**Final Year Project-I Evaluation Report**



December 10, 2014



*Making programmer’s life Better*

**Group Members:**

Asad Ali Azam (11i – 0100)

Muhammad Umair Khan (11i – 0058)

Bilal Amjad (11i – 0110)

**Supervisor:**

Mr. Atif Aftab Ahmed Jillani

# Abstract

Many approaches for software development have been presented in order to make software understandable and programmer friendly. Object Oriented programing approach is one of the most successful and most widely used approach in the current industry because software built through this approach are easy to understand.

Object Oriented Programming was not a widely used approach in the early years but it has caught on because of its more programmer-friendly process. Object oriented programming uses real life objects as its classes, which makes it closer to the real world.

As most of the software systems being used by organizations were created using non-object oriented programming, so they are difficult to understand and cannot be re-used. That’s why, a software tool is needed which would be capable of converting a non-Object Oriented Program into an Object Oriented one.

Table of Contents

Contents

[1 Abstract 1](#_Toc405934179)

[1 Introduction 4](#_Toc405934180)

[2 Literature Review 5](#_Toc405934181)

[3 Project Vision 6](#_Toc405934182)

[3.1 Problem Statement 6](#_Toc405934183)

[3.2 Business Opportunity 6](#_Toc405934184)

[3.3 Objectives 6](#_Toc405934185)

[3.4 Project Scope 6](#_Toc405934186)

[3.5 Constraints 6](#_Toc405934187)

[3.5.1 Language Constraints: 6](#_Toc405934188)

[3.5.2 Time Constraints: 7](#_Toc405934189)

[Software Constraints: 7](#_Toc405934190)

[3.6 Stakeholder 7](#_Toc405934191)

[3.6.1 Stakeholder Summary 7](#_Toc405934192)

[3.6.2 User Summary 7](#_Toc405934193)

[3.6.3 High Level Goals and Problems 8](#_Toc405934194)

[4 Software Requirement Specification 9](#_Toc405934195)

[4.1 List of Features 9](#_Toc405934196)

[4.2 Functional Requirements 9](#_Toc405934197)

[4.2.1 User Services 9](#_Toc405934198)

[4.2.2 System related Requirements 9](#_Toc405934199)

[4.3 Non- Functional Requirements 9](#_Toc405934200)

[5 High Level Use Case 10](#_Toc405934201)

[5.1 Use Case Diagram 11](#_Toc405934202)

[6 Iteration Plan 12](#_Toc405934203)

[7 Iteration I 14](#_Toc405934204)

[7.1 Expanded Use Case 14](#_Toc405934205)

[7.2 Expanded Use Case 15](#_Toc405934206)

[7.3 Activity Diagram 16](#_Toc405934207)

[7.3.1 Generate Data Flow Log 16](#_Toc405934208)

[7.3.2 Generate Class Diagram log: 17](#_Toc405934209)

[7.4 Domain Model 18](#_Toc405934210)

[7.5 System Sequence Diagram 19](#_Toc405934211)

[7.6 Operation Contract 21](#_Toc405934212)

[7.6.1 Load Code: 21](#_Toc405934213)

[7.6.2 Load Data Flow log: 21](#_Toc405934214)

[7.7 Sequence Diagram 22](#_Toc405934215)

[7.8 Class Diagram 24](#_Toc405934216)

[7.9 Architecture Diagram 25](#_Toc405934217)

# Introduction

NO4 is a software tool which will assist developers in understanding and reuse software by converting hard to understand software into easy to understand and programmer friendly software. This software tool will convert poorly designed software into a well written program thus increasing its usability and understandability.

# Literature Review

**Why OOP?**

Since the evolution of Computer Programming, Software reusability has been a challenging task for developers. The software built through structured programming approach works well but its drawbacks are clearly visible when it comes to understanding and extending their functionalities.

In 1960s, concept of Object Oriented Programming was introduced and since then it has been one of the most successful programing paradigm because of its programmer friendly nature. Object Oriented Programming revolves around the concept of relating programming data and functions to real world objects making the program easier to understand and manipulate.

Since the inception of Object Oriented approach, a number of advancements has been made using this approach in design patterns, modeling languages etc. and it has continued to improve. Even more, research is going on in the domain of object oriented paradigm for its further refinement and advancement. Keeping in view all these facts, it can be predicted that Object Oriented Programming will continue to evolve in the future and so developers now prefer to adopt object oriented approach for the software development.

**How NO4 will influence organizations?**

As discussed earlier, Object Oriented Programming approach is the most popular programming approach in today’s world and there are still software present which were built through non object oriented approach. So, a need of a software tool can be felt in industry which would be capable of transforming a program into object oriented program and So, NO4 is the tool which will serve this purpose. This tool will be capable of incorporating object oriented approach in software thus helping organizations to extend functionalities of their existing software thus saving company’s resources.

# Project Vision

## Problem Statement

Software understandability has been one of the most challenging tasks for the developers and designers. The need for software understandability arises when some person needs to extend functionality or reuse some part of the software which was actually made by some other person. Therefore, a software tool is needed which would be capable of converting a hard to understand program to a more user friendly program in order to make the program easier to understand.

## Business Opportunity

In many firms, when a software lacks some functionality or because of some new business demand, it is usually replaced with a new one which is very costly and inconvenient. It is so because developers usually fail to extend functionality of the existing software because that software is too hard to understand. So the firm has no choice than to replace the existing software with a new one.

So, to keep firms from consuming resources on a new software, NO4 would be used to convert the existing software to an understandable program which would it easier for the developer to understand.

## Objectives

* Software program could be graphically displayed.
* Conversion of hard to understand program into a programmer friendly program.
* Making a software program reusable.
* Incorporation of best programming practices in existing software.

## Project Scope

This software tool will convert a software program into Object Oriented program.

## Constraints

### Language Constraints:

JAVA will be the programming language used (Net beans)

### Time Constraints:

Software will be developed within 7 months (maximum).

Software Constraints:

1. We will deal with syntax only not with the semantic.
2. A structured program without proper naming conventions cannot be converted into a quality object oriented program.
3. The user of this tool must have some programming skills for its proper usage.

## Stakeholder

The identification of stakeholders in the requirement phase is necessary because they will be the ones who will be using your final product.

Market Demographics:

This tool is mainly made for the existing organizations that have a hard time in understanding or adding functionality to their old software systems due to the code being structural.

### Stakeholder Summary

There are a number of stakeholders with an interest in the development

|  |  |  |
| --- | --- | --- |
| Name | Description | Responsibilities |
| Organization’s Top Level Management | The CEO or the Manager of the firm. | * Approve funding for the project * Analyze use of the software. |
| IT Head | The IT manager of the organization purchasing the solution | * Will use software to turn the or firm’s old software into object oriented. |

### User Summary

The direct users of the system would be

|  |  |  |
| --- | --- | --- |
| Name | Description | Responsibilities |
| IT Head | The Organization’s IT head who will use the product | * Transform code for better understanding. |
| High Level Managers | The managers of the organization. | * Monitors the work |

### High Level Goals and Problems

#### Concerning Stakeholders

|  |  |  |  |
| --- | --- | --- | --- |
| High Level Goal | Priority | Problems and Concerns | Proposed Solutions |
| Easy to use | High | The business minded staff may not be able to understand the tool | Making the tool as easy to use as possible. Also provide a step-by-step tutorial. |
| Return on Investment | High | The tool may have more cost than developing a new system. | The tool will use built-in features that will not be costly at all. |
| Reliability | High | The object oriented program may not be the best solution for the problem. | The tool will use different noun and verb phrasing techniques which will produce a programmer friendly code. |

# Software Requirement Specification

## List of Features

* Extract design from structural code.
* Conversion of design to class diagram.
* Conversion of class diagram to object oriented code.

## Functional Requirements

Functional requirements are provided below for a better understanding of the system:

### User Services

* User can give the system an input file having structural code.
* User can skip directly to part two if he has the log file of design in the right format
* User can also input a class diagram for code extraction process.

### System related Requirements

* System will detect nouns and verbs from a given code.
* System will convert structural code into a log file having design.
* System will convert a log file into a class diagram.
* System will convert class diagram into object oriented code.

## Non- Functional Requirements

|  |  |
| --- | --- |
| Property | Metric |
| Performance | The performance will differ along with the complexity and length of the code. |
| Size | The application size will be no more than 100 MB. |
| Ease of Use | The graphic user interface will be very user friendly. |
| Reliability | The tool will use all the rules possible to make the extracted code reliable and programmer friendly. |

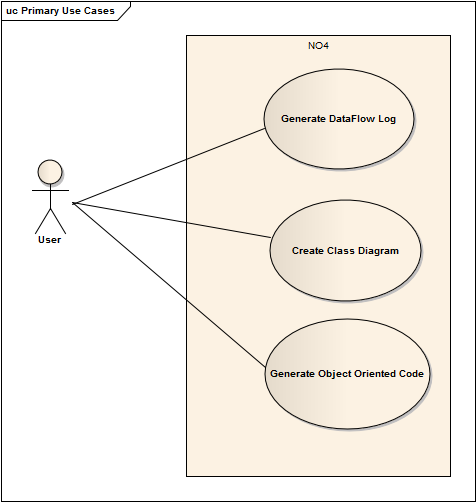
# High Level Use Case

|  |  |
| --- | --- |
| High Level Use Case : | Generate data Flow Log |
| Use case: | Generate Log |
| Actors : | User |
| Type : | Primary |
| Description : | A User takes a structured code and loads it into the tool. The system asks the user to wait while it processes the code. The system generates a log file and gives the user the path to it. The user opens the log file containing data flow and goes for the next phase. |

|  |  |
| --- | --- |
| High Level Use Case: | Create Class Diagram |
| Use Case : | Generate class diagram |
| Actors : | User |
| Type : | Primary |
| Description : | A User takes the Data flow log file and loads the log file path into the tool. System asks the user to wait while it processes. The system generates a Class diagram and gives the user its file path. The user gets the class diagram file and leaves for next phase. |

|  |  |
| --- | --- |
| High Level Use Case : | Generate Object Oriented Code |
| Use Case : | Generate Code |
| Actors : | User |
| Type : | Primary |
| Description : | A User takes a class diagram file and loads the file path into the tool. System asks user to wait while it processes the file. The system then generates an object oriented code and gives the user the file path to it. The user gets the object oriented code file and exits the tool. |

## Use Case Diagram



# Iteration Plan

**Iteration 1:**

Software system will be able to take in a non-object oriented code and examine the flow of control in the code. This process will enable our Software system to write the flow of the program in a file that can be translated into a DFD.

**Iteration 2:**

Software system will perform noun phrasing process on the code which will involve examining each variable and function name.

**Iteration 3:**

On the basis of the data collected on the noun-phrasing process, our software will identify classes.

**Iteration 4:**

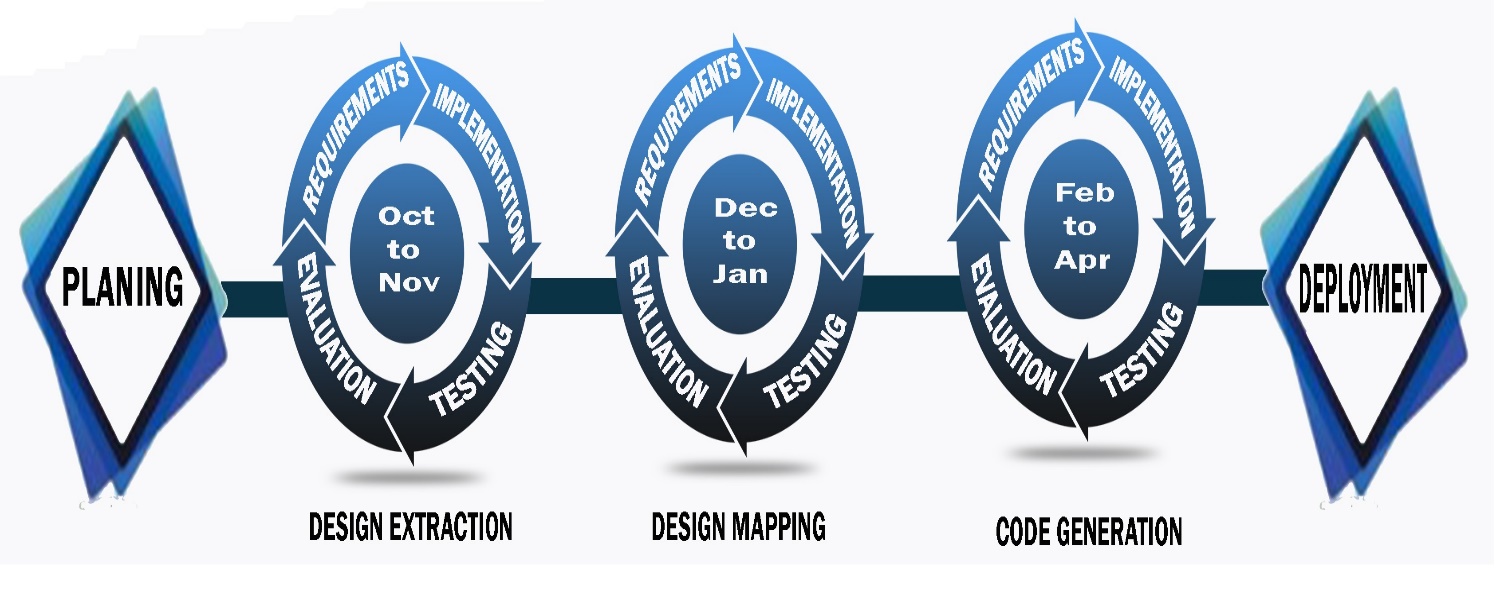
Software system will map each function and variable to its relative class.

**Iteration 5:**

Software system will be programmed with a number of rules to convert the information extracted from above iterations into a class diagram

**Iteration 6:**

The Class diagram will be used to generate a fully working object oriented code.



# Iteration I

## Expanded Use Case

Use case

**Generate Data Flow Log**

**Scope:** Structured code to Data flow

**Level:** User Goal

**Primary Actor:** User

**Stakeholders:** Developer

**Preconditions:** Code should be in structured format.

**Success Guarantee:**

1. Functions are identified
2. Flow of functions are observed
3. Nature of function identified

**Main success scenario:**

|  |  |
| --- | --- |
| User actions | System responsibility |
| 1. User opens the tool and loads the structured code into the tool. |  |
|  |  |
|  | 1. System asks the user to wait while processing. |
|  | 1. System responds with desired data flow log and saves it in the directory. |
| 1. User gets the log file directory. |  |

**Alternative Flows:**

1. The code entered is not structured. No record found.

1a. the naming convention is not properly formatted.

## Expanded Use Case

Use case

**Generate Class diagram Log**

**Scope:** Extracting classes from Data flow log

**Level:** User Goal

**Primary Actor:** User

**Stakeholders:** Developer

**Preconditions:** Data flow log is created from the structured code.

**Success Guarantee:**

1. Classes are identified
2. Linkage between classes id observed.
3. Relation between classes is observed and created

**Main success scenario:**

|  |  |
| --- | --- |
| User actions | System responsibility |
| 1. User loads the data flow log and asks for the class diagram log. |  |
|  |  |
|  | 1. System asks the user to wait while processing. |
|  | 1. System responds with desired class diagram flow log and saves it in the directory. |
| 1. User gets the log file directory. |  |

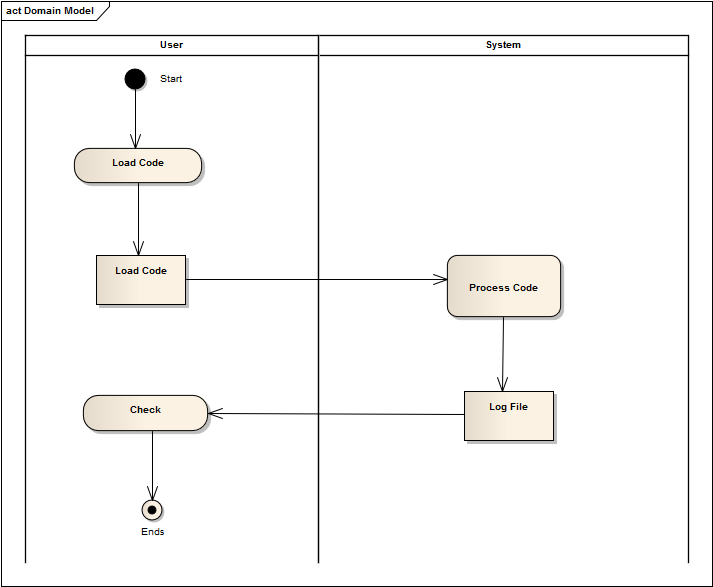
**Alternative Flows:**

1. The data log file is not created. No record found.

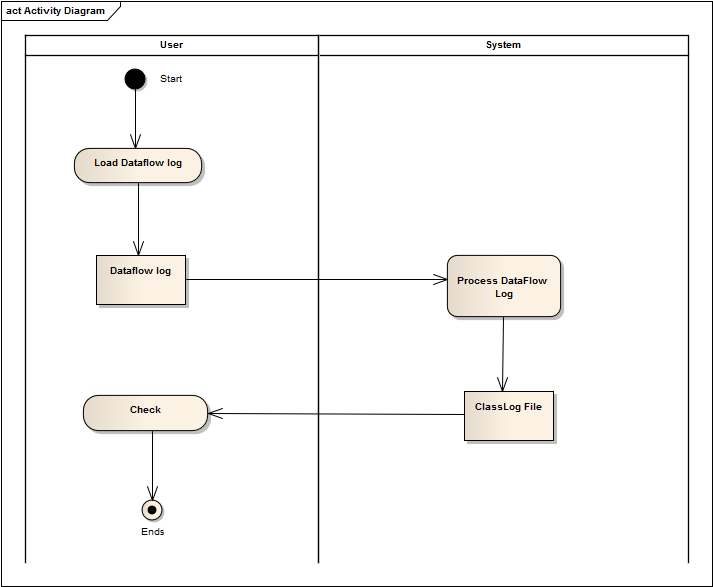
1a. the log file is not properly formatted.

## Activity Diagram

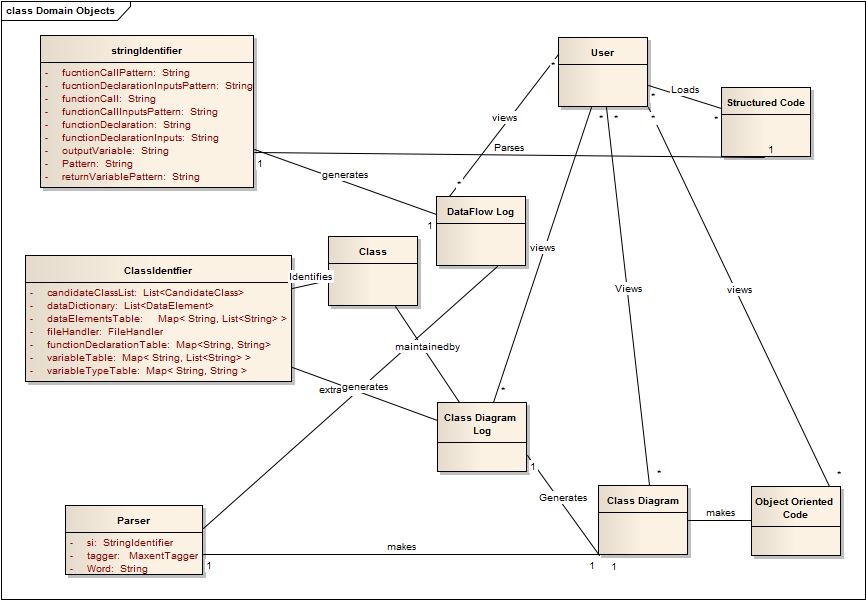
### Generate Data Flow Log



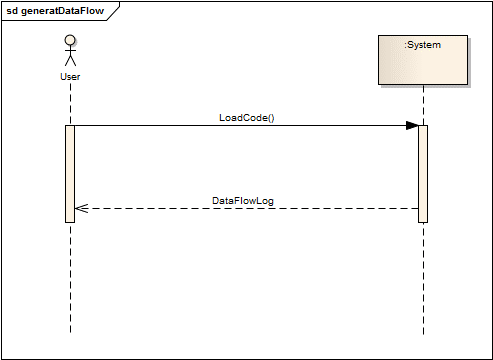
### Generate Class Diagram log:

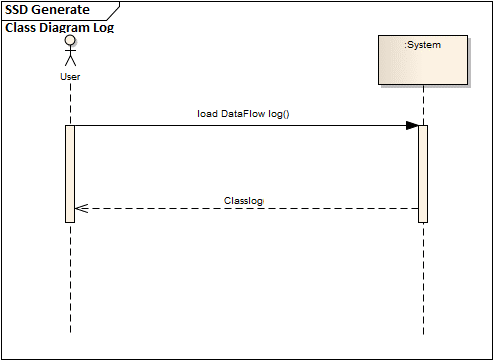


## Domain Model



## System Sequence Diagram





## Operation Contract

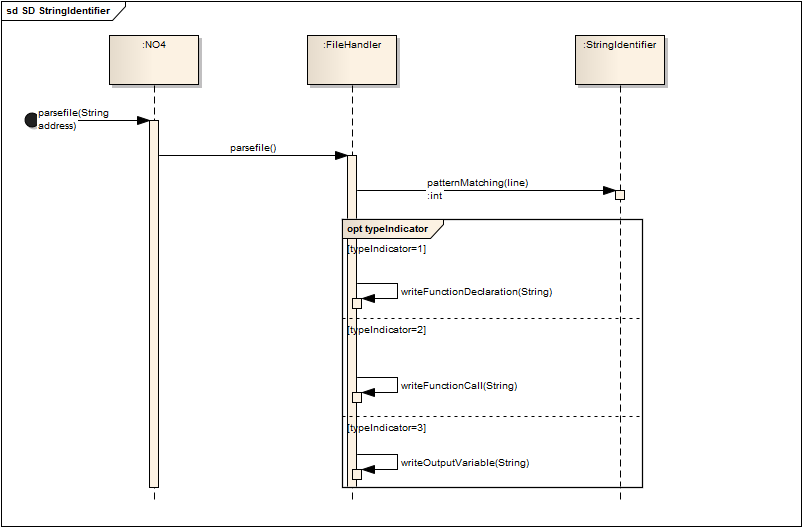
### Load Code:

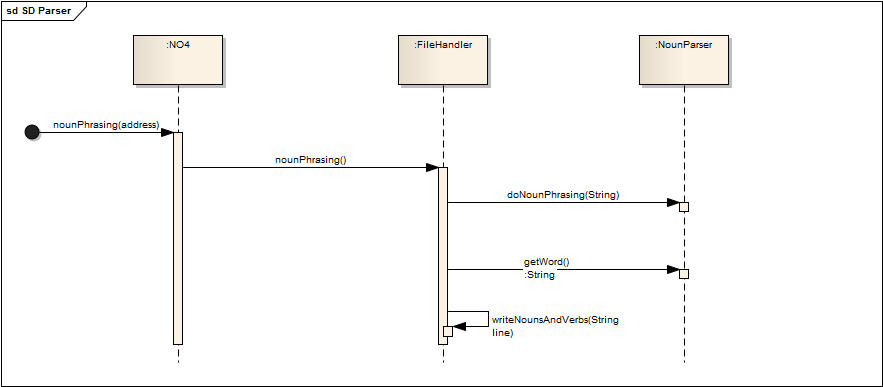
|  |  |
| --- | --- |
| Name : | Load Code() |
| Responsibility: | Generate a log file containing data flow |
| Type : | System |
| Cross reference: | Use Case: Generate Data Flow Log |
| Pre-Conditions : | Code should be in structured format. |
| Post-Condition : | 1. Instance of File Handler has been created. 2. Instance of String Identifier has been created. 3. Association of file handle and String Identifier has been created. |

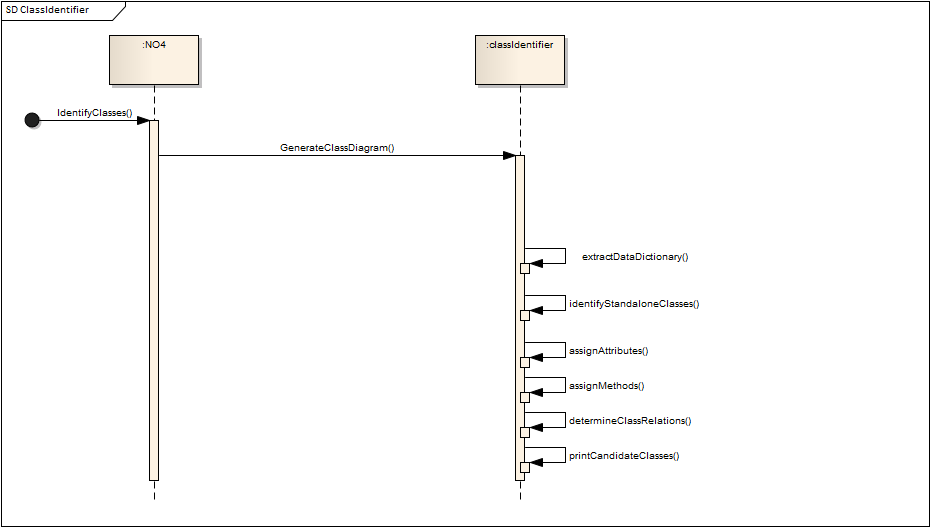
### Load Data Flow log:

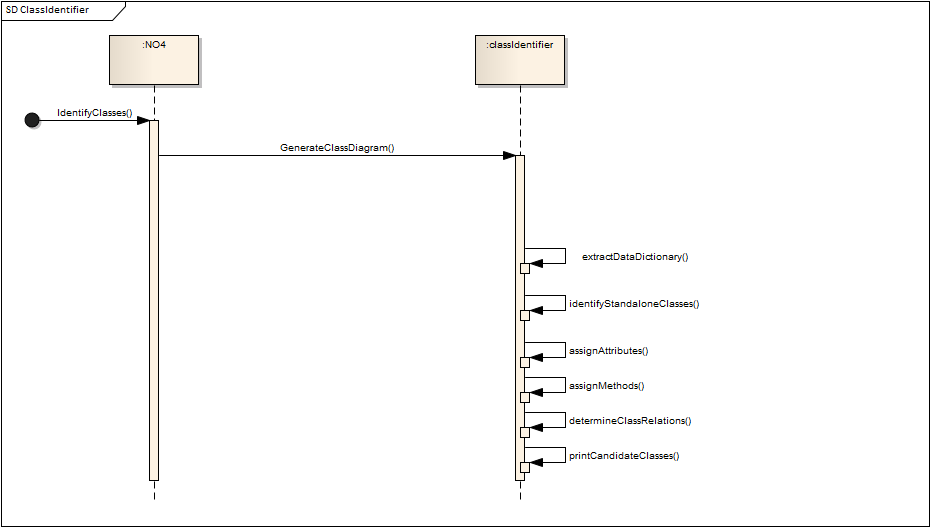
|  |  |
| --- | --- |
| Name : | Load Data Flow Log() |
| Responsibility: | Generate a log file containing class diagram flow |
| Type : | System |
| Cross reference: | Use Case: Generate Class Diagram Log |
| Pre-Conditions : | Data Flow log is generated or maintained. |
| Post-Condition : | 1. Instance of File Handler has been created. 2. Instance of Class Identifier has been created. 3. Association of file handle and Class Identifier has been created. |

## Sequence Diagram

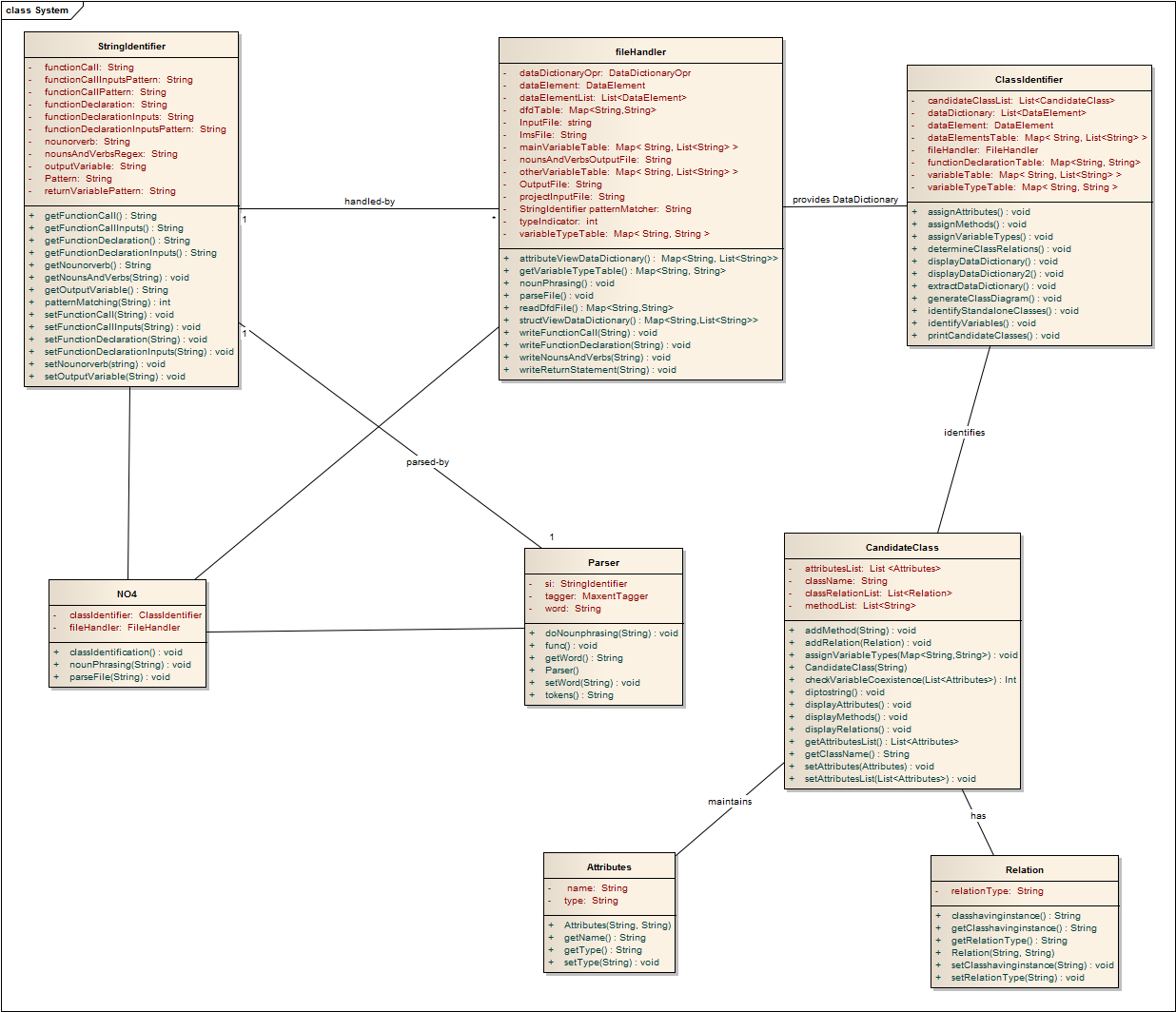








## Class Diagram



## Architecture Diagram

