Sprawozdanie – ZMWO
Programowanie aspektowe w AspectJ

Magdalena Zych
Nr albumu: 293182
Wybrany temat: Repozytorium projektów

1. Stworzony został aspekt *LoggingAspect*. Punkt przecięcia *logging* przecina każde wykonanie funkcji w pakiecie *repository*. W ramach rady *around* tworzony jest log dla pojedynczej metody w postaci linijki tekstu. Czas w milisekundach jest pobierany bezpośrednio przed i po wywołaniu funkcji *proceed*. Tekst zapisywany jest do pliku za pomocą funkcji *writeToFile*.

Kod:

```
public aspect LoggingAspect {

    private FileWriter file = null;
    String separator = "; ";

    pointcut logging():
        execution(* *.*( .. )) && within(repository.*);

        Object around(): logging(){

            // klasa obiektu, na rzecz którego została wykonana metoda
            String line = "Klasa: " + thisJoinPoint.getSignature().getDeclaringType().toString() + separator;
            line += "Metoda: " + thisJoinPoint.getSignature().getName() + separator; // nazwa metody

            long startTime = Calendar.getInstance().getTimeInMillis();
            Object ret = proceed();
            long endTime = Calendar.getInstance().getTimeInMillis();
            long executionTime = endTime - startTime;

            line += "Zwrócony wynik: " + ret + separator; // wynik zwracany przez metode
            line += "Czas wykonania [ms]: " + executionTime + separator; // czas wykonania metody


            CodeSignature sig = (CodeSignature) thisJoinPoint.getSignature();
            String [] parameterNames = sig.getParameterNames();
            line += "Argumenty: ";
            if(parameterNames.length == 0) {
                line += "Brak";
            } else {
                int i = 0;
                for (Object arg : thisJoinPoint.getArgs()) {
                    line += (i + 1) + ". ";
                    line += "Typ: " + arg.getClass() + ", "; // typ argumentu
                    line += "Nazwa: " + parameterNames[i] + ", "; // nazwa argumentu
                    line += "Wartość: " + arg + separator; // wartosc argumentu
                    ++i;
                }
            }

            writeToFile(line);
            return ret;
        }

        private void writeToFile(String text) {
            try {
                if(file == null) {
                    file = new FileWriter("log.txt");
                }

                file.write(text);
                file.write(System.getProperty("line.separator"));
                file.flush();

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
}
```

Rezultaty z pliku log.txt:

```
Klasa: class repository.Task; Metoda: getId; Zwrócony wynik: 1; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Project; Metoda: addTask; Zwrócony wynik: null; Czas wykonania [ms]: 1; Argumenty: 1. Typ: class repository.Task,
Nazwa: task, Wartość: repository.Task@64729b1e;
Klasa: class repository.Task; Metoda: getId; Zwrócony wynik: 2; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Project; Metoda: addTask; Zwrócony wynik: null; Czas wykonania [ms]: 1; Argumenty: 1. Typ: class repository.Task,
Nazwa: task, Wartość: repository.Task@10bbd20a;
Klasa: class repository.Task; Metoda: getId; Zwrócony wynik: 3; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Project; Metoda: addTask; Zwrócony wynik: null; Czas wykonania [ms]: 0; Argumenty: 1. Typ: class repository.Task,
Nazwa: task, Wartość: repository.Task@48503868;
Klasa: class repository.Project; Metoda: getName; Zwrócony wynik: project1; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Repository; Metoda: addProject; Zwrócony wynik: null; Czas wykonania [ms]: 0; Argumenty: 1. Typ: class
repository.Project, Nazwa: project, Wartość: repository.Project@6895a785;
Klasa: class repository.Project; Metoda: getName; Zwrócony wynik: Project2; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Repository; Metoda: addProject; Zwrócony wynik: null; Czas wykonania [ms]: 0; Argumenty: 1. Typ: class
repository.Project, Nazwa: project, Wartość: repository.Project@184f6be2;
Klasa: class repository.Project; Metoda: getName; Zwrócony wynik: project1; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Project; Metoda: getTaks; Zwrócony wynik: [repository.Task@64729b1e, repository.Task@10bbd20a]; Czas wykonania [ms]:
0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getId; Zwrócony wynik: 1; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getDescription; Zwrócony wynik: task1; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getReporter; Zwrócony wynik: repository.Employee@5a1c0542; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Employee; Metoda: getName; Zwrócony wynik: user1; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getReporter; Zwrócony wynik: repository.Employee@5a1c0542; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Employee; Metoda: getLastName; Zwrócony wynik: user1; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getAssignee; Zwrócony wynik: repository.Employee@5a1c0542; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Employee; Metoda: getName; Zwrócony wynik: user1; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getAssignee; Zwrócony wynik: repository.Employee@5a1c0542; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Employee; Metoda: getLastName; Zwrócony wynik: user1; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getStatus; Zwrócony wynik: NEW; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getId; Zwrócony wynik: 2; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getDescription; Zwrócony wynik: task2; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getReporter; Zwrócony wynik: repository.Employee@5a1c0542; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Employee; Metoda: getName; Zwrócony wynik: user1; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getReporter; Zwrócony wynik: repository.Employee@5a1c0542; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Employee; Metoda: getLastName; Zwrócony wynik: user1; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getAssignee; Zwrócony wynik: repository.Employee@396f6598; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Employee; Metoda: getName; Zwrócony wynik: user2; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getAssignee; Zwrócony wynik: repository.Employee@396f6598; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Employee; Metoda: getLastName; Zwrócony wynik: user2; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.Task; Metoda: getStatus; Zwrócony wynik: NEW; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.ProjectExporter; Metoda: exportTasks; Zwrócony wynik: null; Czas wykonania [ms]: 5; Argumenty: 1. Typ: class
java.io.PrintStream, Nazwa: printStream, Wartość: java.io.PrintStream@394e1a0f; 2. Typ: class java.util.LinkedList, Nazwa: tasks, Wartość:
[repository.Task@64729b1e, repository.Task@10bbd20a];
Klasa: class repository.Project; Metoda: getDocuments; Zwrócony wynik: []; Czas wykonania [ms]: 0; Argumenty: Brak
Klasa: class repository.ProjectExporter; Metoda: exportDocuments; Zwrócony wynik: null; Czas wykonania [ms]: 0; Argumenty: 1. Typ: class
java.io.PrintStream, Nazwa: printStream, Wartość: java.io.PrintStream@394e1a0f; 2. Typ: class java.util.LinkedList, Nazwa: documents, Wartość:
[];
Klasa: class repository.ProjectExporter; Metoda: export; Zwrócony wynik: null; Czas wykonania [ms]: 8; Argumenty: 1. Typ: class
repository.Project, Nazwa: project, Wartość: repository.Project@6895a785; 2. Typ: class java.io.PrintStream, Nazwa: printStream, Wartość:
java.io.PrintStream@394e1a0f;
Klasa: class repository.Main; Metoda: main; Zwrócony wynik: null; Czas wykonania [ms]: 71; Argumenty: 1. Typ: class [Ljava.lang.String;,
Nazwa: args, Wartość: [Ljava.lang.String;@27a5f880;
```

2. Wykorzystany został mechanizm Inter-type declarations. Utworzone zostały 2 punkty przecięcia: przy modyfikacji repozytorium i przy tworzeniu repozytorium. Po każdej modyfikacji powinna następować serializacja obiektu typu *Repository*. Przy tworzeniu obiektu *Repository* obiekt powinien być wczytywany z pliku. Niestety przy wczytywaniu obiektu natrafiłam na problem. Obiekt Repository nie wczytuje się poprawnie i próba wykonania na nim czynności (na przykład wywołanie funkcji *getProjects*) powoduje błąd.

```aspectj
public aspect SaveRepositoryAspect {
    declare parents: Repository implements Serializable;

    pointcut creation():
        execution(public Repository.new())
        && !within(SaveRepositoryAspect);

    pointcut modification():
        execution(public void repository.Repository.addProject(Project))
        || execution(public void repository.Repository.updateProject(Project))
        || execution(public void repository.Repository.deleteProject(Project));

    after(): modification(){
        Repository repo = (Repository) thisJoinPoint.getTarget();
        try (ObjectOutputStream outputStream = new ObjectOutputStream(
                new FileOutputStream("serialized_repository.bin"))) {
            outputStream.writeObject(repo);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    Object around(): creation(){
        Repository repo = null;
        try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream("serialized_repository.bin"))) {
            repo = (Repository) inputStream.readObject();

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }

        return repo;
    }
}
```

Nie udało mi się wykonać pozostałych zadań.