ZMWO – laboratorium z programowania aspektowego w AspectJ

Temat ćwiczenia: Telecom

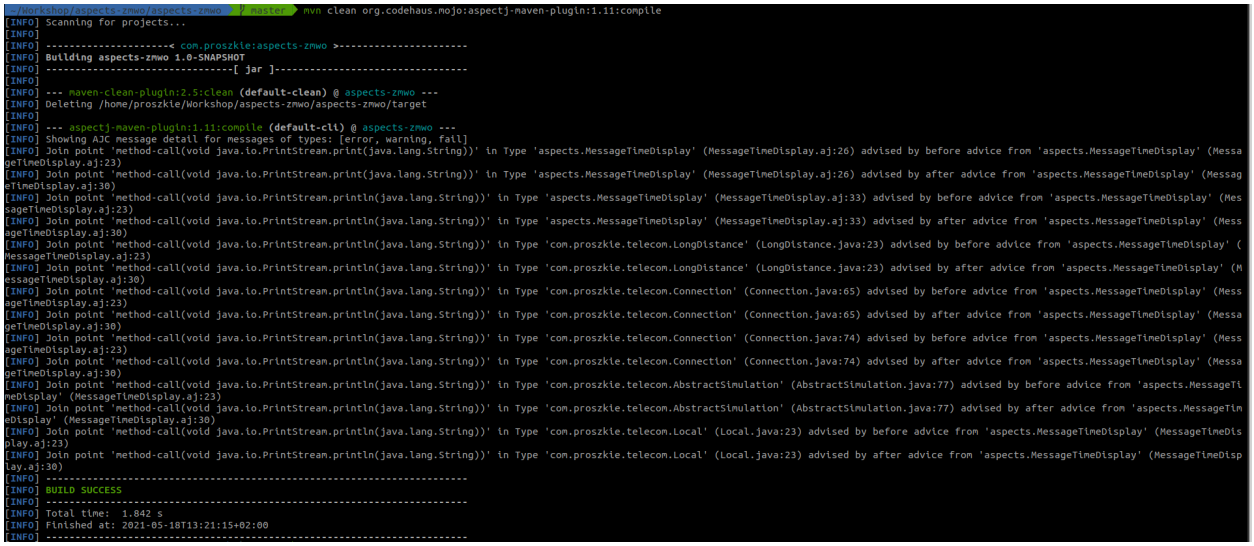
Piotr Proszowski 316914

Przygotowanie środowiska pracy:
Do przeprowadzenia ćwiczenia użyte zostały IDE IntelliJ Ultimate (https://www.jetbrains.com/idea/) oraz narzędzie do budowania Maven (https://maven.apache.org/). Kombinacja ta daje takie same funkcjonalności jak środowisko Eclipse, jednak jako iż dobrze znam wyżej wymienione narzędzia było to dla mnie znacznym uproszczeniem.

Aby skompilować projekt bez aspektów, należy użyć polecenia **mvn clean compile.**

```
~/Workshop/aspects-zmwo/aspects-zmwo  master  mvn clean compile
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.proszkie:aspects-zmwo >----------------------
[INFO] Building aspects-zmwo 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ aspects-zmwo ---
[INFO] Deleting /home/proszkie/Workshop/aspects-zmwo/aspects-zmwo/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ aspects-zmwo ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/proszkie/Workshop/aspects-zmwo/aspects-zmwo/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ aspects-zmwo ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 8 source files to /home/proszkie/Workshop/aspects-zmwo/aspects-zmwo/target/classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.435 s
[INFO] Finished at: 2021-05-18T13:21:09+02:00
[INFO] ------------------------------------------------------------------------
```

Aby podczas kompilacji wpleść aspekty, należy wykonać komendę **mvn clean org.codehaus.mojo:aspectj-maven-plugin:1.11:compile.** Wykorzystana jest tutaj możliwość compile-time weaving.



Przy kompilacji z wplataniem aspektów na ekran zostają wypisane wszystkie sploty.

Aby uruchomić wcześniej skompilowany kod należy użyć komendy **mvn exec:java**

Zadanie 1

Uruchomienie kodu bez aspektów daje poniższy rezultat:

```
~/Workshop/aspects-zmwo/aspects-zmwo  master  mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------< com.proszkie:aspects-zmwo >---------------------
[INFO] Building aspects-zmwo 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ aspects-zmwo ---
jim calls mik...
[new local connection from Jim(650) to Mik(650)]
mik accepts...
connection completed
jim hangs up...
connection dropped
mik calls crista...
[new long distance connection from Mik(650) to Crista(415)]
crista accepts...
connection completed
crista hangs up...
connection dropped
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.358 s
[INFO] Finished at: 2021-05-18T13:25:26+02:00
[INFO] ------------------------------------------------------------------------
```

Widoczna jest symulacja kilku rozmów telefonicznych. Kiedy kod zostanie skompilowany z wplecionymi aspektami, a następnie uruchomiony, program zawiesza się:

```
~/Workshop/aspects-zmwo/aspects-zmwo  master  mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------< com.proszkie:aspects-zmwo >---------------------
[INFO] Building aspects-zmwo 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ aspects-zmwo ---
```

Potwierdzona zostaje więc teza, że problem leży w kodzie dotyczącym aspektów.

Problem leży w źle zdefiniowanym punkcie przecięcia:

```
pointcut printcut(): call(* PrintStream.print*(..));
```

Aspekt MessageTimeDisplay również korzysta z metod PrintStream.print*(..), dlatego advice before będzie wywoływana w nieskończoność.

Jednym z rozwiązań może być ignorowanie całego pakietu, w którym umieszczany jest kod aspektów:

```
pointcut printcut(): call(* PrintStream.print*(..)) && !within(com.proszkie.aspects..*);
```

```
~/Workshop/aspects-zmwo/aspects-zmwo  master  mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.proszkie:aspects-zmwo >----------------------
[INFO] Building aspects-zmwo 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ aspects-zmwo ---
 TIME[Tue May 18 13:50:18 CEST 2021]: jim calls mik...
 EOM [Tue May 18 13:50:18 CEST 2021]
 TIME[Tue May 18 13:50:18 CEST 2021]: [new local connection from Jim(650) to Mik(650)]
 EOM [Tue May 18 13:50:18 CEST 2021]
 TIME[Tue May 18 13:50:19 CEST 2021]: mik accepts...
 EOM [Tue May 18 13:50:19 CEST 2021]
 TIME[Tue May 18 13:50:19 CEST 2021]: connection completed
 EOM [Tue May 18 13:50:19 CEST 2021]
 TIME[Tue May 18 13:50:21 CEST 2021]: jim hangs up...
 EOM [Tue May 18 13:50:21 CEST 2021]
 TIME[Tue May 18 13:50:21 CEST 2021]: connection dropped
 EOM [Tue May 18 13:50:21 CEST 2021]
 TIME[Tue May 18 13:50:21 CEST 2021]: mik calls crista...
 EOM [Tue May 18 13:50:21 CEST 2021]
 TIME[Tue May 18 13:50:21 CEST 2021]: [new long distance connection from Mik(650) to Crista(415)]
 EOM [Tue May 18 13:50:21 CEST 2021]
 TIME[Tue May 18 13:50:21 CEST 2021]: crista accepts...
 EOM [Tue May 18 13:50:21 CEST 2021]
 TIME[Tue May 18 13:50:21 CEST 2021]: connection completed
 EOM [Tue May 18 13:50:21 CEST 2021]
 TIME[Tue May 18 13:50:23 CEST 2021]: crista hangs up...
 EOM [Tue May 18 13:50:23 CEST 2021]
 TIME[Tue May 18 13:50:23 CEST 2021]: connection dropped
 EOM [Tue May 18 13:50:23 CEST 2021]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.282 s
[INFO] Finished at: 2021-05-18T13:50:23+02:00
[INFO] ------------------------------------------------------------------------
```

Po wprowadzonej zmianie program działa tak jak powinien: wyświetla czas przyjścia wiadomości oraz czas po jej przetworzeniu.

Zadanie 2

W celu wykonania tego ćwiczenia został stworzony aspekt com.proszkie.aspects.MethodsCallInvestigator.

Zdefiniowany został odpowiedni punkt przecięcia:

```
pointcut methodExecution(Object caller): this(caller) && execution(* com.proszkie.telecom..*(..));
```

Interesujące są wywołania wszystkich metody w pakiecie com.proszkie.telcom oraz obiekt, który je wywołuje.

Została również zdefiniowana porada around, która pozwala na zmierzenie czasu wywoływanej metody oraz przechwycenie jej rezultatu:

```
Object around(Object caller): methodExecution(caller) {
    Timer timer = new Timer();
    timer.start();
    Object result = proceed(caller);
    timer.stop();
    appendToFile(caller, result, timer, thisJoinPoint);
    return result;
}
```

Następnie wszystkie zebrane informacje zostają zapisane do pliku:

```
private void appendToFile(Object caller, Object result, Timer timer, JoinPoint joinPoint) {
    int currentId = id.incrementAndGet();
    int indentLevel = currentId - oldestId.get() - 1;
    appendToFile(indentLevel, caller, result, timer, joinPoint);
    oldestId.incrementAndGet();
}

private void appendToFile(int indentLevel, Object caller, Object result, Timer timer, JoinPoint joinPoint) {
    appendToFile(indentLevel, content: "Class of object calling the method: " + caller.getClass());
    appendToFile(indentLevel, content: "Called method signature: " + joinPoint.getSignature());
    appendToFile(indentLevel, content: "Called method args: " + Arrays.toString(joinPoint.getArgs()));
    appendToFile(indentLevel, content: "Called method result: " + result);
    appendToFile(indentLevel, content: "Processing time: " + timer.getTime() + " ms");
}

private void appendToFile(int indentLevel, String content) {
    String indent = IntStream.range(0, indentLevel).mapToObj(i -> "---").collect(Collectors.joining());
    try {
        Files.write(debugFile.toPath(), indent.concat(content.concat("\n")).getBytes(StandardCharsets.UTF_8), StandardOpenOption.APPEND);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Po uruchomieniu programu pod ścieżką /tmp/aspects.log pojawia się plik zawierający 225 linijek. Fragment pliku:

```
Class of object calling the method: class com.proszkie.telecom.Customer
Called method signature: boolean com.proszkie.telecom.Customer.localTo(Customer)
---Class of object calling the method: class com.proszkie.telecom.Customer
---Called method signature: String com.proszkie.telecom.Customer.toString()
---Called method args: []
---Called method result: Jim(650)
---Processing time: 0 ms
Called method args: [Jim(650)]
Called method result: true
Processing time: 1 ms
Class of object calling the method: class com.proszkie.telecom.Customer
Called method signature: String com.proszkie.telecom.Customer.toString()
Called method args: []
Called method result: Jim(650)
Processing time: 0 ms
Class of object calling the method: class com.proszkie.telecom.Customer
Called method signature: String com.proszkie.telecom.Customer.toString()
Called method args: []
Called method result: Mik(650)
Processing time: 0 ms
Class of object calling the method: class com.proszkie.telecom.Customer
Called method signature: void com.proszkie.telecom.Customer.addCall(Call)
Called method args: [com.proszkie.telecom.Call@17765eec]
Called method result: null
Processing time: 0 ms
Class of object calling the method: class com.proszkie.telecom.Customer
Called method signature: Call com.proszkie.telecom.Customer.call(Customer)
---Class of object calling the method: class com.proszkie.telecom.Customer
---Called method signature: String com.proszkie.telecom.Customer.toString()
---Called method args: []
```

Linijki poprzedzone znakami “---” wskazują na poziom zagnieżdżenia metody. Jeśli w środku metody A zostaje wywołana metoda B, informacje na temat metody B zostaną natychmiastowo wypisane (poprzedzone znakami “---”), a następnie zostanie wypisana reszta danych na temat metody A.

Zadanie 3

W celu wykonania zadania został stworzony plik z danymi uwierzytelniającymi:

```
~/Workshop/aspects-zmwo/aspects-zmwo  master  cat /tmp/zmwo-credentials.txt
Jim(650):1234
Mik(650):7890
```

W aspekcie CallAuthorization plik jest wczytywany przy starcie programu oraz zapisywany do pamięci:

```
private static final Map<String, String> credentials = new HashMap<>();
private static final File credentialsFile = new File( pathname: "/tmp/zmwo-credentials.txt");

static {
    loadCredentials();
}

private static void loadCredentials() {
    if (!credentialsFile.exists()) {
        throw new IllegalStateException("Credentials file was not found");
    }
    try {
        Files.readAllLines(credentialsFile.toPath()) List<String>
                .stream() Stream<String>
                .map(line -> line.split( regex: ":")) Stream<String[]>
                .forEach(splitted -> credentials.put(splitted[0], splitted[1]));
    } catch (IOException e) {
        throw new IllegalStateException("Credentials file has been corrupted");
    }
}
```

Zdefiniowany został punkt przecięcia:

```
pointcut customerCall(Customer caller): this(caller) && execution(* com.proszkie.telecom.Customer.call(..));
```

W momencie gdy wywołana zostaje metoda Customer.call(..) użytkownik proszony jest o hasło. W przypadku złego hasła zamiast wywołania właściwej metody (Customer.call) użytkownik komunikowany jest o błędzie:

```
Object around(Customer caller): customerCall(caller) {
    System.out.println("Password for " + caller + " required: ");
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    try {
        String password = br.readLine();
        if (credentials.containsKey(caller.toString())) {
            if (password.equals(credentials.get(caller.toString()))) {
                System.out.println("Successful authorization.");
                return proceed(caller);
            } else {
                System.out.println("Wrong password");
            }
        } else {
            System.out.println("Customer does not exist");
        }
    } catch (IOException e) {
        System.out.println("Error occured during reading the password");
    }

    return Optional.empty();
}
```

```
 TIME[Wed May 19 15:51:05 CEST 2021]: jim calls mik...
 EOM [Wed May 19 15:51:05 CEST 2021]
Password for Jim(650) required:
uji
Wrong password
 TIME[Wed May 19 15:51:08 CEST 2021]: mik calls crista...
 EOM [Wed May 19 15:51:08 CEST 2021]
Password for Mik(650) required:
7890
Successful authorization.
 TIME[Wed May 19 15:51:11 CEST 2021]: [new long distance connection from Mik(650) to Crista(415)]
 EOM [Wed May 19 15:51:11 CEST 2021]
 TIME[Wed May 19 15:51:12 CEST 2021]: crista accepts...
 EOM [Wed May 19 15:51:12 CEST 2021]
 TIME[Wed May 19 15:51:12 CEST 2021]: connection completed
 EOM [Wed May 19 15:51:12 CEST 2021]
 TIME[Wed May 19 15:51:13 CEST 2021]: crista hangs up...
 EOM [Wed May 19 15:51:13 CEST 2021]
 TIME[Wed May 19 15:51:13 CEST 2021]: connection dropped
 EOM [Wed May 19 15:51:13 CEST 2021]
```

Aby osiągnąć taki wynik jak powyżej należało minimalnie zrefaktoryzować metodę Customer.call(..):

```
public Optional<Call> call(Customer receiver) {
    Call call = new Call( caller: this, receiver);
    addCall(call);
    return Optional.of(call);
}
```

Oraz miejsca w których była wywołana:

```
say( s: "jim calls mik...");
Optional<Call> c1 = jim.call(mik);
c1.ifPresent(call -> {
    wait( seconds: 1.0);
    say( s: "mik accepts...");
    mik.pickup(call);
    wait( seconds: 2.0);
    say( s: "jim hangs up...");
    jim.hangup(call);
    report(jim);
    report(mik);
    report(crista);
});
```

Zadanie 4

W celu wykonania tego zadania, poprzedni aspekt CallAuthorization musiał zostać lekko zmodyfikowany. W poradzie “before” pobierane jest hasło od użytkownika i zapisywane w polu “password”:

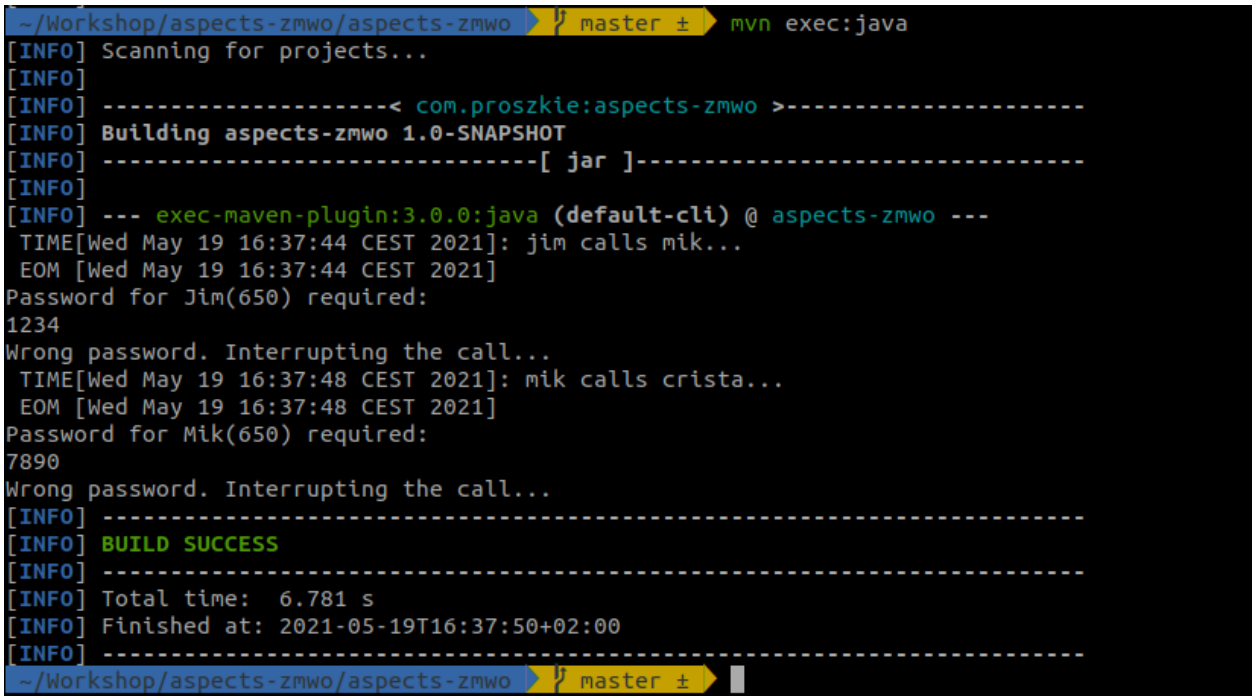
```
before(Customer caller): customerCall(caller) {
    System.out.println("Password for " + caller + " required: ");
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    try {
        password = br.readLine();
    } catch (IOException e) {
        System.out.println("Error occured during reading the password");
    }
}
```

Następnie uprzednio zapisane hasło jest sprawdzane w taki sam sposób jak poprzednio:

```
Object around(Customer caller): customerCall(caller) {
    if (credentials.containsKey(caller.toString())) {
        if (password.equals(credentials.get(caller.toString()))) {
            System.out.println("Successful authorization.");
            return proceed(caller);
        } else {
            System.out.println("Wrong password. Interrupting the call...");
        }
    } else {
        System.out.println("Customer does not exist");
    }

    return Optional.empty();
}
```

Stworzony został aspekt Hacker, który zmienia zawartość pola “password” tuż przed wywołaniem porady uwierzytelniającej. Wynikiem tego działania jest zawsze błędne hasło:

```
public aspect Hacker {

    pointcut callAuthorization(): within(com.proszkie.aspects.CallAuthorization) && adviceexecution();

    before(): callAuthorization() {
        try {
            Field passwordField = thisJoinPoint.getThis().getClass().getDeclaredField( name: "password");
            passwordField.setAccessible(true);
            passwordField.set(thisJoinPoint.getThis(), "");
        } catch (NoSuchFieldException | IllegalAccessException ignored) {
        }
    }
}
```



Aby uchronić się przed działaniem aspektu dostarczonego przez hackera został stworzony aspekt “Policeman”, który przerywa aspekt Hackera:

```
public aspect Policeman {

    pointcut hackerCall(): within(com.proszkie.aspects.Hacker) && adviceexecution();

    void around(): hackerCall() {
        return;
    }
}
```

Po kompilacji i uruchomieniu programu wszystko działa jak należy:

```
~/Workshop/aspects-zmwo/aspects-zmwo  master ±  mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.proszkie:aspects-zmwo >----------------------
[INFO] Building aspects-zmwo 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ aspects-zmwo ---
 TIME[Wed May 19 16:48:18 CEST 2021]: jim calls mik...
 EOM [Wed May 19 16:48:18 CEST 2021]
Password for Jim(650) required:
1234
Successful authorization.
 TIME[Wed May 19 16:48:20 CEST 2021]: [new local connection from Jim(650) to Mik(650)]
 EOM [Wed May 19 16:48:20 CEST 2021]
 TIME[Wed May 19 16:48:21 CEST 2021]: mik accepts...
 EOM [Wed May 19 16:48:21 CEST 2021]
 TIME[Wed May 19 16:48:21 CEST 2021]: connection completed
 EOM [Wed May 19 16:48:21 CEST 2021]
 TIME[Wed May 19 16:48:23 CEST 2021]: jim hangs up...
 EOM [Wed May 19 16:48:23 CEST 2021]
 TIME[Wed May 19 16:48:23 CEST 2021]: connection dropped
 EOM [Wed May 19 16:48:23 CEST 2021]
 TIME[Wed May 19 16:48:23 CEST 2021]: mik calls crista...
 EOM [Wed May 19 16:48:23 CEST 2021]
Password for Mik(650) required:
7890
Successful authorization.
 TIME[Wed May 19 16:48:25 CEST 2021]: [new long distance connection from Mik(650) to Cris
 EOM [Wed May 19 16:48:25 CEST 2021]
 TIME[Wed May 19 16:48:25 CEST 2021]: crista accepts...
 EOM [Wed May 19 16:48:25 CEST 2021]
 TIME[Wed May 19 16:48:25 CEST 2021]: connection completed
 EOM [Wed May 19 16:48:25 CEST 2021]
 TIME[Wed May 19 16:48:27 CEST 2021]: crista hangs up...
 EOM [Wed May 19 16:48:27 CEST 2021]
 TIME[Wed May 19 16:48:27 CEST 2021]: connection dropped
 EOM [Wed May 19 16:48:27 CEST 2021]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  9.580 s
[INFO] Finished at: 2021-05-19T16:48:27+02:00
[INFO] ------------------------------------------------------------------------
~/Workshop/aspects-zmwo/aspects-zmwo  master ±
```