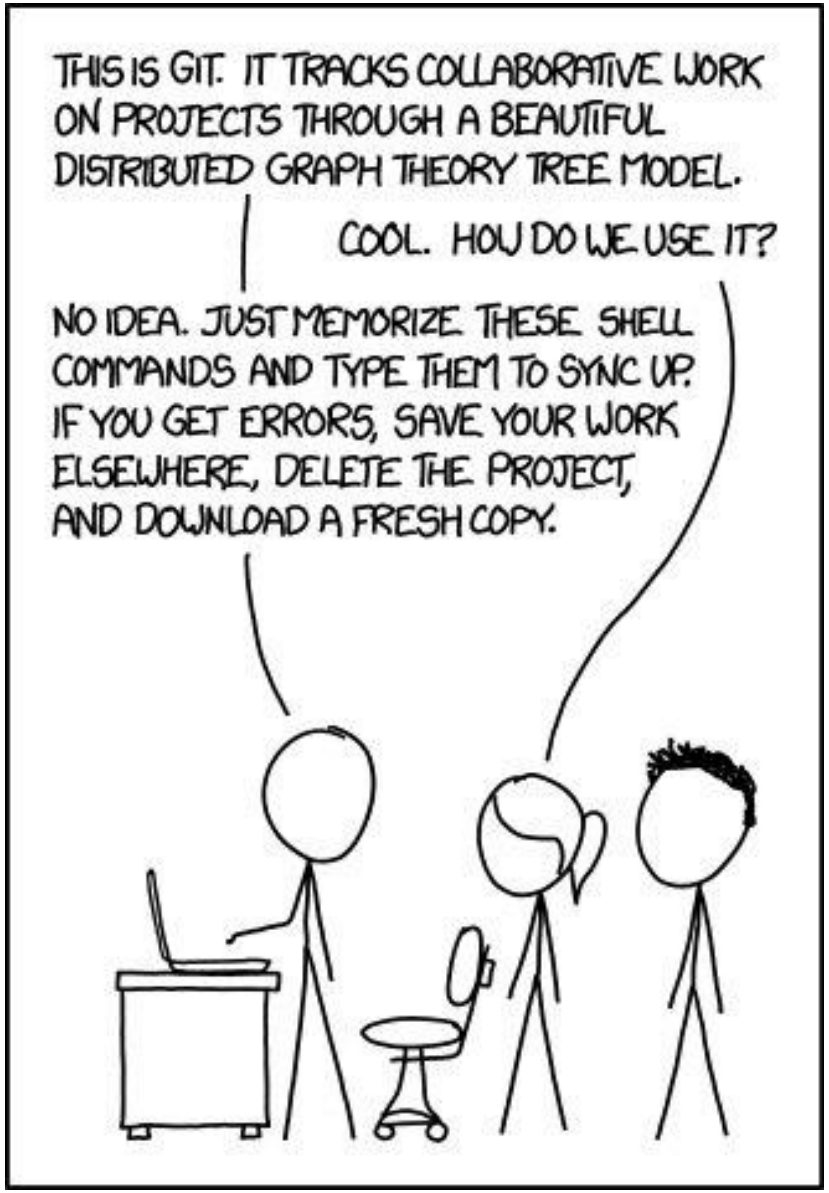
- management tool for your (code) files
- (distributed) version control system
- not connected to any particular language
- GitHub is a hosting service for git
- GitLab is an alternative for on-premises installation

Why do we use it?

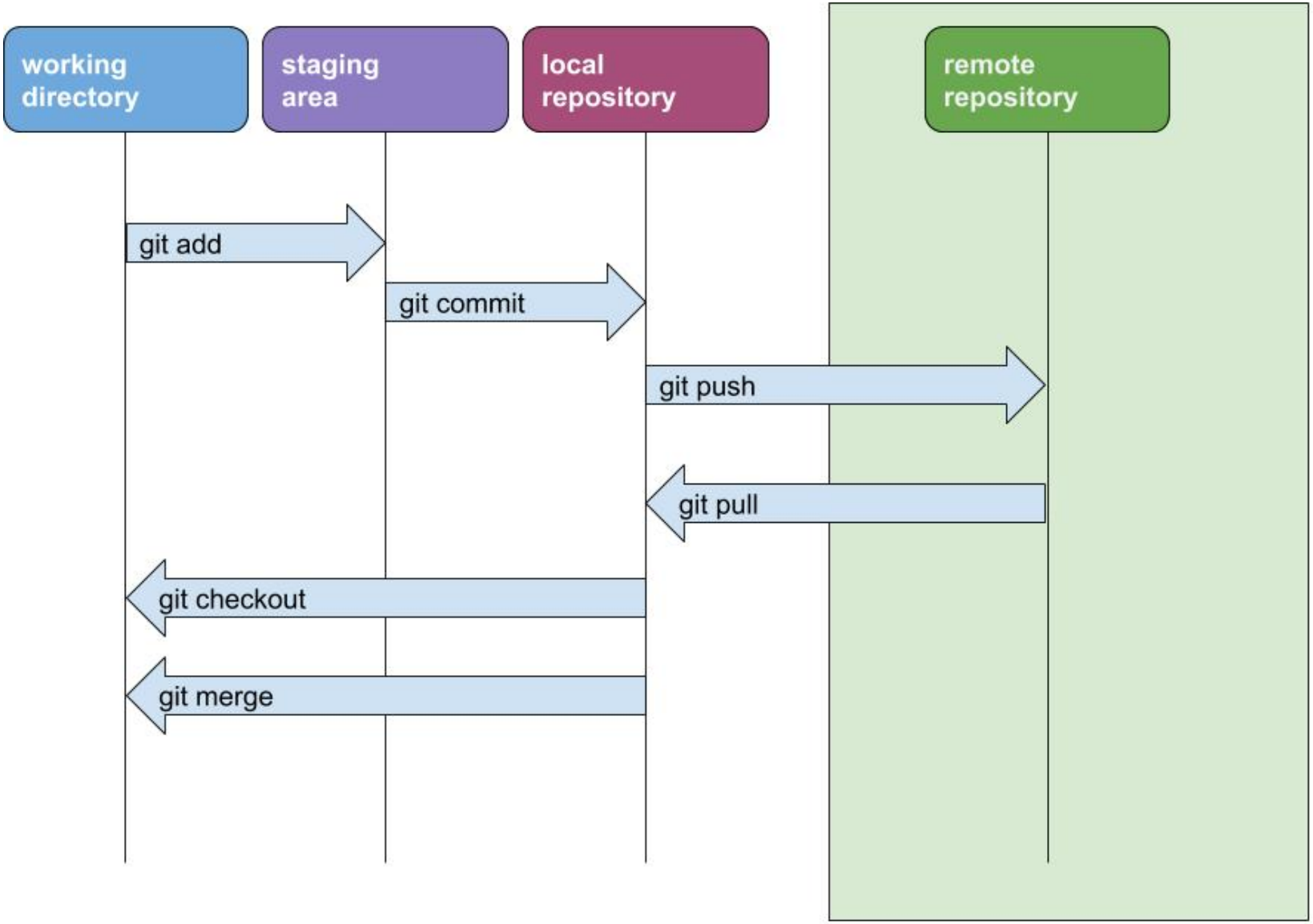
"If you screw things up or lose files, you can easily recover"

- maintain versions of software
- never lose content
- collaborate with others

Git - How does it work?



Git - How does it work?

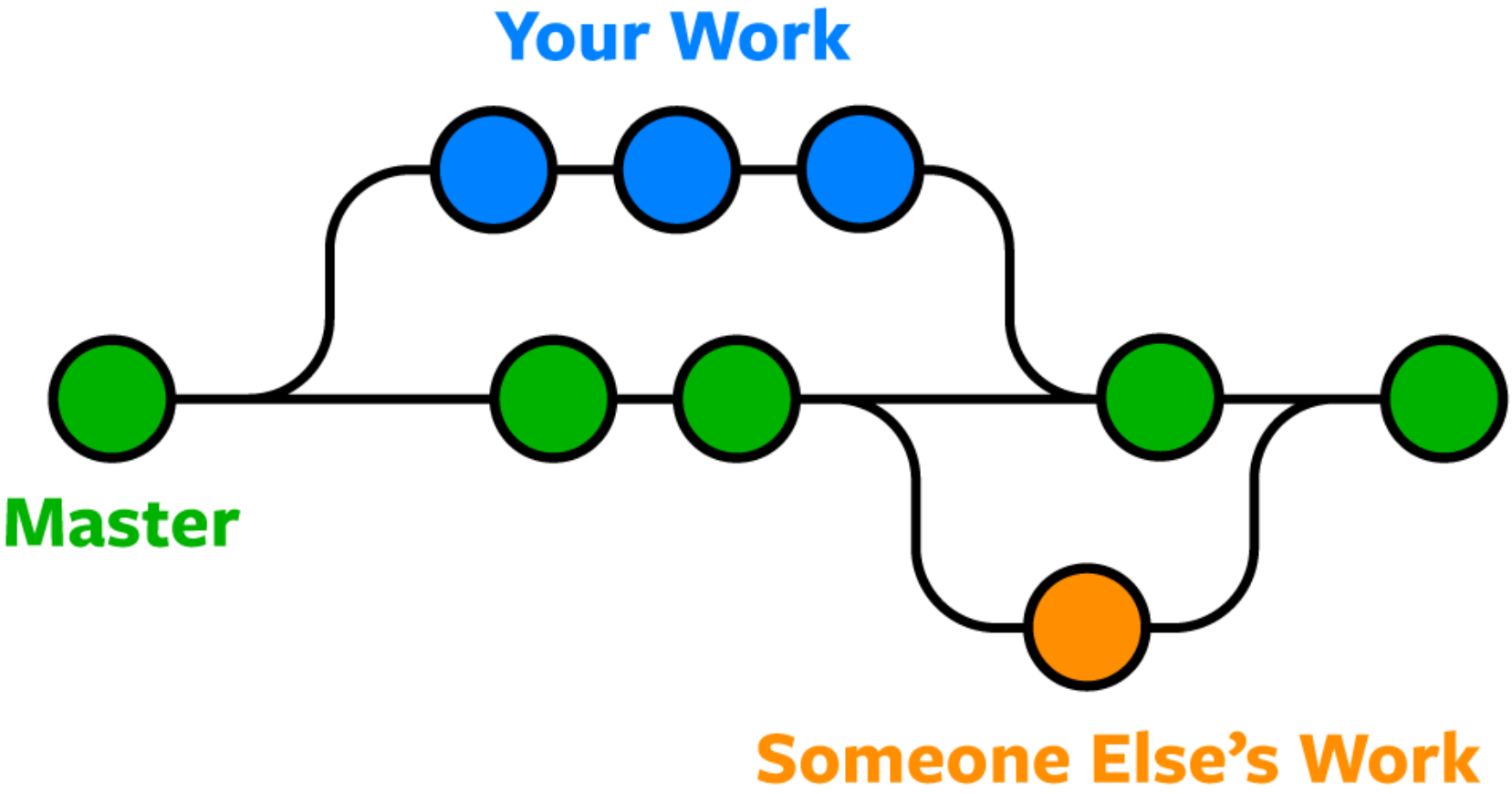


Git - Commands

- `init`: Initializes a git repository - creates the initial .git directory
- `clone`: Creates a GIT repository copy from a remote source
- `add`: Adds file changes in your working directory to your staging area (also called index)
- `commit`: Takes all of the changes written in the index, creates a new commit object pointing to them and sets the branch to point to that new commit
- `push`: Pushes all local commits to the remote repository
- `pull`: Fetches the commits from the remote repository and applies them to your local files
- `reset`: Resets your index and/or working directory to the state of a previous commit
- `status`: Shows you the status of files in the index versus the working directory

Git - Branches

- a chain of commits establishes a branch
- each commit is a snapshot of the project at a specific point
- branches can be split or merged back together
- commands: `branch`, `checkout`, `merge`



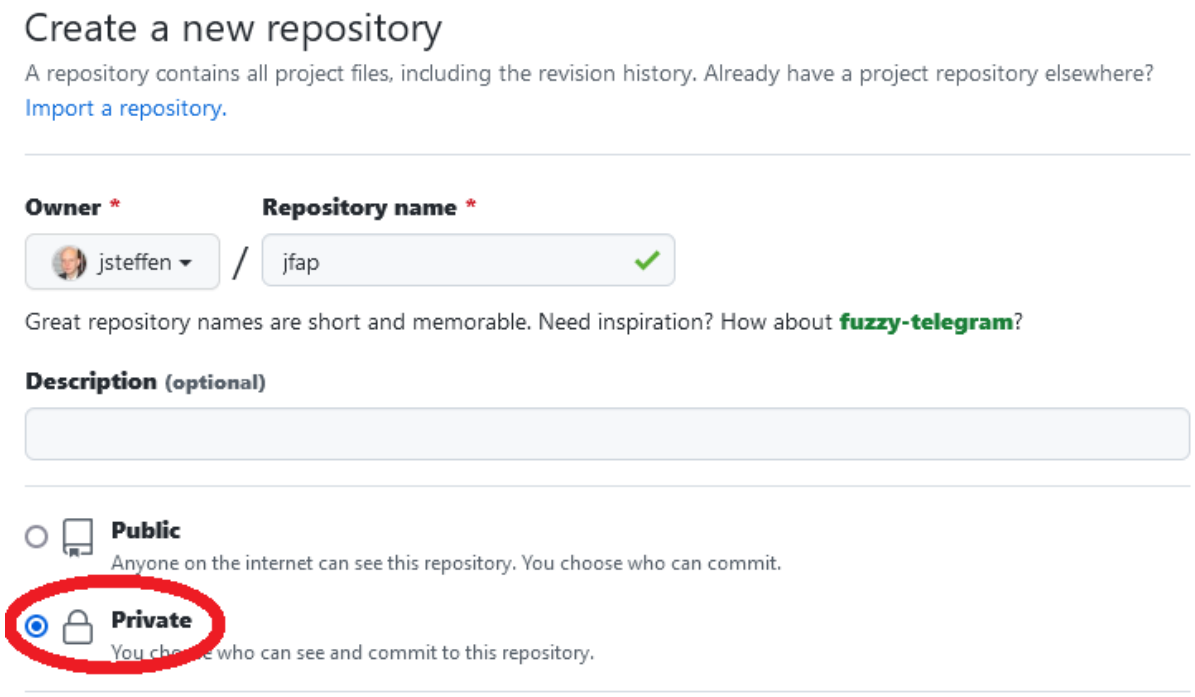
Git Help

Useful stuff:

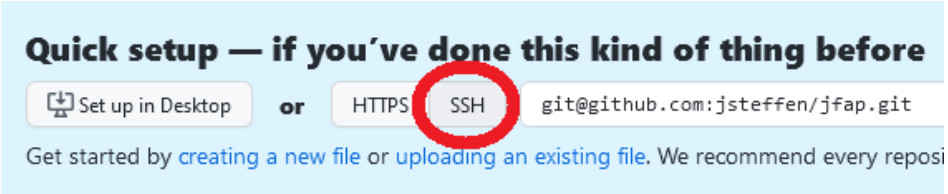
- cheat-sheet:
  `https://education.github.com/git-cheat-sheet-education.pdf`
- git GUIs:
  - `gitk`
  - `https://www.gitkraken.com`
  - git integration in IDEs

Github

- Please make your repository **private !**



- Setup repository to use **ssh !**



Maven™

Maven

What is it?

- project management tool
  - compiling
  - packaging
  - managing classpath
  - pre/post processing tools
- everything defined in a central piece of information POM (Project Object Model) file
- automatically manage dependencies
- "convention over configuration"
- support for all modern IDEs

Maven - Concepts

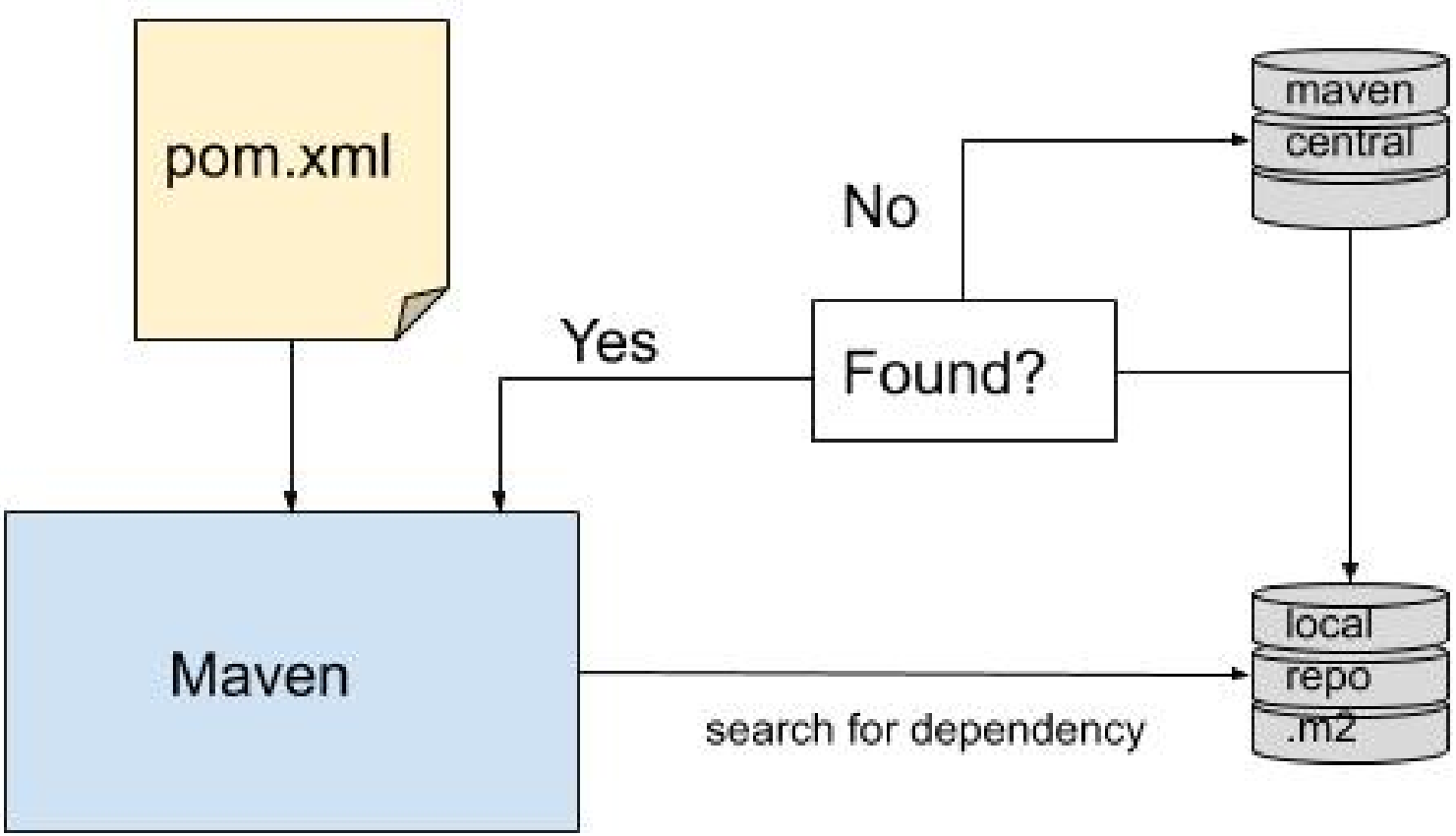
How does it work? (simplified)

- main concept is the **build lifecycle**
- life cycle is divided into **phases**
- phases are sub-divided into **goals**
- `validate`, `compile`, `test`, `package` and `install` are the important phases of the default build life cycle
- phases are defined by (default) plugins
- plugins can be modified and edited in the pom.xml

Maven - pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-ins
  <modelVersion>4.0.0</modelVersion>

  <groupId>de.unisaar.jfap</groupId>
  <artifactId>jfap</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

</project>
```

Maven - Repositories




SLF4J
Simple Logging
Facade for Java

Logging

What's logging?

statements inserted in program code describing what's going on, e.g. exceptions, warnings, errors

- debugging/tracing
- error location, exception handling
- alerting in servers

How?

Many possibilities (simplelogging, java.util.logging, log4j2 ...)
$\Rightarrow$ focus on SLF4J and Logback

Logging - SLF4J and Logback

SLF4J (Simple Logging Facade for Java)

simple facade or interface for various logging frameworks (e.g. java.util.logging, logback, log4j) allowing the end user to plug in the desired logging framework.

Logback

was developed as replacement for Log4j.

- used in many project, mostly behind SLF4J
- logback-classic natively implements SLF4j
- supports multiple output appenders per logger
- provides multiple log levels (`TRACE`, `DEBUG`, `INFO`, `WARN` and `ERROR`)
- fine-grained configuration

SLF4J Setup

1. Add dependencies to project
   - `slf4j-api` and `logback-classic`

SLF4J Setup

1. Add dependencies to project
   - `slf4j-api` and `logback-classic`
2. Add Logback configuration file to classpath in `src/main/resources/logback.xml`

SLF4J Setup

1. Add dependencies to project
   - `slf4j-api` and `logback-classic`
2. Add Logback configuration file to classpath in `src/main/resources/logback.xml`
3. Use logger in Java code

```java
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Hero {

  Role r ;
  int x , y ;

  private static Logger LOGGER = LoggerFactory.getLogger(Hero.class);


  /*...*/
  boolean moveTo ( int deltaX , int deltaY ) {
    LOGGER.info("Moving to {},{}", deltaX, deltaY);
    return false;
  }
```

documentation `https://www.slf4j.org/manual.html`

SLF4J Configuration

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration debug="false">
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
  <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <logger name="de.unisaar" level="INFO"/>
  <root level="DEBUG">
    <appender-ref ref="STDOUT"/>
    <appender-ref ref="FILE"/>
  </root>
</configuration>
```

SLF4J Configuration

```
<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
  <encoder>
    <charset>UTF-8</charset>
    <pattern>%date{dd MMM HH:mm:ss} %-5level %logger{0}: %message%n</pattern>
  </encoder>
  <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
    <level>DEBUG</level>
  </filter>
</appender>
```

SLF4J Configuration

```
<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <!--See also http://logback.qos.ch/manual/appenders.html#RollingFileAppender-->
  <File>jfap.log</File>
  <encoder>
    <charset>UTF-8</charset>
    <pattern>[%thread] %date{yyy-MM-dd HH:mm:ss} %-5level %logger{0}: %message%n</patte
  </encoder>
  <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
    <maxIndex>10</maxIndex>
    <FileNamePattern>jfap.log.%i</FileNamePattern>
  </rollingPolicy>
  <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
    <MaxFileSize>1MB</MaxFileSize>
  </triggeringPolicy>
  <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
    <level>DEBUG</level>
  </filter>
</appender>
```