
MINNESOTA INCOME TAX CALCULATOR

REENGINEERING THE LEGACY CODE

1.

	Date 10/6/2022
--	----------------

GOAL OF THE PROJECT

The goal of this project is to reengineer a Java application. At a glance, the application serves for **the income tax calculation of the Minnesota state citizens**. The tax calculation accounts for the marital status of a given citizen, his income, and the amount of money that he has spend, as witnessed by a set of receipts declared along with the income. The legacy application takes as **input txt or xml** files that contain the necessary data for each citizen. The tax calculation is based on a complex algorithm provided by the Minnesota state. The application further produces graphical representations of the data in terms of **bar and pie charts**. Finally. the application produces respective **output** reports in **txt or xml**.

REVERSE ENGINEERING

FIRST CONTACT

1. Skim the documentation: The legacy application has been developed based on a more detailed requirements specification that is available along with the application source code (MinnesotaIncomeTaxCalculation-Requirements.pdf). In a first step, study the documentation to get more information concerning the application's architecture and use cases.
2. Do a mock installation: The application source code is provided as an eclipse project (2023-IncomeTaxCalculatorProject folder). Setup a running version of the project and try to use its basic functionalities.
3. Read all the source code once and try to understand the legacy architecture, the role/responsibilities of each class, and so on.

INITIAL UNDERSTANDING

1. Specify the use cases as they are implemented in the legacy applications.
2. Capture the legacy architecture in terms of a UML package diagram.
3. Specify the detailed design in terms of UML class diagrams.

	Date 10/6/2022
--	----------------

REENGINEERING

1. Implement tests for the use cases of the application.
2. Detect and report issues for refactoring/restructuring. In particular, you can look for Duplicate Code, God Classes, and Navigation Code. Reengineer the code to fix these issues.
3. Migrate the application towards a Spring Web-based architecture to improve the front-end and the overall usability of the application.