Code Transformation using Spoon Library for Software Diversification

Salma BABA

Introduction

This report demonstrates how the Spoon library is used for code transformation in Java. The task focuses on transforming `if` conditions and converting `for` loops into `while` loops, with an emphasis on ensuring the transformed code behaves identically to the original code. This exercise can be related to the concept of software diversification, where the goal is to modify code in ways that improve robustness and reliability without altering its overall behavior.

Code Transformation Overview

The Spoon library is used to analyze and transform Java code programmatically. In this project, the transformation focuses on two main aspects:

- **Inverting `if` conditions:** This transformation involves negating the condition in an `if` statement and swapping the `then` and `else` blocks, maintaining the original logic. For example:

```
if (temperature > 25) {
    // hot weather
} else {
    // cool weather
}
```

  becomes

```
if (!(temperature > 25)) {
    // cool weather
} else {
    // hot weather
}
```

- **Converting `for` loops to `while` loops:** This transformation converts `for` loops into `while` loops, while keeping the original logic intact. For example:

```
for (int i = 0; i < 10; i++) {
    System.out.println(i);
}
```

becomes

```
int i = 0;
while (i < 10) {
    System.out.println(i);
    i++;
}
```

Code Explanation

The transformation is performed using a `SpoonWeather` class that uses Spoon's API to process the original `WeatherAdvisor.java` file. The key steps in the transformation process include:

- Setting up the `Launcher` and configuring the Spoon environment to process the source code.
- Adding processors that apply transformations to the `if` conditions and `for` loops.
- Running the transformation and printing the transformed code.

The transformed code is then saved in the `spooned/` directory for further use.

Testing the Transformation

Once the transformation is applied, tests are run to compare the outputs of both the original and transformed versions. The `JUnit` testing framework is used to verify that the outputs from the two versions match, ensuring that the transformation does not affect the program's behavior.

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.example.spoon.WeatherAdvisorTest
Original Version Output:
It's a cool day. Dress warmly.
The sky is not clear. Keep an umbrella just in case.
It's windy outside. Hold on to your hat!
No rain today. You can leave your umbrella at home.

Transformed Version Output:
It's a cool day. Dress warmly.
The sky is not clear. Keep an umbrella just in case.
It's windy outside. Hold on to your hat!
No rain today. You can leave your umbrella at home.
```

Figure 1: Test Output Screenshot

Conclusion

This project demonstrates the use of Spoon for automated code transformation. The transformation of `if` conditions and `for` loops illustrates how Spoon can be used to diversify Java code while ensuring that the program behaves identically. The tests confirm that the transformation preserves the original functionality, supporting the idea of software diversification for enhanced system reliability and robustness.