**Final Year Project Evaluation Report**



June 18, 2015



*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

# Acknowledgement

NO4 is a research based tool which is developed by the Team NO4. With the grace of Allah Almighty and with the help of our supervisor, we have completed the project as per described in scope. No such system exists that provides the same functionality as NO4, so it is a copyrighted product and all the rights are reserved with the team and the university.

**Group Members:**

Bilal Amjad (i110110) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Muhammad Umair Khan (i110058) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Asad Ali Azam (i110100) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Supervisor**

Mr. Atif Aftab Ahmad Jillani

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Anti-Plagiarism Declaration

This is to declare that all the publications produced under the title ‘NO4’ are the sole contributions of the undersigned and no parts hereof have been reproduced on as it is basis (plagiarized). I am conscious that the incorporation of material from other works or a paraphrase of such material without acknowledgement will be treated as plagiarism. All referenced parts have been cited properly. In case of any violation of this declaration, the undersigned will be held responsible.

**Date:** 17th June, 2015

**Group Members:**

Bilal Amjad (11i-0110) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Muhammad Umair Khan (11i-0058) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Asad Ali Azam (11i-0100) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 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

[Anti-Plagiarism Declaration 2](#_Toc422255577)

[Abstract 3](#_Toc422255578)

[1 Introduction 6](#_Toc422255579)

[2 Literature Review 7](#_Toc422255580)

[3 Project Vision 8](#_Toc422255581)

[3.1 Problem Statement 8](#_Toc422255582)

[3.2 Business Opportunity 8](#_Toc422255583)

[3.3 Objectives 8](#_Toc422255584)

[3.4 Project Scope 8](#_Toc422255585)

[3.5 Constraints 8](#_Toc422255586)

[3.5.1 Language Constraints: 8](#_Toc422255587)

[3.5.2 Time Constraints: 8](#_Toc422255588)

[Software Constraints: 9](#_Toc422255589)

[3.6 Stakeholder 9](#_Toc422255590)

[3.6.1 Stakeholder Summary 9](#_Toc422255591)

[3.6.2 User Summary 9](#_Toc422255592)

[3.6.3 High Level Goals and Problems 10](#_Toc422255593)

[4 Software Requirement Specification 11](#_Toc422255594)

[4.1 List of Features 11](#_Toc422255595)

[4.2 Functional Requirements 11](#_Toc422255596)

[4.2.1 User Services 11](#_Toc422255597)

[4.2.2 System related Requirements 11](#_Toc422255598)

[4.3 Non- Functional Requirements 11](#_Toc422255599)

[5 High Level Use Case 12](#_Toc422255600)

[5.1 Use Case Diagram 13](#_Toc422255601)

[6 Iteration Plan 14](#_Toc422255602)

[7 Algorithm 16](#_Toc422255603)

[7.1 Algorithm for generating Object Oriented Code from a structured code 16](#_Toc422255604)

[8 Iterations 18](#_Toc422255605)

[8.1 Expanded Use Case 18](#_Toc422255606)

[8.2 Expanded Use Case 19](#_Toc422255607)

[8.3 Expanded Use Case 20](#_Toc422255608)

[8.4 Expanded Use Case 21](#_Toc422255609)

[8.5 Expanded Use Case 22](#_Toc422255610)

[8.6 Activity Diagram 23](#_Toc422255611)

[8.6.1 Data Flow Log 23](#_Toc422255612)

[8.6.2 Data Dictionary log: 24](#_Toc422255613)

[8.6.3 Class Diagram log: 25](#_Toc422255614)

[8.6.4 Generate Object Oriented Code: 26](#_Toc422255615)

[8.7 Domain Model 27](#_Toc422255616)

[8.8 System Sequence Diagram 28](#_Toc422255617)

[8.8.1 Data Flow Log 28](#_Toc422255618)

[8.8.2 Class Diagram Log 29](#_Toc422255619)

[8.8.3 Generate Object Oriented Code 30](#_Toc422255620)

[8.9 Operation Contract 31](#_Toc422255621)

[8.9.1 dataExtraction(): 31](#_Toc422255622)

[8.9.2 classIdentification(): 31](#_Toc422255623)

[8.9.3 generateCode(): 32](#_Toc422255624)

[8.10 Sequence Diagram 33](#_Toc422255625)

[8.10.1 Parse File 33](#_Toc422255626)

[8.10.2 Class Identification 34](#_Toc422255627)

[8.10.3 Main Writer 35](#_Toc422255628)

[8.10.4 Function Writer 35](#_Toc422255629)

[8.10.5 Generate Code 36](#_Toc422255630)

[8.11 Class Diagram 37](#_Toc422255631)

[9 Screenshots: 38](#_Toc422255632)

[9.1 Splash Screen 38](#_Toc422255633)

[9.2 Project and File Selection 39](#_Toc422255634)

[9.3 Input File 40](#_Toc422255635)

[9.4 Data Flow Log 41](#_Toc422255636)

[9.5 Data Dictionary Log 42](#_Toc422255637)

[9.6 Class Diagram Log 43](#_Toc422255638)

[9.7 UML Model 44](#_Toc422255639)

[9.8 Object Oriented Code 45](#_Toc422255640)

[10 Architecture Diagram 46](#_Toc422255641)

# Introduction

With the advancement is technology in all aspects of our lives, there is a growing need for applications and tools that make the life of a common man easier. This growing need is being fulfilled by thousands of developers and software engineers who are working hard to develop applications, tools, systems, API’s, drivers and hundreds of other things that will help a common user to achieve his goals easily and efficiently, but the problem that arises is that to develop something, you need to write a code in any programming language and nowadays, with the rise in the complexity of systems, the code becomes complex too.

So to conquer the above problem, the concept of Object Oriented Programming was introduced and it has been one of the most successful programing paradigms 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.

The systems that were developed before the paradigm shift to Object Oriented Development are still under use in some organizations while others have replaced them by spending huge amounts of money. So, there is a need of a software tool that would be capable of transforming an old program into an object oriented program.

NO4 is a software tool which will assist developers in understanding and reusing 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. The software will also provide the users the facility to visualize the design of the converted code this making their job easier.

# Literature Review

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 have 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.

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. |

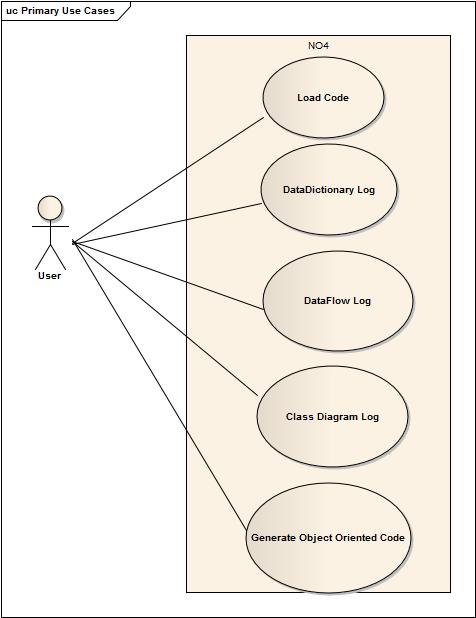
# High Level Use Case

|  |  |
| --- | --- |
| 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

1.       Generation of DFD.

2.       Class identification and attributes assigning

3.       Assignment of methods to classes

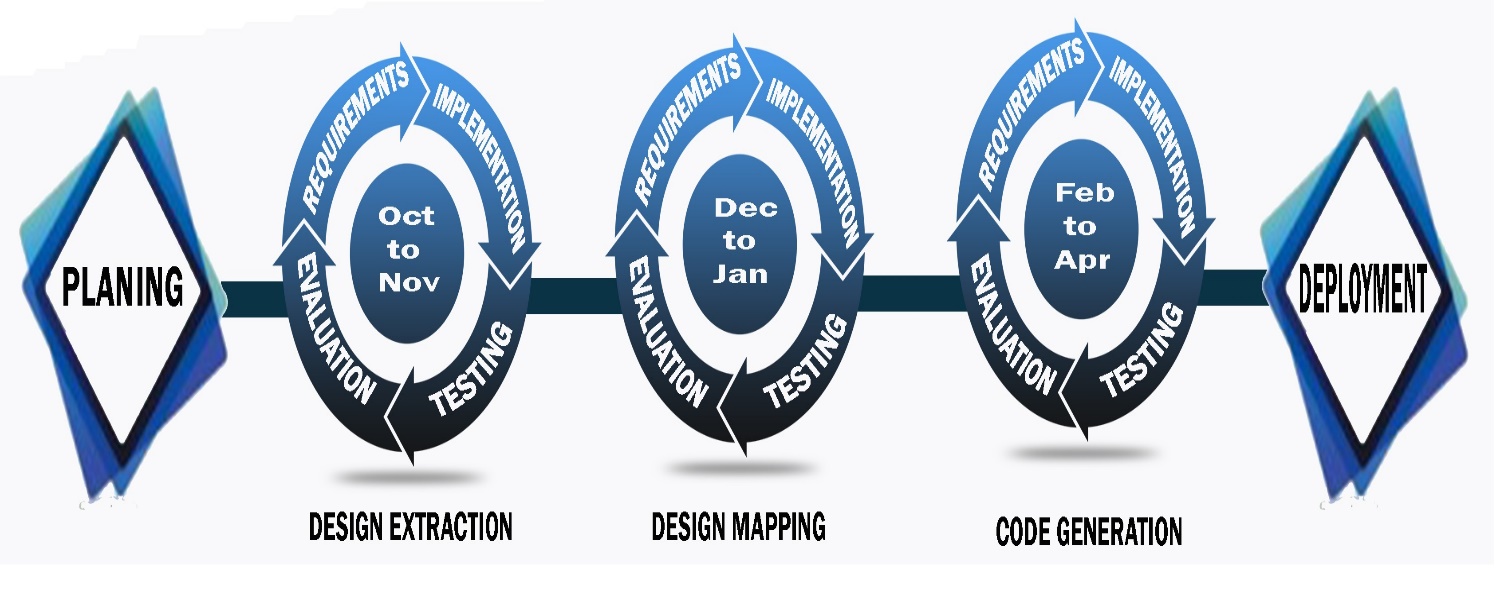
4.       Graphical representation of class diagram

5.       Auto Generation of data dictionary

6.     Skeleton Code generation

7.       Making of ‘main’ function compatible with object oriented code

8.       Making functions of structured code compatible with object oriented code



# Algorithm

## Algorithm for generating Object Oriented Code from a structured code

* **Handling of Global Variables**

Scan whole program

**if** (global variable found)

assign global variable to each class

* **Generation of Skeleton code of Object Oriented Program**

Attributes of type int/float/double/String/non-primitive type should be set using "this" pointer in mutators

Attributes of type char should be set using "strcpy" command in mutators

Accessors of attributes of type "char" should be pointer functions

* **Generation of Function-Class HashMap**

Incorporate all the classes and their functions in a hashmap such that:

**Map <String, List<String>>**

where function name is the ‘key’ of the hashmap and class name is the ‘value’

* **Transformation of “main” function**

Scan each statement of "main" function

**if** (assignment found)

**if** ( statement contains dot operator)

transform statement into an accessor/mutator call

append name of the class object (whose method is being called) followed by a dot before accessor/mutator

**else if** (function call found)

initialize class object for the class whose function is being called in main

append class’s object name followed by a dot before each function call

* **Transformation of other functions:**

Scan each statement of the function.

**if** (function call found)

extract class name from function name.

**if** (extracted class name == class name of the function being iterated)

do nothing

**else**

append the class name before the function call with ‘dot’ operator

initialize the object of that class at the start of the function

# Iterations

## Expanded Use Case

Use case

**Load Code**

**Scope:** Load the structure code in the system.

**Level:** User Goal

**Primary Actor:** User

**Stakeholders:** Developer

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

**Success Guarantee:**

Code is loaded and shown in the system.

**Main success scenario:**

|  |  |
| --- | --- |
| User actions | System responsibility |
| 1. User opens the tool and makes a new project and specifies the project path. |  |
|  |  |
|  | 1. System asks the user for the source code path. |
| 1. User loads the code giving the source code path. |  |
|  | 1. System responds with the given source code. |

**Alternative Flows:**

1. The path does not exists. No record found.

1a. the file does not exists.

## Expanded Use Case

Use case

**Display 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

**Display Data Dictionary Log**

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

**Level:** User Goal

**Primary Actor:** User

**Stakeholders:** Developer

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

**Success Guarantee:**

1-Attributes are extracted.

2-Name of classes are identified.

**Main success scenario:**

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

**Display Class diagram Log**

**Scope:** Extracting classes from Data flow logand data dictionary

**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 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 class diagram log and saves it in the directory. |
| 1. User gets the class diagram log file. |  |

**Alternative Flows:**

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

1a. the log file is not properly formatted.

## Expanded Use Case

Use case

**Generate object oriented Code**

**Scope:** Class Diagram log to object oriented code

**Level:** User Goal

**Primary Actor:** User

**Stakeholders:** Developer

**Preconditions:** Class Diagram Log should be generated.

**Success Guarantee:**

1-Classes are made

2-Attributes and objects are made.

3-Object Oriented Code is generated.

**Main success scenario:**

|  |  |
| --- | --- |
| User actions | System responsibility |
| 1. User observes the data flow log, data dictionary and class diagram log is generated. |  |
|  |  |
|  | 1. System asks the user to generate the code. |
|  | 1. System asks the user to wait while code is generated. |
| 1. User gets the object oriented code. |  |

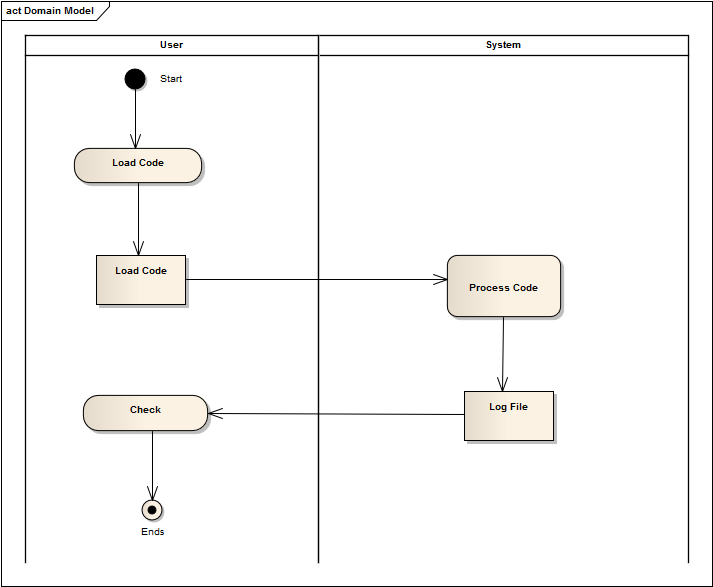
**Alternative Flows:**

1. The data flow log, data dictionary and class diagram log are not generated. No record found.

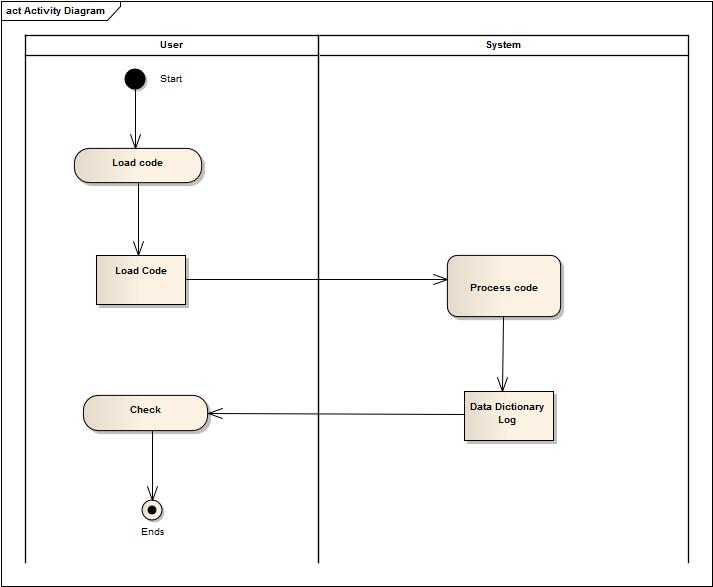
1a. classes are not identified correctly.

## Activity Diagram

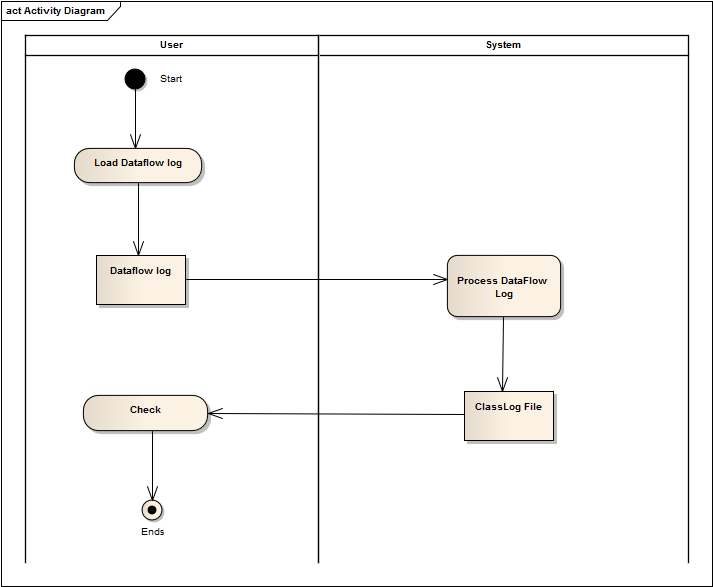
### Data Flow Log



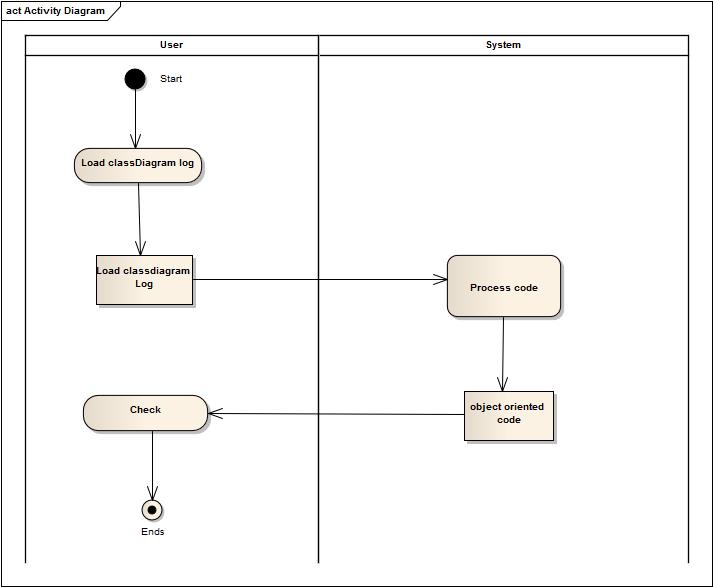
### Data Dictionary log:



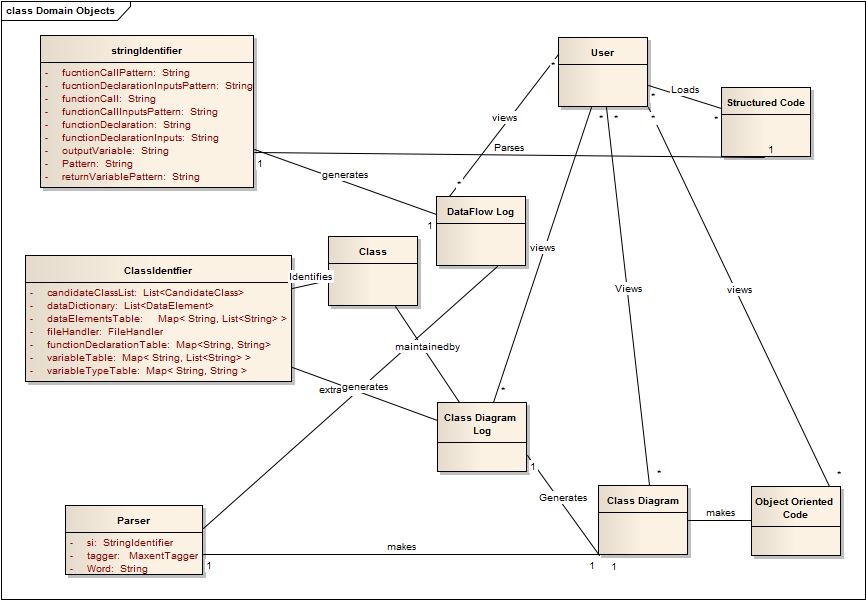
### Class Diagram log:



### Generate Object Oriented Code:

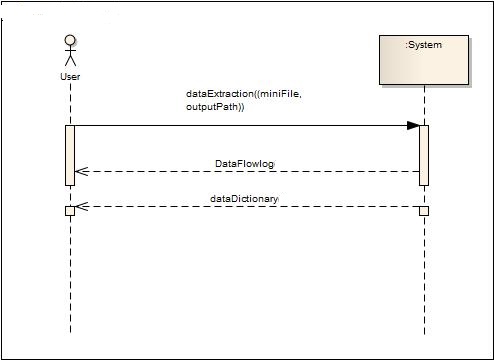


## Domain Model

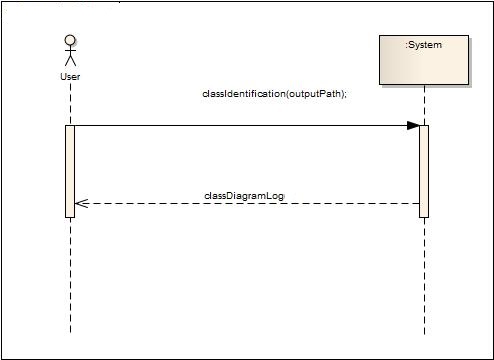


## System Sequence Diagram

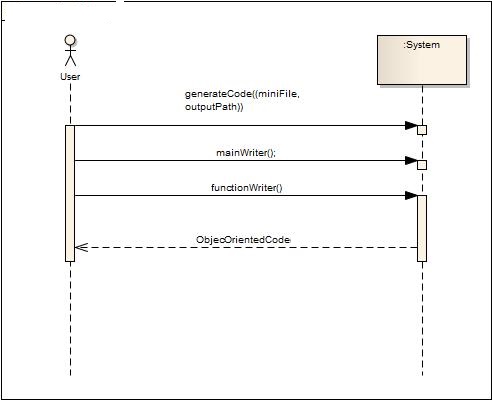
### Data Flow Log



### Class Diagram Log



### Generate Object Oriented Code



## Operation Contract

### dataExtraction():

|  |  |
| --- | --- |
| Name : | dataExtraction() |
| Responsibility: | Generate a log file containing data flow log and data dictionary |
| Type : | System |
| Cross reference: | Use Case: Generate Class Diagram Log |
| Pre-Conditions : | Data Flow log is generated or maintained. |
| Post-Condition : | 1. Instance of regex has been created. 2. Instance of Structs has been created 3. Association of regex and Structs has been created. |

### classIdentification():

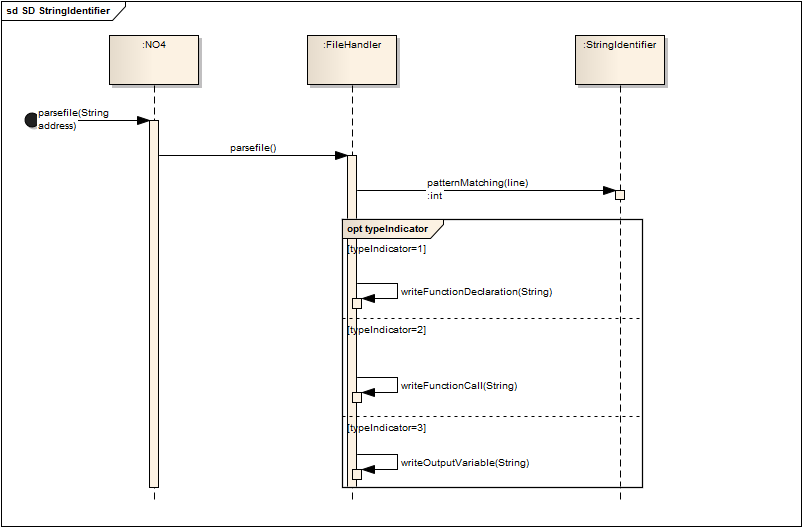
|  |  |
| --- | --- |
| Name : | classIdentification() |
| Responsibility: | Generate a class diagram log |
| Type : | System |
| Cross reference: | Use Case: Generate Class Diagram Log |
| Pre-Conditions : | Data Flow log is generated or maintained.  Data Dictionary is generated and maintained. |
| 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. |

### generateCode():

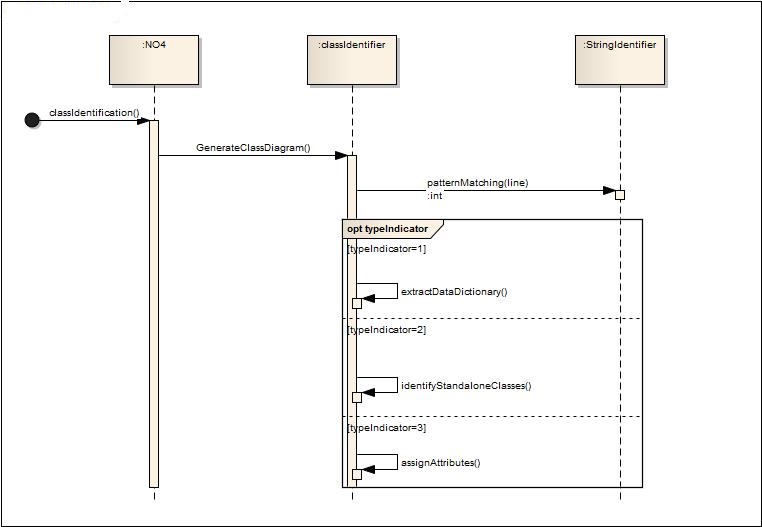
|  |  |
| --- | --- |
| Name : | generateCode() |
| Responsibility: | Generate Object Oriented Code |
| Type : | System |
| Cross reference: | Use Case: Generate object oriented code |
| Pre-Conditions : | Class diagram log is generated or maintained.  Dataflow Log is generated and maintained.  Data Dictionary is generated and maintained. |
| Post-Condition : | 1. Instance of Global Variable detector has been created. 2. Instance of cd Code has been created. 3. Instance of Generate code from cd is has been created. |

## Sequence Diagram

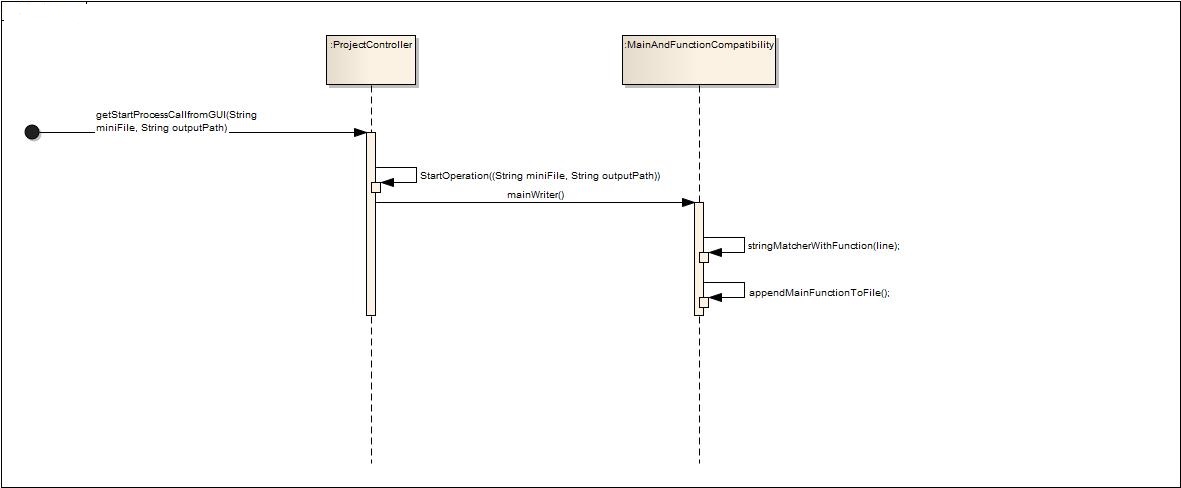
### Parse File



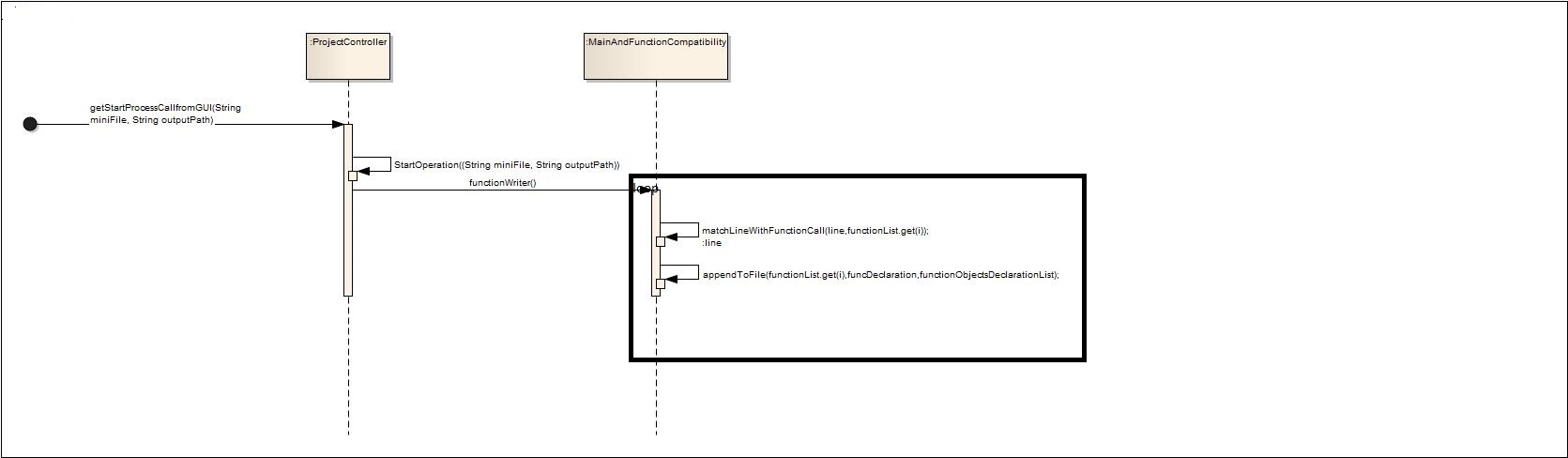
### Class Identification



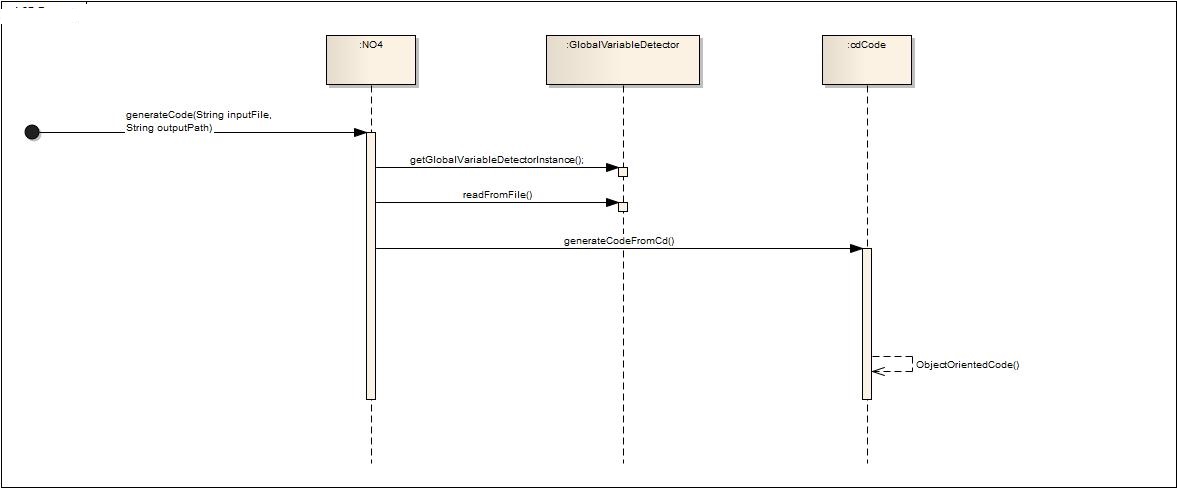
### Main Writer



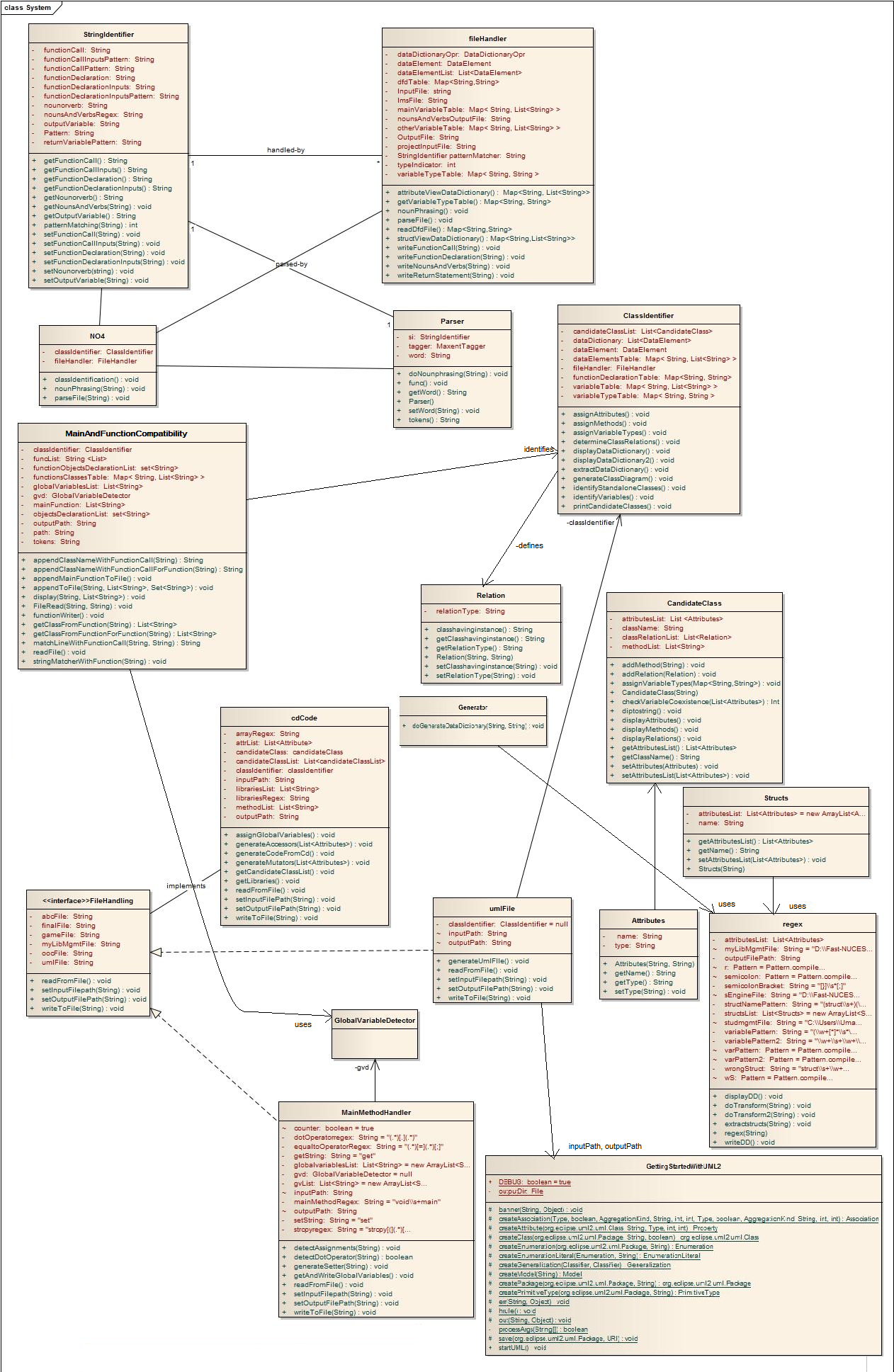
### Function Writer



### Generate Code

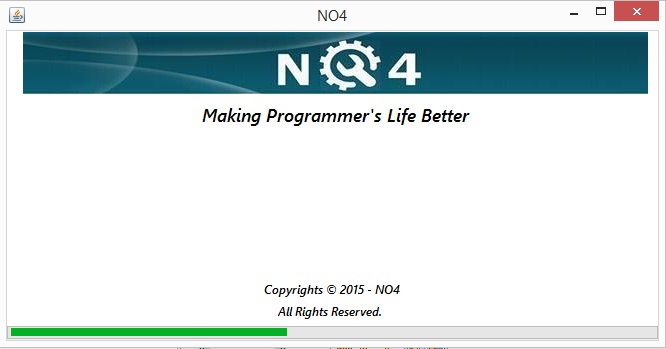


## Class Diagram

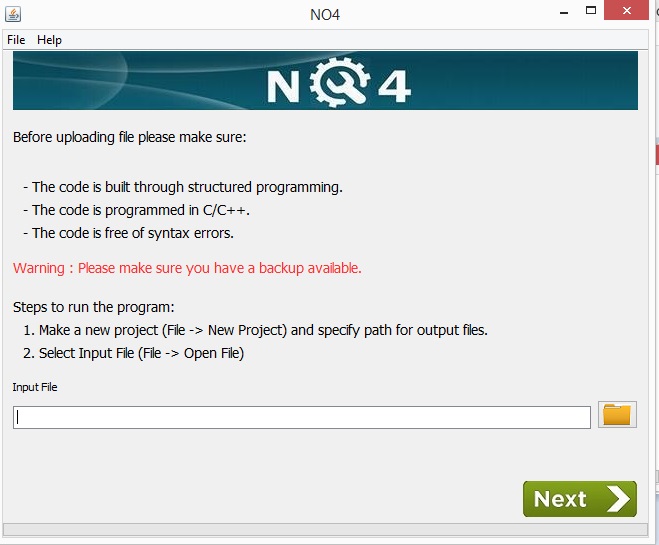


# Screenshots:

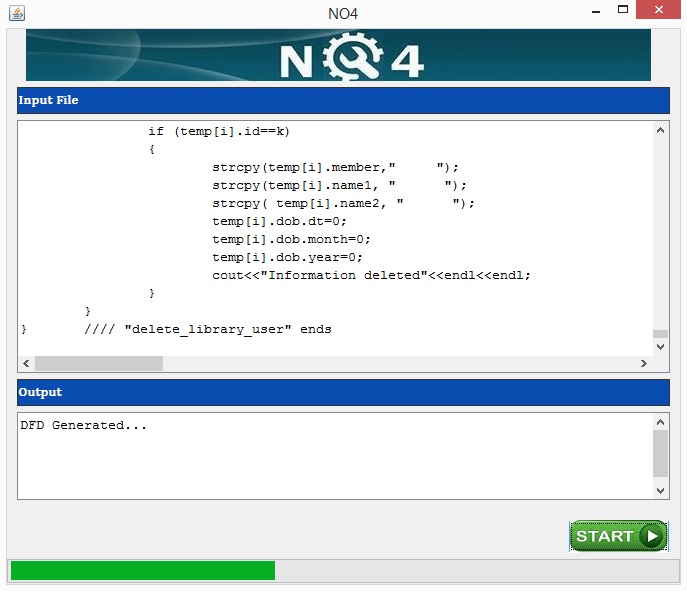
## Splash Screen



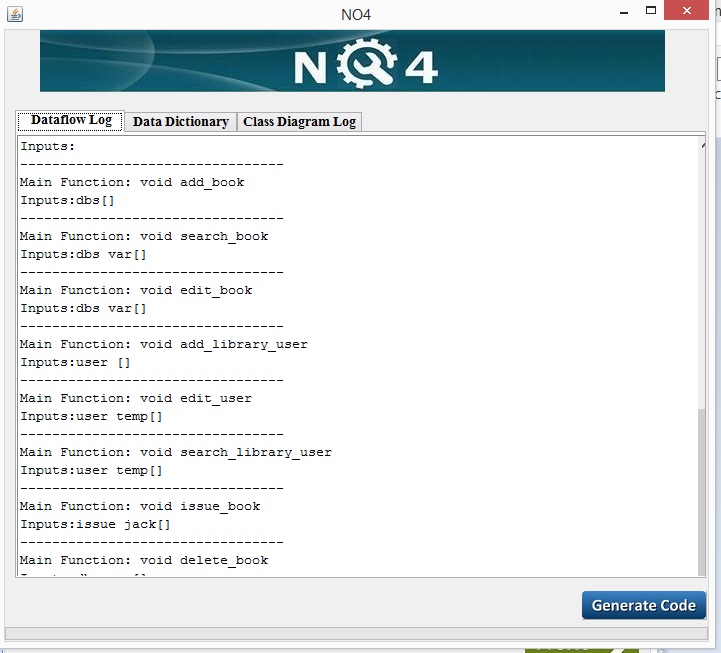
## Project and File Selection



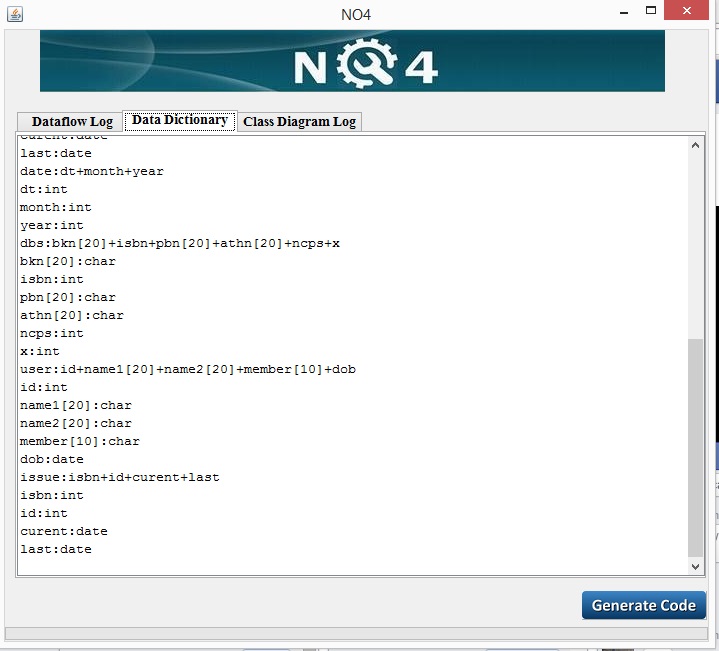
## Input File



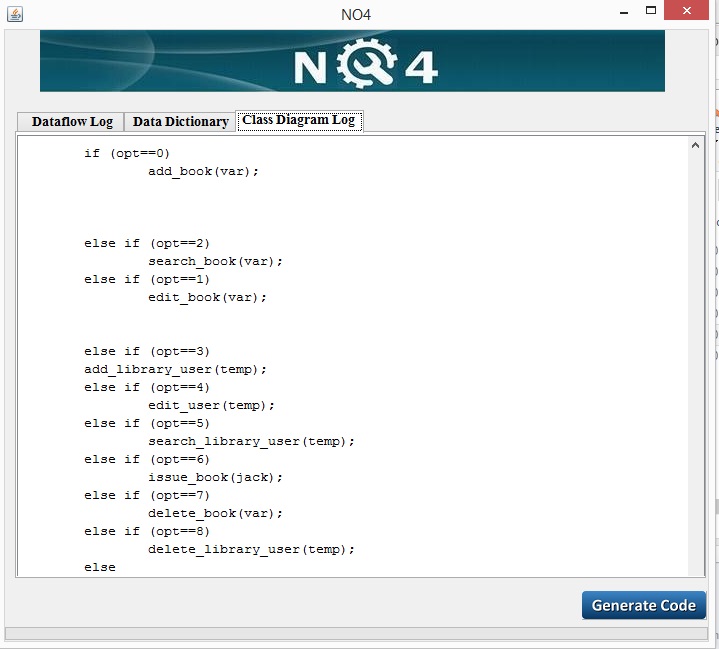
## Data Flow Log



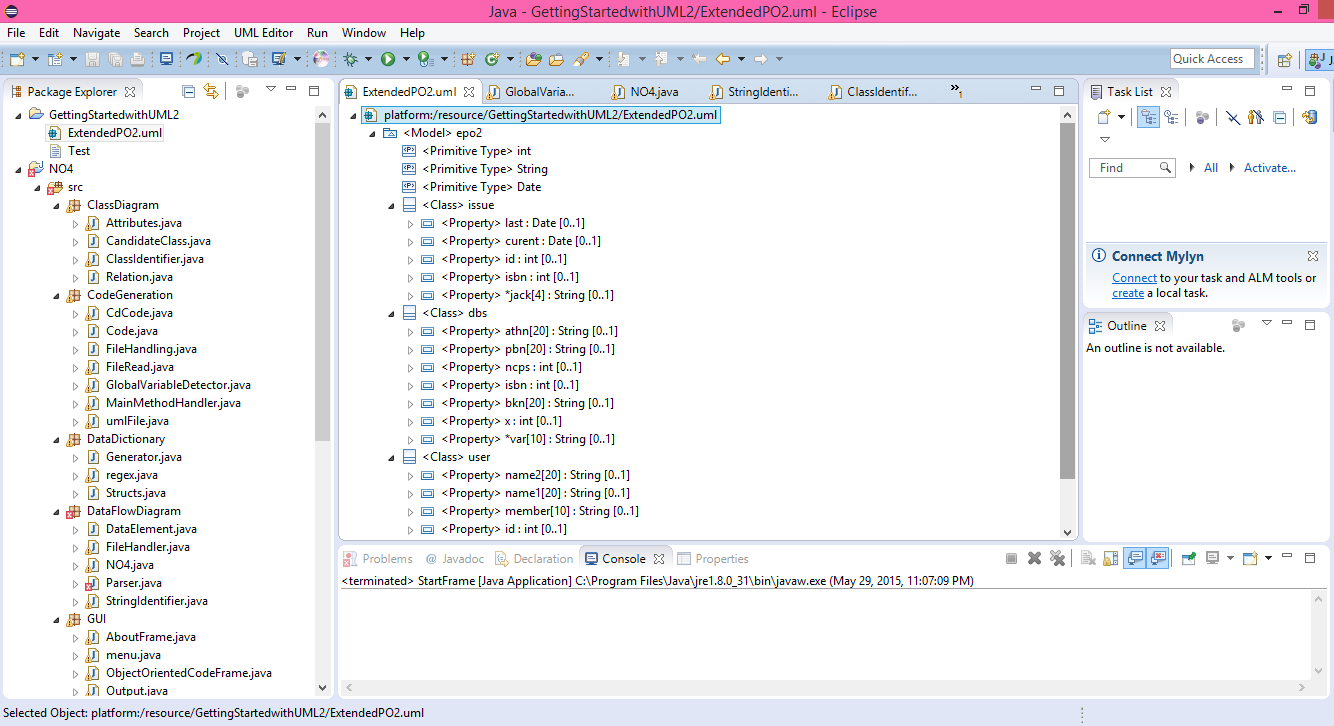
## Data Dictionary Log



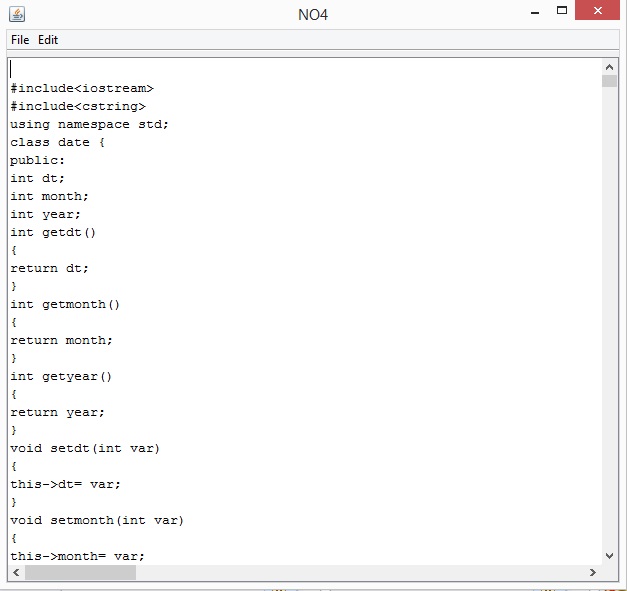
## Class Diagram Log



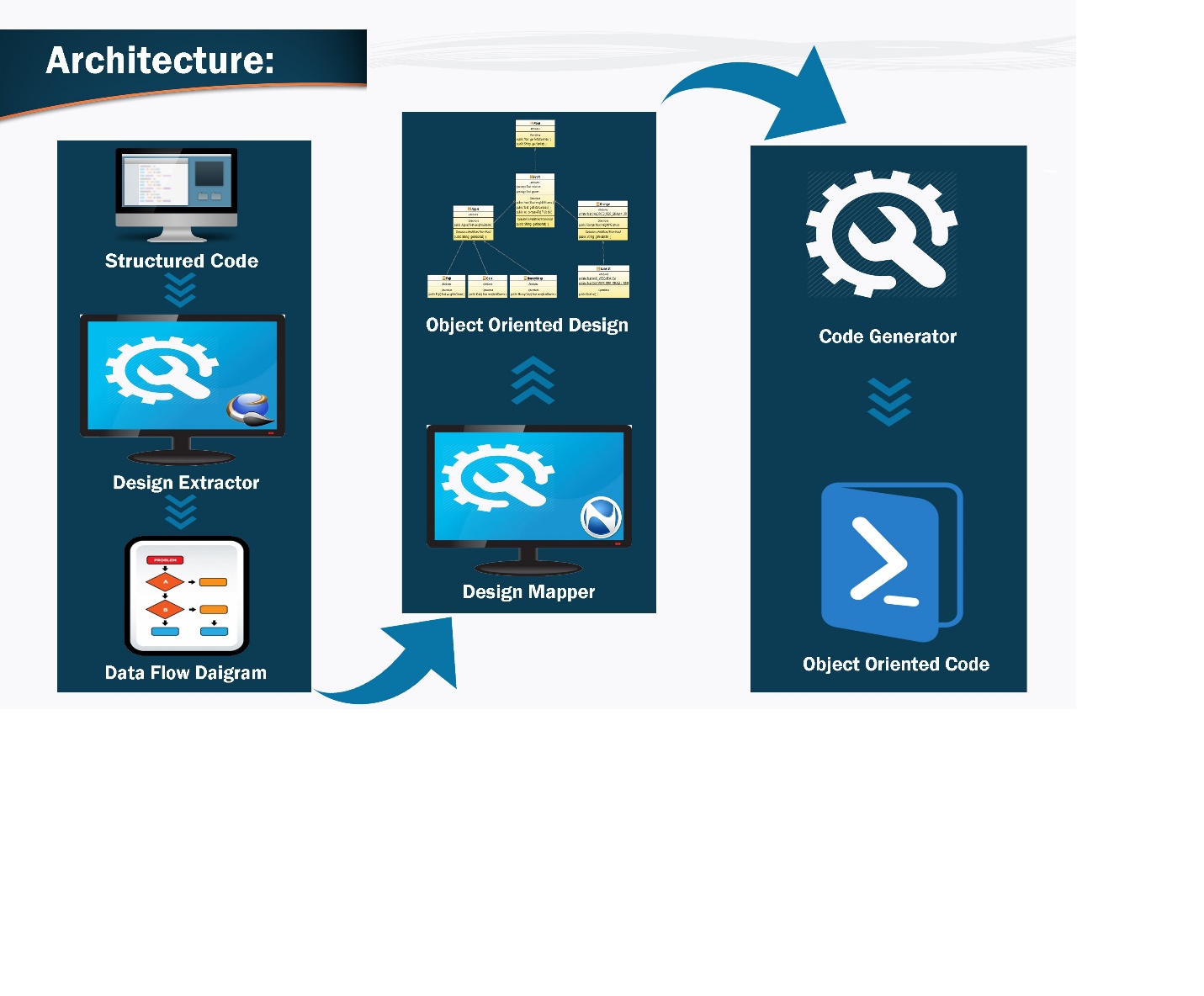
## UML Model



## Object Oriented Code



# Architecture Diagram



# Plagiarism Report

Turnitin Originality Report

Making programmer’s life Better by Bilal Amjad

From Project Reports (Computer Sciences)

* Processed on 15-Jun-2015 11:45 PKT
* ID: 550591900
* Word Count: 2391

Similarity Index

3%

Similarity by Source

Internet Sources:

1%

Publications:

0%

Student Papers:

3%

**sources:**

**1**

2% match (student papers from 10-Dec-2014)

[Submitted to Higher Education Commission Pakistan on 2014-12-10](file:///C:\paperInfo.asp?r=17.7062242149706&svr=02&lang=en_us&oid=488559568&perc=2)

**2**

< 1% match (student papers from 10-Dec-2014)

[Submitted to Higher Education Commission Pakistan on 2014-12-10](file:///C:\paperInfo.asp?r=17.7062242149706&svr=02&lang=en_us&oid=488561294&perc=0)

**3**

< 1% match ()

<http://www.lv.psu.edu/cad18/ist240/handouts/use_case.html>