



Jaypee University of Information Technology
Solan (H.P.)
LEARNING RESOURCE CENTER

Acc. Num. **SP05108** Call Num:

General Guidelines:

- ◆ Library books should be used with great care.
- ◆ Tearing, folding, cutting of library books or making any marks on them is not permitted and shall lead to disciplinary action.
- ◆ Any defect noticed at the time of borrowing books must be brought to the library staff immediately. Otherwise the borrower may be required to replace the book by a new copy.
- ◆ The loss of LRC book(s) must be immediately brought to the notice of the Librarian in writing.

Learning Resource Centre-JUIT



SP05108

MODEL FOR OBJECT ORIENTED LANGUAGE

BY

MOHIT ARORA-051211

PRANSHU SACHAN-051266

UTKARSH JAIN-051248



MAY 2009

**Submitted in partial fulfillment of the Degree of
Bachelor of Technology**

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

**JAYPEE UNIVERSITY OF INFORMATION
TECHNOLOGY-WAKNAGHAT**

CERTIFICATE

This is to certify that the work entitled, "Model for Object Oriented Language" submitted by Mohit Arora, Utkarsh Jain and Pranshu Sachan in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science and Engineering of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.



Brig (retd.) S.P. Ghara

(Head of Computer Science and IT Department)



Mr. Satish Chandra

(Project Coordinator)



Infosys Technologies Limited
No. 350, Habbal Electronics City
Hootagalli, Mysore - 570 018
Tel: 91 821 240 4101 Fax: 91 821 240 4203
www.infosys.com

Certificate of Project Completion at Infosys

This is to certify that Mr./Ms. Mohit Arora has undertaken the project titled "Model for Object Oriented Language" at our organization Infosys Technologies Limited, Mysore, under the guidance of Mr. Ameer Syed for the period 19 January 2009 to 14 May 2009.

Signature of Project Manager

AMEER SYED

Registered Office: Plot No. 44, Electronics City, Hosur Road, Bangalore - 560 100, India





Infosys Technologies Limited
No. 350, Hobbal Electronics City
Hootagalli, Mysore - 570 018
Tel: 91 821 240 4101 Fax: 91 821 240 4200
www.infosys.com

Certificate of Project Completion at Infosys

This is to certify that Mr./Ms. Pranshu Sachan has undertaken the project titled "Model for Object Oriented Language" at our organization Infosys Technologies Limited, Mysore, under the guidance of Mr. Ameer Syed for the period 19 January 2009 to 14 May 2009.

Signature of Project Manager

AMEER SYED

Registered Office : Plot No. 44, Electronics City, Hosur Road, Bangalore - 560 100, India

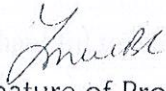




Infosys Technologies Limited
No. 350, Hebbal Electronics City
Hootagalli, Mysore - 570 018
Tel: 91 821 240 4101 Fax: 91 821 240 4200
www.infosys.com

Certificate of Project Completion at Infosys

This is to certify that Mr./Ms. Utkarsh Jain has undertaken the project titled "Model for Object Oriented Language" at our organization Infosys Technologies Limited, Mysore, under the guidance of Mr. Ameer Syed for the period 19 January 2009 to 14 May 2009.


Signature of Project Manager

AMEER SYED

Registered Office: Plot No. 44, Electronics City, Hosur Road, Bangalore - 560 100, India



ACKNOWLEDGMENT

Working on a Live Project is the greatest pleasure for every student and the success of the project enhances the confidence in the student.

Very first I will take opportunity to thank Jaypee University of Information Technology, Wazirpur for giving me this golden opportunity to work with esteemed organization like Infosys Technologies Ltd. It is a dream of every person to do project in Infosys and I am very glad that I got this opportunity.

I thank E&R department, Infosys Technologies Ltd for giving me the opportunity to work freely.

The key to success of a project lies in the hands of the team working on it. Words will be poor gratitude but I take it as an opportunity to offer my heartfelt gratitude to our esteemed Project Lead Mr. Ameer Syed and Project Sponsor Mr. Alsaad Ishaq for his constant encouragement, enlightened comments, and problem solving skills, help and guidance throughout this project.

I also thank my Batch owner at Infosys Technologies Ltd, Mr. Nachiket Shahu Deshmukh for supporting at every step and guiding at crucial points.

I am very much thankful to Mr. Satish Chandra, my project guide in the college for his constant support through out the completion of project through all the possible ways of communication.

I thank my entire project mates and all the colleagues of Infosys Technologies Ltd. who contributed their knowledge and experiences whenever required in this period.

TABLE OF CONTENTS

Chapter I: Introduction

Chapter II: Functional Requirements

Chapter III: Non Functional Requirements.

Chapter IV: Hardware And Software Requirements.

Chapter V: Data Flow Diagram

Chapter VI: Approach Used

Chapter VII: Use Case Diagram

Chapter VIII: Activity Diagram

Chapter IX: Sequence Diagram

Chapter X: Class Diagram

Chapter XI: Class Design Specification

Chapter XII: User Interface

Chapter XIII: Future Scope

Chapter XIV: Conclusion

Chapter XV: Appendix

LIST OF FIGURES

- Figure 1: Platform Used**
- Figure 2: Visual Studio Screenshot**
- Figure 3: Data Flow Diagram (Level 0)**
- Figure 4: Data Flow Diagram (Level 1)**
- Figure 5: Use Case Diagram**
- Figure 6: Activity Diagram**
- Figure 7: Sequence Diagram**
- Figure 8: Class Diagram for object oriented language**
- Figure 9: Class Diagram for procedural language**
- Figure 10: Selection Form**
- Figure 11: Notepad Form**
- Figure 12: Notepad Form**
- Figure 13: Main Form**
- Figure 14: C Sharp Screen shot**
- Figure 15: C++ Screen shot**
- Figure 16: Java Screen shot**
- Figure 17: Java Screen shot**
- Figure 18: C Screen shot**
- Figure 19: Input Values Screen shot**

LIST OF ABBREVEATIONS

GUI: Graphical User Interface.

XML: Extensible Markup Language.

.NET: Microsoft .Net framework

DLL: Dynamic Link Library

VS.NET: Visual Studio .NET

OOPL: Object Oriented Programming Language

ABSTRACT

To develop an efficient Model for Object Oriented Languages. This application will help the programmer to write a commented and indented code in C/C++/JAVA and C# and hence provide a model for writing his programming logics.

Chapter 1

INTRODUCTION

1.1 Problem Statement

1.1.1 Technology Used

1.1.1.1 Platform used

1.1.1.2 Models of Programming Language

1.1.1.3 Purpose of the Project

Chapter I

Introduction

The code generator for an intermediate language serves as the intermediate language to a target language. The intermediate language is a language which contains all the information of a source program like class (union, struct, module), variable, alias, constant, data type, function, procedure, etc. and arguments, return data type.

Chapter I

INTRODUCTION

I.I Problem Statement

I.II Technology Used

I.III Platform Used

I.IV Models of Programming Languages

I.V Purpose of the Project

Chapter I

Introduction

I.I Problem Statement

The code generator for an intermediate language converts the Intermediate language into C/C++/C#/Java code. An intermediate language design is a template like a .XML file containing all the information of a sample program like class (name, size, modifier), variables (size, modifier, data types) function (name, size, access specifier) and arguments (name, data types).

User will be provided with this template to give the information of the program as mentioned above. This information's in the .XML file will undergo the process of Lexical Phase in order to extract the values of the respective information provided by the user in the .XML file. Based on the values the code will be generated.

Depending on the user requirements the respective code (C, C++, C#, Java) will be generated. Code generated will have proper indentations, comments and the code except the logic. Finally the user has to code only the functionality or logic irrespective of the inclusion of header files, main, variable declarations, initializations, comments and indentations. Further the implementations of Inheritance, polymorphism and object oriented concepts has been done.

Apart from that, this OOPL code generator has to be developed based on the aspect oriented, process oriented and component model oriented.

Thus the code generator makes the coder to ease of the effort taken to program.

I.II Technology Used

- Dot net Framework

I.III Platform Used

The platform used is **Microsoft .NET platform**. The .NET platform is a computing platform layer that is positioned above the Windows operating system. It is used to develop distributed applications for the Internet. The .NET platform is composed of a core group of elements, supplemented by other components like development tools and protocols, web clients and consumer applications, and web services and enterprise servers. Microsoft Windows functions as the operating system layer underneath the .NET framework. The .NET framework is included with Windows XP SP2, Windows Server 2003, and Windows Vista. It can also be installed on most of the older versions of Windows. The .NET framework is the core of the .NET platform. It is an environment for building, deploying, and running web services and other applications.

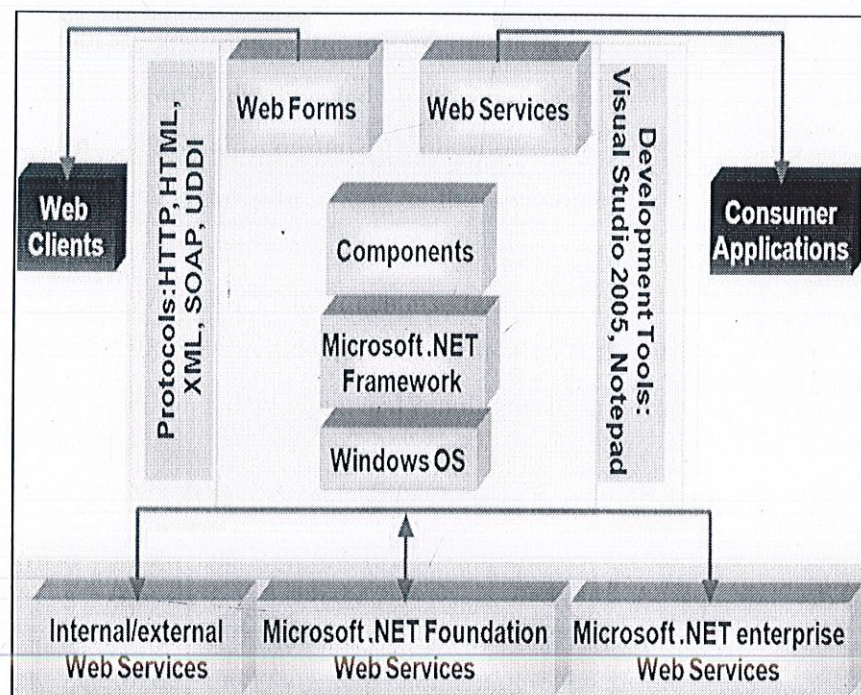


Figure 1: Platform Used

A .NET component is a pre-compiled class module with a DLL extension. At run-time, it is invoked and loaded into memory to be used by a consumer application. .NET components are used to create both web and Windows applications. They help expose the required functionality of an application to the outside world. The .NET platform also comprises web forms. Web forms are standard interfaces that can be downloaded from the Internet. A web form contains text boxes for users to enter data. Users can then submit the form to the receiver.

Visual Studio 2008 Tool



The tool used to develop applications is *Visual Studio.NET*. Microsoft Visual Studio 2008 Team System is the Integrated Development Environment (IDE). This helps in fast development of Microsoft applications on .NET platform. We can develop Console application as well as Windows Application in C# using Visual Studio. Visual Studio 2008 Team System provides automated code analysis and help in creating test cases for unit testing. VS.NET is a good tool as this makes most of the coding easy for any developer thereby increasing the productivity. This is Microsoft's main software development product for developers.

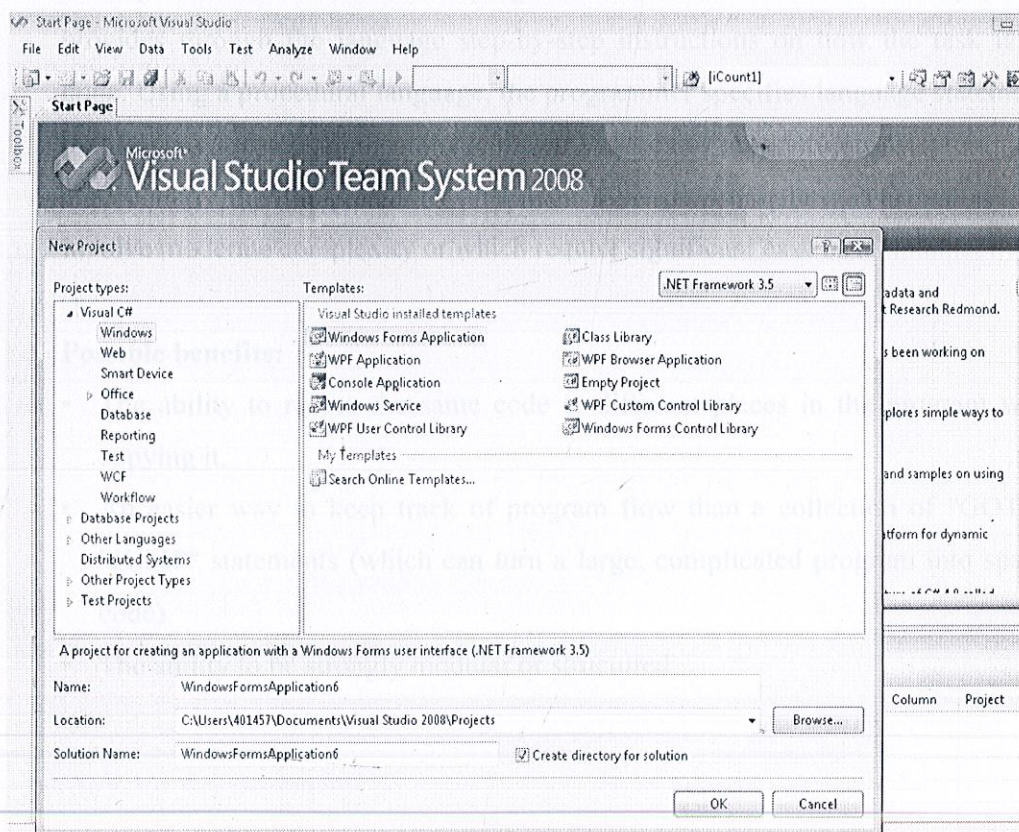


Figure 2: Visual Studio Screenshot

I.IV Models of Programming Languages

I.IV.I Procedural Languages

Procedural programming can sometimes be used as a synonym for imperative programming (specifying the steps the program must take to reach the desired state), but can also refer (as in this article) to a programming paradigm based upon the concept of the procedure call. Procedures, also known as routines, subroutines, methods, or functions (not to be confused with mathematical functions, but similar to those used in functional programming) simply contain a series of computational steps to be carried out. Any given procedure might be called at any point during a program's execution, including by other procedures or itself. A procedural programming language provides a programmer a means to define precisely each step in the performance of a task. The programmer knows what is to be accomplished and provides through the language step-by-step instructions on how the task is to be done. Using a procedural language, the programmer specifies language statements to perform a sequence of algorithmic steps. Procedural programming is often a better choice than simple sequential or unstructured programming in many situations which involve moderate complexity or which require significant ease of maintainability.

Possible benefits:

- The ability to re-use the same code at different places in the program without copying it.
- An easier way to keep track of program flow than a collection of "GOTO" or "JUMP" statements (which can turn a large, complicated program into spaghetti code).
- The ability to be strongly modular or structured.

LIV.II Object Oriented Languages

An object-oriented programming language (also called an OO language) is one that allows or encourages, to some degree, object-oriented programming techniques such as encapsulation, inheritance, modularity, and polymorphism. Simula (1967) is generally accepted as the first language to have the primary features of an object-oriented language. It was created for making simulation programs, in which what came to be called objects were the most important information representation. Smalltalk (1972 to 1980) is arguably the canonical example, and the one with which much of the theory of object-oriented programming was developed.

- Languages called "pure" OO languages, because everything in them is treated consistently as an object, from primitives such as characters and punctuation, all the way up to whole classes, prototypes, blocks, modules, etc. They were designed specifically to facilitate, even enforce, OO methods. Examples: Smalltalk, Eiffel, Ruby
- Languages designed mainly for OO programming, but with some procedural elements. Examples: C++, Java, Python.
- Languages that are historically procedural languages, but have been extended with some OO features. Examples: FORTRAN 2003, Perl, COBOL 2002.
- Languages with most of the features of objects (classes, methods, inheritance, reusability), but in a distinctly original form. Examples: Oberon (Oberon-1 or Oberon-2).
- Languages with abstract data type support, but not all features of object-orientation, sometimes called object-based languages. Examples: Modula-2 (with excellent encapsulation and information hiding), Pliant, CLU.

I.V Purpose of the Project

I.V.I Current System / Business Process

Object Oriented Programming languages like C++, C#, Java as well as procedural languages like C.

I.V.I.II Limitations of the Current System / Business Process

- Lot of effort is required for making the code maintainable. i.e. Comments and Indentation.
- The code written in one language cannot be converted into another language if required easily
- There is lot of static codes that are there for the basic system to work. Like getter / setter functions, constructors and destructors.
- No Support for Component Based Modeling.

I.V.II Proposed System

We are looking to make use of XML as our input file as XML is a structured file format and being used all over the world for the data transfer and is being known by almost all the programmers all over the world. Moreover since XML is a strongly tagged language, implementing Object Oriented features like inheritance and polymorphism would become much easier. Taking XML file as input would also allow us to handle the errors in a much more efficient manner.

We are also looking forward to extend the design to procedural languages like C and allowing the user to add readymade components like Login component so that he does not need to write the code for this component and thus make the job of the programmer much easier.

The Design could also be further extended to include support for advance languages like Aspect Oriented Languages.

I.V.II.I Objectives of the Proposed System

This system is used to formulate a technique to describe a Class, member functions, member variables, their types and sizes. In this system we planned to define keywords of the language which can represent a class, member variables etc.

The class that would be formed would also include support for inheritance and polymorphism as well as various components like login component.

This system converts the description into a Class file and related cpp file, cs file in case of C++ and C#. The language defined above would be parsed and lexically analyzed by creating a Parser and Lexical analyzer.

The System also include support for procedural language (C) to describe structures, functions, variables their types and sizes and also support for advanced features like polymorphism.

Chapter II

Chapter II

FUNCTIONAL REQUIREMENTS

II.I Functional Requirements

II.II Standard Requirements

II.III Functionality Overview

II.IV External Interface Requirements

Chapter II

Functional Requirements

II.I Functional Requirements

Designing the intermediate language which will parse the tags and other Functionalities. This will automatically generate the class, member functions, member variables, constructors, and destructors in case of C++ and similarly for Java. In case of C #, it will automatically generate the class, member functions, member variables, constructors, destructors and set and get method. In case of C it will generate the corresponding structures, functions, variables as well as their types and sizes.

The xml files will be given as an input to our system. At the initial stage the tags will be parsed from the xml files. This will be done by using the lexical analyzer and parser. The parsed code (parse tree) will be undergoing semantically checking and syntactical checking.

The xml files will be stored in the database as in the simpler format of tables. Apart from this the keywords, member functions, member methods, header files will be stored separately in the database of the corresponding object oriented languages.

The Design will also include support for Component Object Model for the languages C/C++/C#/JAVA.

Finally, we have to implement the concepts of Object Oriented languages in this system.

II.II Standard Requirements

The standard requirements for our system are the intermediate language design.

II.II.I Detailed Design

Lexical

Lexical analysis is the process of converting a sequence of characters into a sequence of tokens. Programs performing lexical analysis are called lexical analyzers or lexers. A lexer consists of a scanner and a tokenizer. A token is a categorized block of text. The block of text corresponding to the token is known as a lexeme. A lexical analyzer processes lexemes to categorize them according to function, giving them meaning. This assignment of meaning is known as tokenization.

Parser

Parsing is the process of analyzing a sequence of tokens to determine its grammatical structure with respect to a given formal grammar. A parser is the component of a compiler that carries out this task.

Parsing transforms input text into a data structure, usually a tree, which is suitable for later processing and which captures the implied hierarchy of the input. Lexical analysis creates tokens from a sequence of input characters and it is these tokens that are processed by a parser to build a data structure such as parse tree or abstract syntax trees.

Semantic

Semantic analysis is the phase in which the compiler adds semantic information to the parse tree and builds the symbol table. This phase performs semantic checks such as type checking (checking for type errors), or object binding (associating variable and function references with their definitions), or definite assignment (requiring all local variables to be initialized before use), rejecting incorrect programs or issuing warnings. Semantic analysis usually requires a complete parse tree, meaning that this phase logically follows the parsing phase, and logically precedes the code generation phase, though it is often possible to fold multiple phases into one pass over the code in a compiler implementation

Code generator

Code generation is the process by which a compiler's code generator converts some internal representation of source code into a form (e.g., machine code) that can be readily executed by a machine (often a computer).

II.II.II User Input

The Input to given tool will be a xml file using the element and attributes format for inputting the user requirements (root element will represent the class and the data members will be represented by the further child elements and the data types and size will represent the attributes of that data member).

II.II.II.I XML

XML (Extensible Markup Language) is a general-purpose specification for creating custom markup languages. It is classified as an extensible language, because it allows the user to define the mark-up elements. XML's purpose is to aid information systems in sharing structured data, especially via the Internet, to encode documents, and to serialize data; in the last context, it compares with text-based serialization languages XML, in combination with other standards, makes it possible to define the content of a document separately from its formatting, making it easy to reuse that content in other applications or for other presentation environments. Most importantly, XML provides a basic syntax that can be used to share information between different kinds of computers, different applications, and different organizations without needing to pass through many layers of conversion.

An XML document has two correctness levels:

- **Well-formed:** A well-formed document conforms to the XML syntax rules.
- **Valid:** A valid document additionally conforms to semantic rules, either user-defined or in an XML schema.

II.II.II.II Advantages of using XML:

XML is a language that is able to represent not only data itself, but also the nature and structure of that data. This combination of data and metadata in the one file means XML has considerable advantages over traditional data formats. As XML files are plain text and comply with a globally accepted standard, the files are very useful in almost any application where interoperability is required.

XML is a cross platform format that is not exclusive to any particular operating system or development platform.

The System.Xml namespace in this framework provides considerable functionality for reading, creating and manipulating data in XML format.

II.II.III Component Based Model

Component-Based Development claims to offer a radically new approach to the design, construction, implementation and evolution of software applications. Software applications are assembled from components from a variety of sources; the components themselves may be written in several different programming languages and run on several different platforms.

II.II.IV Coding

Infosys coding standards – languages, libraries, configuration management procedures etc.

II.III Functionality Overview

This tool allows the user to generate a commented code so that almost half of his work will be done in advance. The user will input a xml file by browsing it in the file system, which contains the users requirements and then selecting an output file where the generated code needs to be saved according to the file extension given by the user the corresponding button will be enabled and as soon as the users selects that button the output code would be generated in the user given language choice. The user also needs to select a output variables where the values of the output file will be saved just to verify that the values which the system has extracted is in accordance with that of the users need. In case there are any errors in the XML input file then the error button will be enabled and all rest buttons would be disabled, and on click of that error button an error file is displayed. One can proceed further only after rectifying all the errors.

II.IV External System Interface Requirements

The system that is being developed might interface with many other existing Object Oriented languages like C++, C#, Java and Procedural Language like C.

Chapter III

Non Functional Requirements

III.I Usability

Our system must allow the user to work easily. Design will be given to the user so that the user can use the system well and easily to use the system. Our system must be easy to understand so that user can readily work.

III.II Reliability

Our system depends on the inputs given by the user since the keywords given by the user will be parsed and automatically converted into respective code.

Chapter III

III.III Performance

In high end configuration, the system must be able to handle a large number of concurrent users and access the system effectively. System is capable of performing large volumes of data.

NON FUNCTIONAL REQUIREMENTS

III.IV Maintainability

Our design will be the abstraction class for object oriented languages like C++ or C#, Java and for procedural language like C.

III.V Atnki language Support

Our System supports multiple language. I.e. it can convert the Xmi files into corresponding code in C/C++ or Java.

Chapter III

Non Functional Requirements

III.I Usability

Our system makes the user to work easily. Demo will be given to the user, which makes the user to know the system well and ready to use the system. Our intermediate language is easy to understand so that user can readily work.

III.II Reliability

Our system depends on the inputs given by the user since the keywords given by the user will be parsed and automatically generates the respective code.

III.III Performance

In high end configuration the response time of our system will be good. Concurrent users can access the system effectively. System is capable of performing huge volumes of data.

III.IV Maintainability

Our design will be the abstraction class for object oriented languages like C++, C#, Java and for procedural language like C.

III.V Multi language Support

Our System supports multiple language i.e. it can convert the Xml files into corresponding code in C/C++/C# or Java.

Chapter IV

Hardware and Software Requirements

IV.1 Deployment environment requirements

Hardware Requirements

| | |
|-----------|----------------------|
| Monitor | 17" TFT Display |
| Keyboard | 106 keys |
| Mouse | Optical Scroll Mouse |
| Hard disk | 40 GB |
| RAM | 512 MB |

Chapter IV

Software Requirements

| | |
|-------------------|-------------------|
| OS for Web Server | Microsoft Windows |
|-------------------|-------------------|

HARDWARE AND SOFTWARE REQUIREMENTS

| | |
|-----------------|--------------------|
| Database Server | MS SQL Server 2005 |
|-----------------|--------------------|

IV.2 Development Environment Requirements

Hardware Requirements

| | |
|-----------|------------------------------|
| IDE | Microsoft Visual Studio 2008 |
| Monitor | Acer 17" TFT Display |
| Keyboard | Acer 106 keys |
| Mouse | Acer Optical Scroll Mouse |
| Hard disk | 40 GB |
| RAM | 512 MB |

Chapter IV

Hardware and Software Requirements

IV.I Deployment environment requirements

Hardware Requirements

| | |
|-----------|------------------------|
| Monitor | : 17" TFT Display |
| Keyboard | : 106 keys |
| Mouse | : Optical Scroll Mouse |
| Hard disk | : 40 GB |
| RAM | : 512 MB |

Software Requirements

| | |
|-------------------|--|
| OS for Web Server | : Microsoft Windows. : .Net Framework 3.5 |
| Database Server | : MS SQL Server 2005 |

IV.II Development Environment Requirements

Hardware Requirements

| | |
|-----------|--------------------------------|
| IDE | : Microsoft Visual Studio 2008 |
| Monitor | : Acer 17" TFT Display |
| Keyboard | : Acer 106 keys |
| Mouse | : Acer Optical Scroll Mouse |
| Hard disk | : 160 GB |
| RAM | : 2 GB |

Chapter V

Data Flow Diagrams

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design).

Level 0

Chapter V

DATA FLOW DIAGRAMS

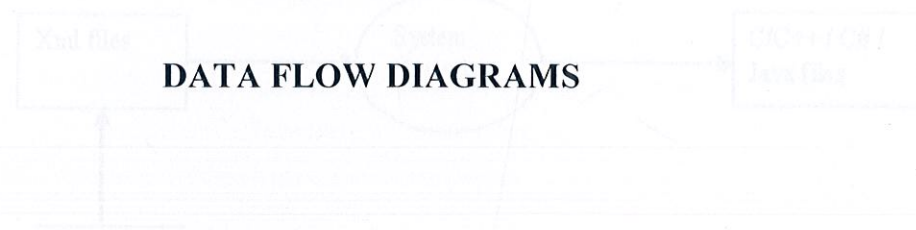


Figure 3: Data Flow Diagram (Level 0)

Chapter V

Data Flow Diagrams

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design).

Level 0

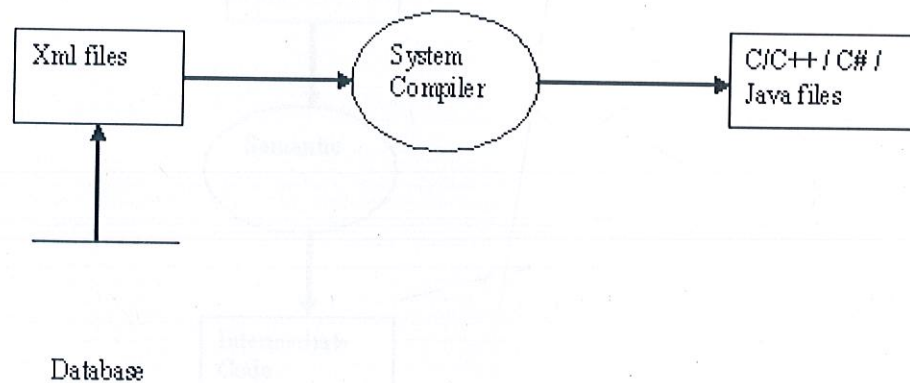
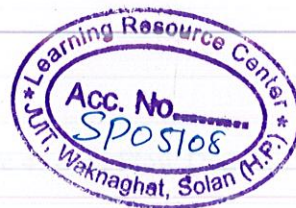


Figure 3: Data Flow Diagram (Level 0)



Level 1

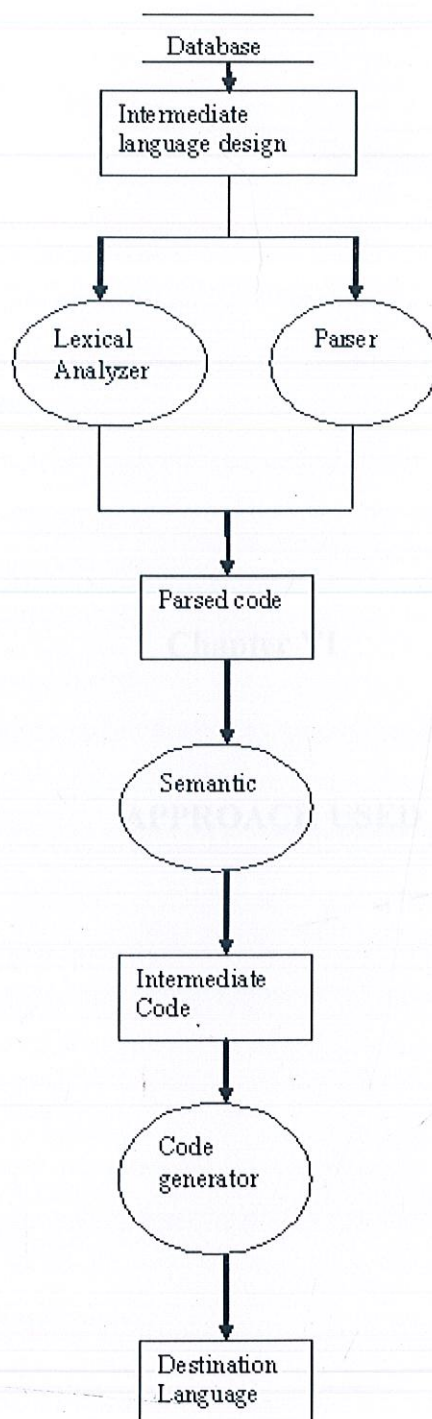


Figure 4: Data Flow Diagram (Level 1)

Chapter IV

Approach Used

In this project we have used the Object Oriented approach. The Project is divided into two modules: Input Oriented part and Procedural Part.

Input XML File

The Input xml file includes the Element attribute format in which a user can input his requirements in the template provided. We can add any number of classes, functions, variables, structures through this file.

Chapter VI

Object Oriented Module

In this module we extracted the values given by the user in the xml input file and then we used these values to generate the corresponding code in C++, Java, C#. The code generated was properly commented and indexed. In case of Java separate files for each public class are generated. The high level programming features such as inheritance and polymorphism were also taken care of and implemented in a very appropriate manner. Constructors and user defined data types such as structures are also implemented.

APPROACH USED

Procedural Module

In this module also we extracted the values given by the user in the xml input file and then we used these values to generate the corresponding code in C. The code generated was properly commented and indexed. User defined data types such as structures and unions are also implemented.

Error File

All the errors encountered in the xml file such as missing attributes, empty tags, incomplete tags or some other inconsistencies are catch back and a separate error file.

Chapter IV

Approach Used

In this project we have used the Object Oriented approach. The Project is divided into two modules Object Oriented part and Procedural Part.

Input XML File

The Input xml file includes the Element attribute format in which a user can input his requirements in the template provided. We can add any number of classes, functions, variables, structures through this file.

Object Oriented Module

In this module we extracted the values given by the user in the xml input file and then we used these values to generate the corresponding code in C#, Java, C++. The code generated was properly commented and indented. In case of java separate files for each public class are generated. The high level programming features such as inheritance and polymorphism were also taken care of and implemented in a very appropriate manner. Constructors and user defined data types such as structures are also implemented.

Procedural Module

In this module also we extracted the values given by the user in the xml input file and then we used these values to generate the corresponding code in C. The code generated was properly commented and indented. User defined data types such as structures and unions are also implemented.

Error File

All the errors encountered in the xml file such as missing attributes, empty tags, incomplete tags or some other inconsistencies are reflected back into a separate error file.

This file provides proper line number and type of error in the xml file. On selecting the error in the error file the corresponding line is highlighted in the input file. The user is not allowed to proceed further without correcting the inconsistencies in the input file.

Chapter VII

USE CASE DIAGRAMS AND DESCRIPTION

Chapter VII

Use Case Diagram - And Description

Chapter VII

USE CASE DIAGRAMS AND DESCRIPTION

Chapter VII

Use Case Diagrams And Description

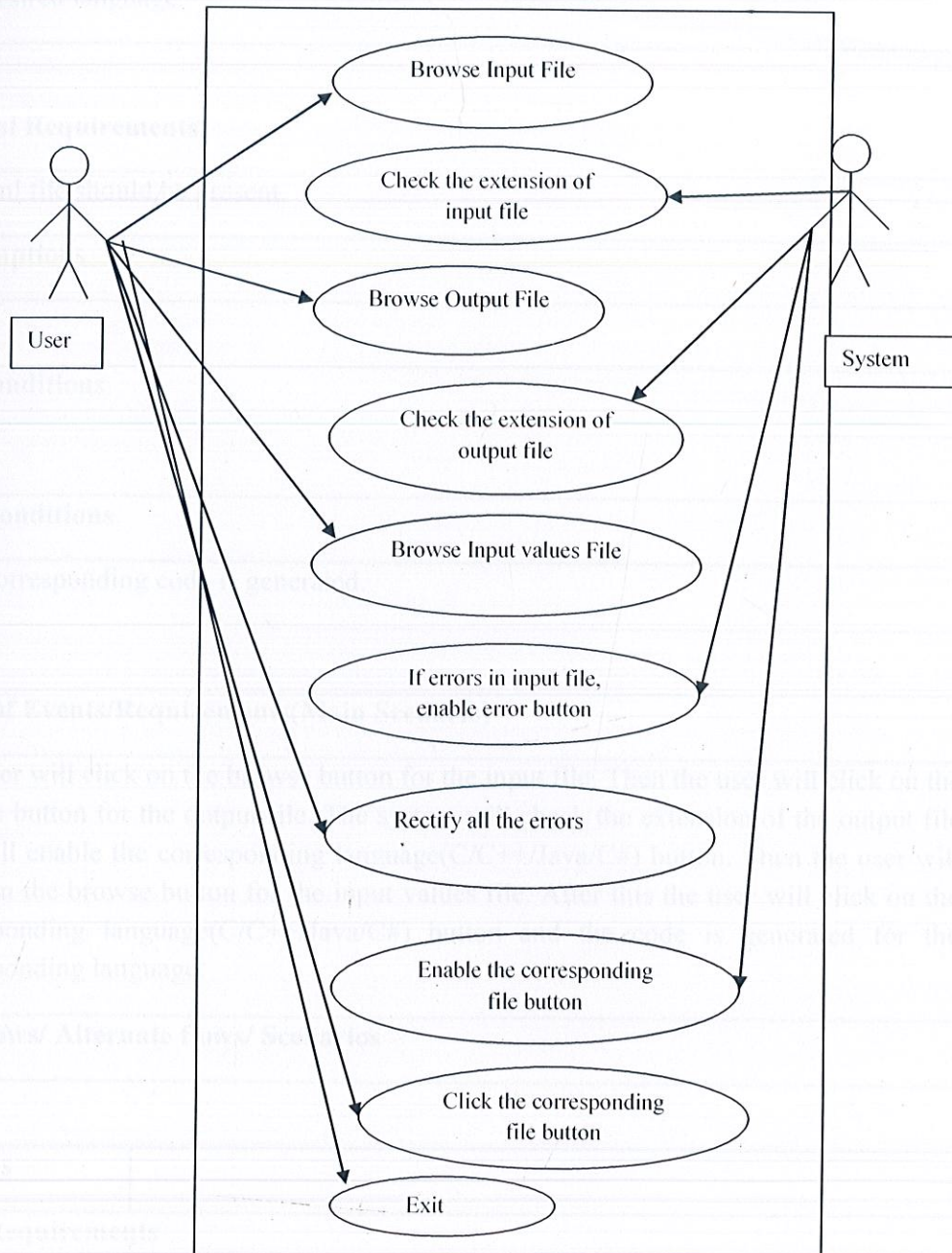


Figure 5: Use Case Diagram

| |
|--|
| Description |
| This use case describes the steps the user has to go through for the conversion of xml file into desired language. |

| |
|--------------------------------------|
| Special Requirements |
| An Xml file should be present. |
| Assumptions |
| None |
| Pre-conditions |
| None |
| Post-conditions |
| The Corresponding code is generated. |

| |
|--|
| Flow of Events/Requirements(Main Scenario) |
| The user will click on the browse button for the input file. Then the user will click on the browse button for the output file. The system will check the extension of the output file and will enable the corresponding language(C/C++/Java/C#) button. Then the user will click on the browse button for the input values file. After this the user will click on the corresponding language(C/C++/Java/C#) button and the code is generated for the corresponding language. |
| Sub flows/ Alternate flows/ Scenarios |
| None |
| Screens |
| |
| Data Requirements |
| None |
| Issues |

Chapter VIII

Activity Diagram

Chapter VIII

ACTIVITY DIAGRAM

Chapter VIII

Activity Diagram

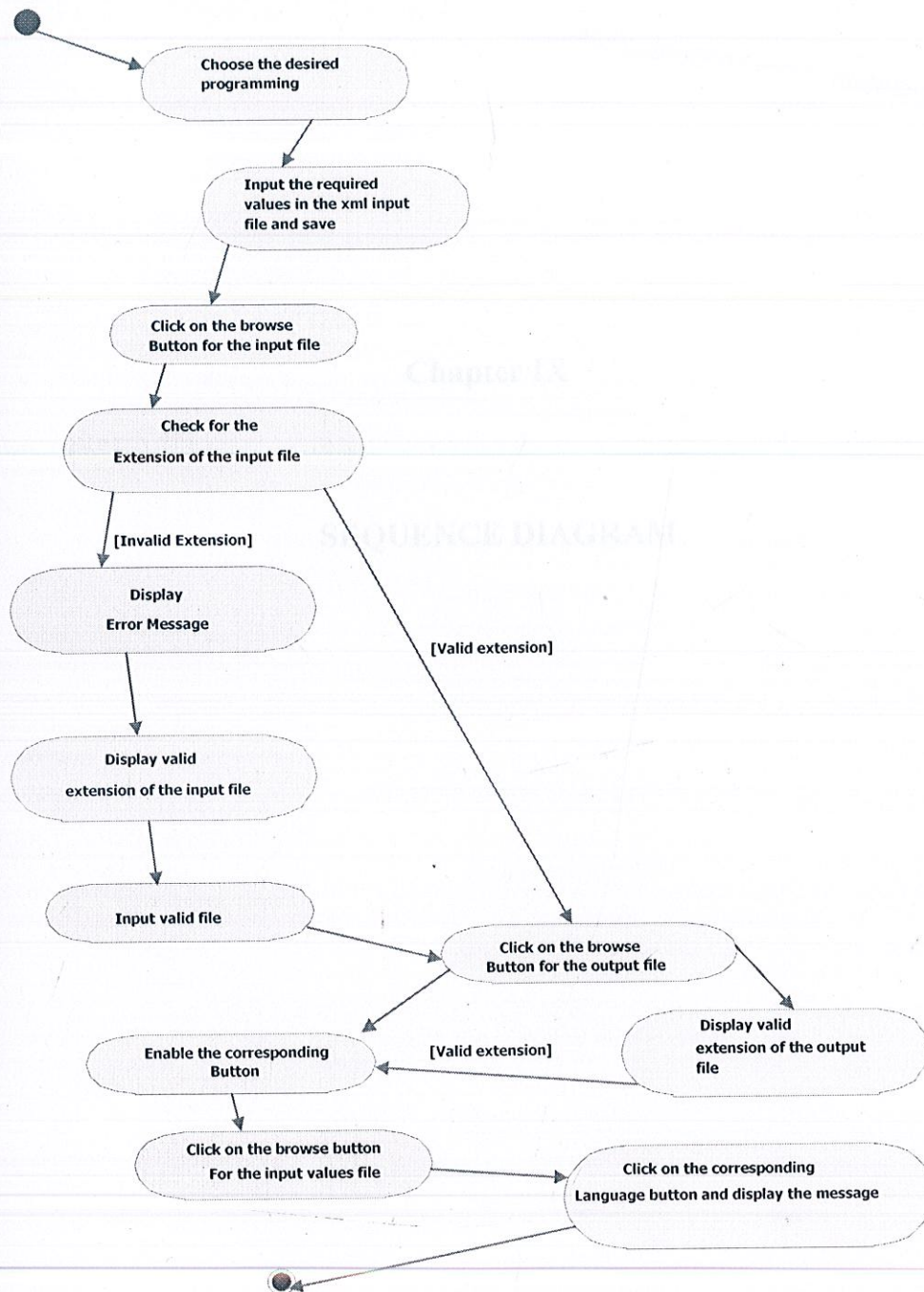


Figure 6: Activity Diagram

Chapter IX

SEQUENCE DIAGRAM

Figure 7. Sequence Diagram

Chapter IX

Sequence Diagram

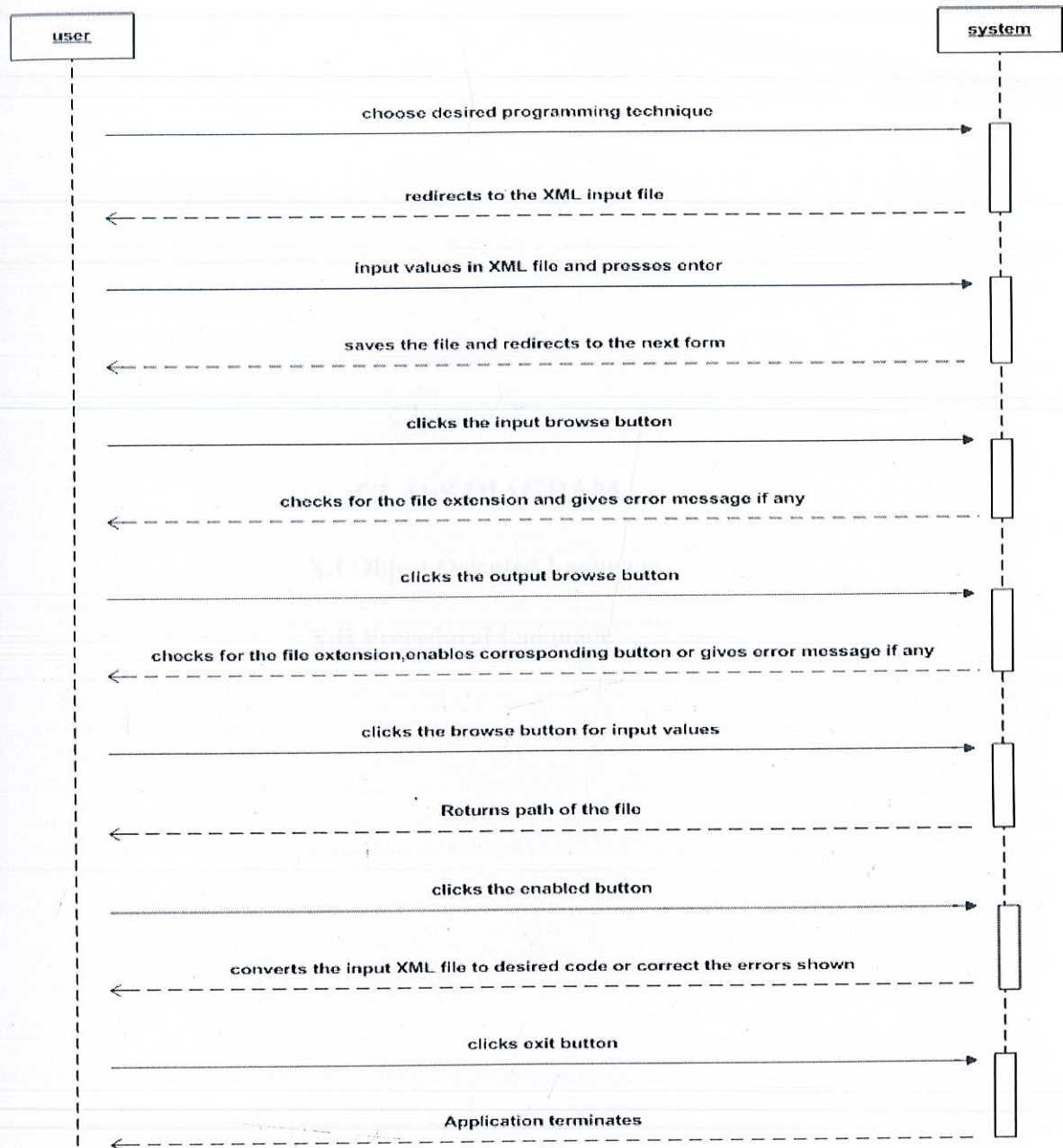


Figure 7: Sequence Diagram

Chapter X

Class Diagram

X.I Object Oriented

Language

Chapter X

CLASS DIAGRAM

X.I Object Oriented Language

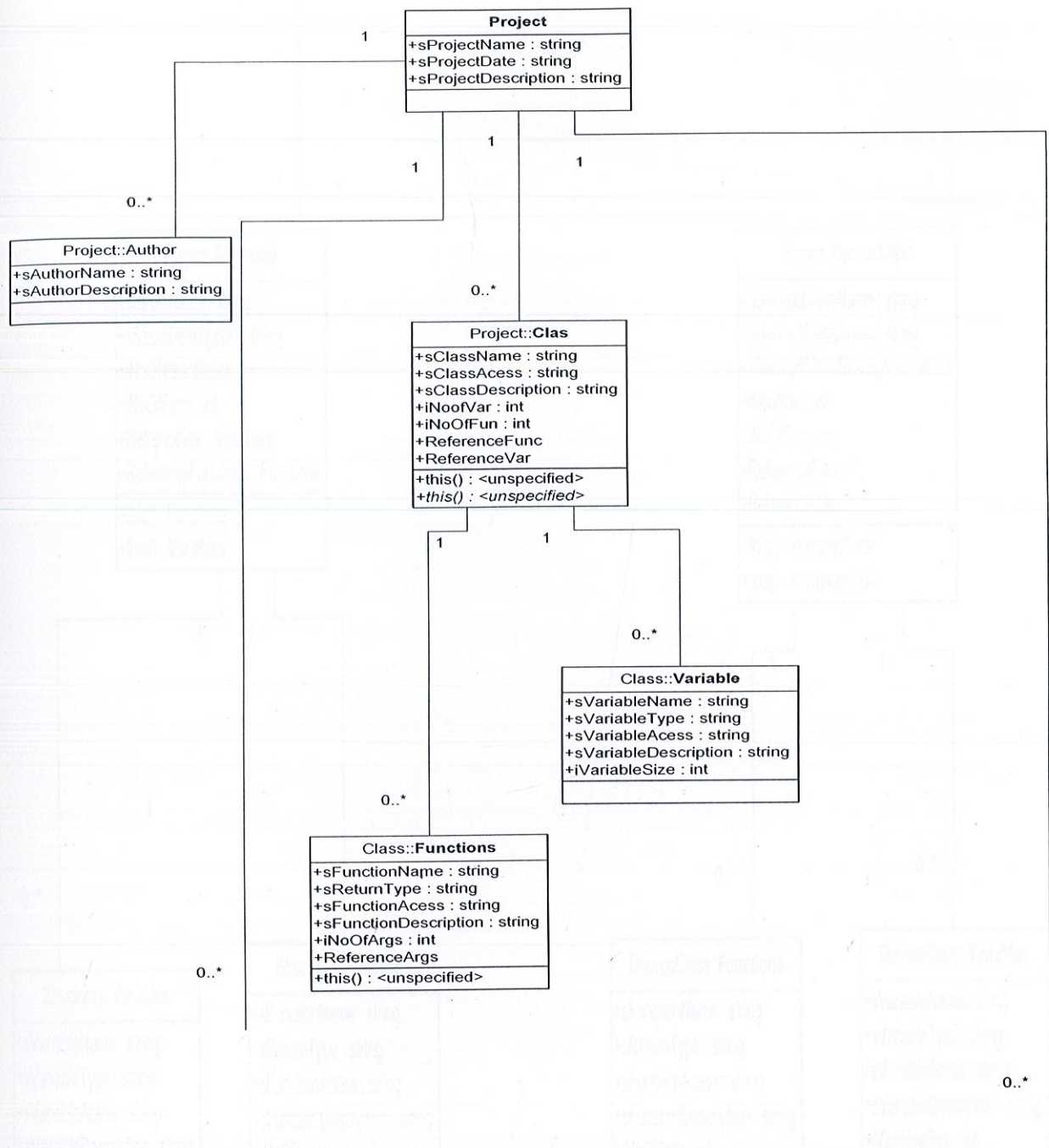
X.II Procedural Language

Chapter X

Class Diagram

X.I Object Oriented

Language:



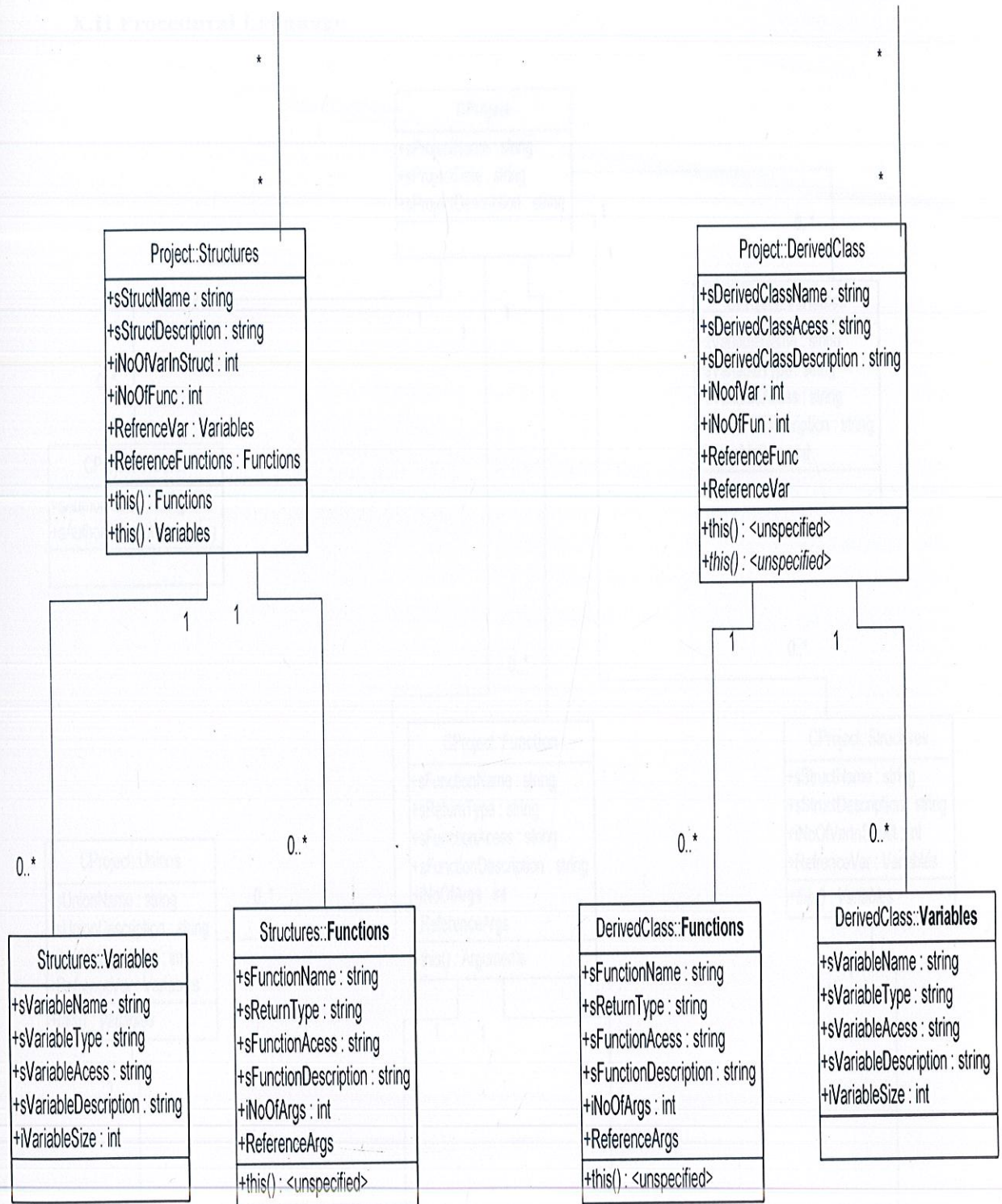
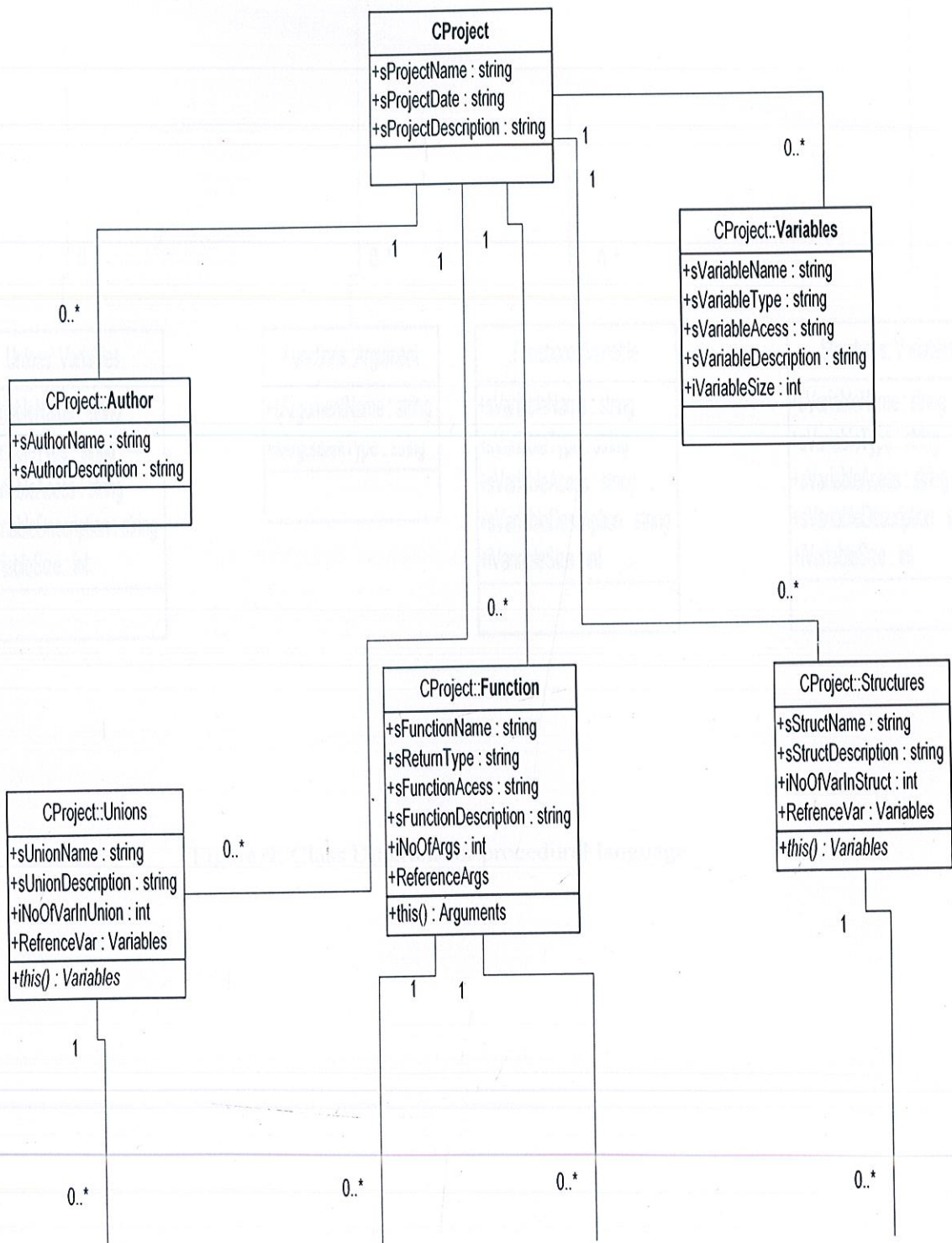


Figure 8: Class Diagram for object oriented language

X.II Procedural Language



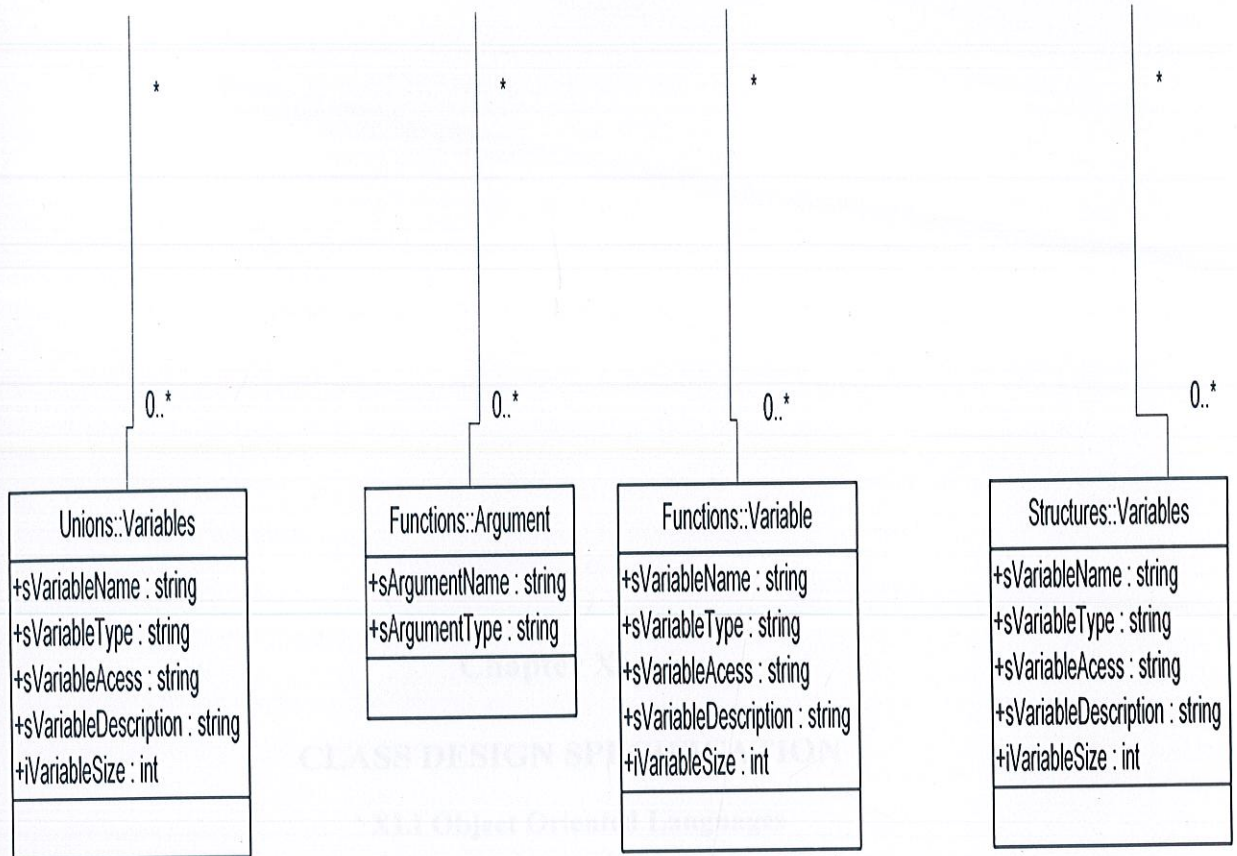


Figure 9: Class Diagram for procedural language

Chapter XI

CLASS DESIGN SPECIFICATION

XI.I Object Oriented Languages

| | |
|--------------------|--|
| Class Name | Project |
| Class Description | This Class contains Data Members used to identify the project uniquely |
| Class Inheritance | None |
| Classes Referenced | Chapter XI CLASS DESIGN SPECIFICATION XI.I Object Oriented Languages XI.II Procedural Languages XI.III Main Program |
| Sub System | None |
| Class Type | Complete |
| Change History | None |

Chapter XI

CLASS DESIGN SPECIFICATION

XI.I Object Oriented Languages

Class Identification

| | |
|---------------------------|--|
| Class Name | Project |
| Class Description | This Class contains Data Members used to identify the project uniquely |
| Class Inheritance | None |
| Classes Referenced | Main Program |
| Sub System | None |
| Class Type | Complete |
| Change History | None |

Attributes (or Data members)

| Attribute Name | Description | Type | Scope | Range |
|---------------------|---|------------------------|--------|-------|
| sProjectName | Stores the Project Name | String | Public | |
| sProjectDescription | Stores the project description | String | Public | |
| sProjectDate | Stores the Project Date | String | Public | |
| iNoOfAuthor | Stores the no of arguments | Integer | Public | |
| iNoOfClasses | Stores the no of classes in the project | Integer | Public | |
| aAuthor | Creates an array of Author Class | Array of Authors Class | Public | |

Methods (or class functions)

Method 1

| | |
|-----------------------------------|-------------------------------|
| Method Signature | public Author this[int index] |
| Purpose and algorithm description | Indexer for Author object |
| Screen validations, if any | None |

| | |
|--------------------------|--------|
| Tables/Files Used | None |
| Scope of method | Public |

Method 2

| | |
|--|--------------------------------|
| Method Signature | public Classes this[int index] |
| Purpose and algorithm description | Indexer for Classes object |
| Screen validations, if any | None |
| Tables/Files Used | None |
| Scope of method | Public |

Class Identification

| | |
|---------------------------|---|
| Class Name | Author |
| Class Description | This Class contains Data Members used to identify the author uniquely |
| Class Inheritance | None |
| Classes Referenced | Project |
| Sub System | None |
| Class Type | Complete |

Attributes (or Data members)

| Attribute Name | Description | Type | Scope | Range |
|--------------------|-------------------------------|--------|--------|-------|
| sAuthorName | Stores the Author Name | String | Public | |
| sAuthorDescription | Stores the Author description | String | Public | |

Class Identification

| | |
|---------------------------|--|
| Class Name | Classes |
| Class Description | This Class contains Data Members used to declare the Class |
| Class Inheritance | None |
| Classes Referenced | Project |
| Sub System | None |
| Class Type | Complete |
| Change History | None |

Attributes (or Data members)

| Attribute Name | Description | Type | Scope | Range |
|----------------|---------------------------|--------|--------|-------|
| sClassName | Stores the Class Name | String | Public | |
| sClassAccess | Stores the Classes Access | String | Public | |

| | | | | |
|-------------------|---|-------------------------|--------|--|
| | Specifier | | | |
| sClassDescription | Stores the Class Description | String | Public | |
| iNoOfVar | Stores the no of variables | Integer | Public | |
| iNoOfFunctions | Stores the no of functions in the class | Integer | Public | |
| vVariable | Creates an array of Variable Class | Array of Variable Class | Public | |
| fFunction | Creates an array of Functions Class | Array of Function Class | Public | |

Methods (or class functions)

Method 1

| | |
|--|---------------------------------|
| Method Signature | public Variable this[int index] |
| Purpose and algorithm description | Indexer for Variable object |
| Screen validations, if any | None |
| Tables/Files Used | None |
| Scope of method | Public |

Method 2

| | |
|--|----------------------------------|
| Method Signature | public Function this[long index] |
| Purpose and algorithm description | Indexer for Functions object |
| Screen validations, if any | None |
| Tables/Files Used | None |
| Scope of method | Public |

Class Identification

| | |
|---------------------------|---|
| Class Name | Function |
| Class Description | This Class contains Data Members used by a Function |
| Class Inheritance | None |
| Classes Referenced | Classes |
| | |

| | |
|-------------------|----------|
| Sub System | None |
| Class Type | Complete |

Attributes (or Data members)

| Attribute Name | Description | Type | Scope | Range |
|----------------------|---|-------------------------|--------|-------|
| sFunctionName | Stores the function name | String | Public | |
| sReturnType | Stores the function return type | String | Public | |
| sFunctionAccess | Stores the function Access Specifier | String | Public | |
| sFunctionDescription | Stores the function Description | String | Public | |
| iNoOfArg | Stores the no of arguments in the functions | Integer | Public | |
| aArgument | Creates an array of Argument Class | Array of Argument Class | Public | |

Methods (or class functions)

Method 1

| | |
|--|---------------------------------|
| Method Signature | public Argument this[int index] |
| Purpose and algorithm description | Indexer for Argument object |
| Screen validations, if any | None |
| Tables/Files Used | None |
| Scope of method | Public |

Class Identification

| | |
|---------------------------|---|
| Class Name | Argument |
| Class Description | This Class contains Data Members used to identify the Argument uniquely |
| Class Inheritance | None |
| Classes Referenced | Function |
| | |

| | |
|-----------------------|----------|
| Sub System | None |
| Class Type | Complete |
| Change History | None |

Attributes (or Data members)

| Attribute Name | Description | Type | Scope | Range |
|----------------|--------------------------|--------|--------|-------|
| sArgumentName | Stores the Argument Name | String | Public | |
| sArgumentType | Stores the Argument Type | String | Public | |

Class Identification

| | |
|---------------------------|---|
| Class Name | Variable |
| Class Description | This Class contains Data Members used to declare a Variable |
| Class Inheritance | None |
| Classes Referenced | Function |

| | |
|-----------------------|----------|
| Sub System | None |
| Class Type | Complete |
| Change History | None |

Attributes (or Data members)

| Attribute Name | Description | Type | Scope | Range |
|----------------------|---------------------------------|---------|--------|-------|
| sVariableName | Stores the Variable Name | String | Public | |
| sVariableType | Stores the Variable Type | String | Public | |
| sVariableAccess | Stores the Variable Access | String | Public | |
| sVariableDescription | Stores the variable Description | String | Public | |
| iVariableSize | Stores the Variable Size | Integer | Public | |

XI.II Procedural Languages

Class Identification

| | |
|---------------------------|--|
| Class Name | Project |
| Class Description | This Class contains Data Members used to identify the project uniquely |
| Class Inheritance | None |
| Classes Referenced | Main Program |
| Sub System | None |
| Class Type | Complete |

Attributes (or Data members)

| Attribute Name | Description | Type | Scope | Range |
|---------------------|--------------------------------|--------|--------|-------|
| sProjectName | Stores the Project Name | String | Public | |
| sProjectDescription | Stores the project description | String | Public | |

| | | | | |
|--------------|--|--------------------------|--------|--|
| | | | | |
| sProjectDate | Stores the Project Date | String | Public | |
| iNoOfAuthor | Stores the no of arguments | Integer | Public | |
| iNoOfFunc | Stores the no of Functions in the project | Integer | Public | |
| aAuthor | Creates an array of Author Class | Array of Authors Class | Public | |
| iNoOfVar | Stores the no of global variables in the project | Integer | Public | |
| aVariables | Creates an array of Variable Class | Array of Variables Class | Public | |
| AFunctions | Creates and array of functions | Array of function Class | Public | |

Methods (or class functions)

Method 1

| | |
|--|-------------------------------|
| Method Signature | public Author this[int index] |
| Purpose and algorithm description | Indexer for Author object |
| Screen validations, if any | None |

| | |
|--------------------------|--------|
| Tables/Files Used | None |
| Scope of method | Public |

Method 2

| | |
|--|----------------------------------|
| Method Signature | public Functions this[int index] |
| Purpose and algorithm description | Indexer for Functions object |
| Screen validations | None |
| Tables/Files Used | None |
| Scope of method | Public |

Method 3

| | |
|-------------------------|----------------------------------|
| Method Signature | public Variables this[int index] |
|-------------------------|----------------------------------|

| | |
|--|------------------------------|
| Purpose and algorithm description | Indexer for Variables object |
| Screen validations, if any | None |
| Tables/Files Used | None |
| Scope of method | Public |

Class Identification

| | |
|---------------------------|---|
| Class Name | Author |
| Class Description | This Class contains Data Members used to identify the author uniquely |
| Class Inheritance | None |
| Classes Referenced | Project |
| Sub System | None |
| Class Type | Complete |

| | |
|-----------------------|------|
| Change History | None |
|-----------------------|------|

Attributes (or Data members)

| Attribute Name | Description | Type | Scope | Range |
|--------------------|-------------------------------|--------|--------|-------|
| sAuthorName | Stores the Author Name | String | Public | |
| sAuthorDescription | Stores the Author description | String | Public | |

Class Identification

| | |
|---------------------------|---|
| Class Name | Function |
| Class Description | This Class contains Data Members used by a Function |
| Class Inheritance | None |
| Classes Referenced | Project |
| Class Type | Complete |

Attributes (or Data members)

| Attribute Name | Description | Type | Scope | Range |
|----------------|-------------|------|-------|-------|
|----------------|-------------|------|-------|-------|

| | | | | |
|----------------------|---|-------------------------|--------|--|
| sFunctionName | Stores the function name | String | Public | |
| sReturnType | Stores the function return type | String | Public | |
| sFunctionAccess | Stores the function Access Specifier | String | Public | |
| sFunctionDescription | Stores the function Description | String | Public | |
| iNoOfArg | Stores the no of arguments in the functions | Integer | Public | |
| aArgument | Creates an array of Argument Class | Array of Argument Class | Public | |

Methods (or class functions)

Method 1

| | |
|--|---------------------------------|
| Method Signature | public Argument this[int index] |
| Purpose and algorithm description | Indexer for Argument object |
| Screen validations, if any | None |

| | |
|--------------------------|--------|
| Tables/Files Used | None |
| Scope of method | Public |

Class Identification

| | |
|---------------------------|---|
| Class Name | Variable |
| Class Description | This Class contains Data Members used to declare a Variable |
| Class Inheritance | None |
| Classes Referenced | Project |
| Sub System | None |
| Class Type | Complete |
| Change History | None |
| | |

Attributes (or Data members)

| Attribute Name | Description | Type | Scope | Range |
|----------------------|---------------------------------|---------|--------|-------|
| sVariableName | Stores the Variable Name | String | Public | |
| sVariableType | Stores the Variable Type | String | Public | |
| sVariableAccess | Stores the Variable Access | String | Public | |
| sVariableDescription | Stores the variable Description | String | Public | |
| iVariableSize | Stores the Variable Size | Integer | Public | |

Class Identification

| | |
|--------------------------|---|
| Class Name | Argument |
| Class Description | This Class contains Data Members used to identify the Argument uniquely |
| Class Inheritance | None |

| | |
|---------------------------|----------|
| Classes Referenced | Function |
| Sub System | None |
| Class Type | Complete |
| Change History | None |

Attributes (or Data members)

| Attribute Name | Description | Type | Scope | Range |
|----------------|--------------------------|--------|--------|-------|
| sArgumentName | Stores the Argument Name | String | Public | |
| sArgumentType | Stores the Argument Type | String | Public | |

XLIII Main Program

Class Identification

| | |
|--------------------------|--|
| Class Name | Main Program |
| Class Description | This Class contains Data Members and functions used by the whole project |
| | |

| | |
|---------------------------|----------|
| Class Inheritance | None |
| Classes Referenced | None |
| Sub System | None |
| Class Type | Complete |
| Change History | None |

Attributes (or Data members)

| Attribute Name | Description | Type | Scope | Range |
|----------------|----------------------------------|---------|--------|-------|
| iCount | Stores the count of no Authors | Integer | Public | |
| iCount1 | Stores the count of no Classes | Integer | Public | |
| iCount2 | Stores the count of no Variables | Integer | Public | |
| iCount3 | Stores the count of no Functions | Integer | Public | |
| iCount4 | Stores the count of no Arguments | Integer | Public | |

| | | | | |
|------|---------------------------------------|-----------|--------|--|
| Str1 | Stores the filename | String | Public | |
| Proj | Creates a reference for project class | Reference | Public | |

Methods (or class functions)

Method 1

| | |
|--|------------------------------|
| Method Signature | MainProgram() |
| Purpose and algorithm description | Constructor for main program |
| Screen validations, if any | None |
| Tables/Files Used | None |
| Scope of method | Public |

Method 2

| | |
|--|--|
| Method Signature | btnInput(Object,EventArgs) |
| Purpose and algorithm description | Function for event for Input Button used |

| | |
|-----------------------------------|---------|
| Screen validations, if any | None |
| Tables/Files Used | None |
| Scope of method | Private |

Method 3

| | |
|--|---|
| Method Signature | btnOutputCode(Object,EventArgs) |
| Purpose and algorithm description | Function for event for Output Button used |
| Screen validations, if any | None |
| Tables/Files Used | None |
| Scope of method | Private |

Method 4

| | |
|--|------------------------------|
| Method Signature | btnOutFile(Object,EventArgs) |
| Purpose and algorithm description | |

| | |
|-----------------------------------|---|
| | Function for event if Output File Button used |
| Screen validations, if any | None |
| Tables/Files Used | None |
| Scope of method | Private |

Method 5

| | |
|--|--|
| Method Signature | btnJava(Object,EventArgs) Private |
| Purpose and algorithm description | Function for event if Java Button is Pressed |
| Screen validations, if any | None |
| Tables/Files Used | None |
| Scope of method | Private |

Method 6

| | |
|--|--|
| Method Signature | btnCSharp(Object,EventArgs) |
| Purpose and algorithm description | Function for event if CSharp Button is Pressed |
| Screen validations, if any | None |
| Tables/Files Used | None |
| Scope of method | Private |

Method 7

| | |
|--|---|
| Method Signature | btnC(Object,EventArgs) |
| Purpose and algorithm description | Function for event if C Button is Pressed |
| Screen validations, if any | None |
| Tables/Files Used | None |
| Scope of method | |

| | |
|--|---------|
| | Private |
|--|---------|

Method 8

| | |
|--|---|
| Method Signature | btnCplusplus(Object,EventArgs) |
| Purpose and algorithm description | Function for event if C++ Button is Pressed |
| Screen validations, if any | None |
| Scope of method | Private |

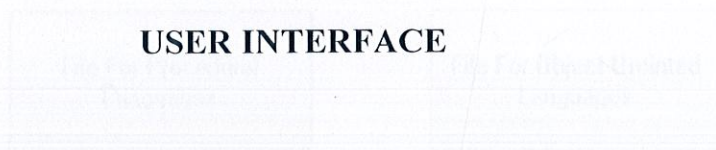
Chapter XII

User Interface

In this form the user selects his desired language whether Procedural or Object Oriented Language whichever is required. On selecting that the user is redirected to the next form which is the form for the xml input file.

Chapter XII

USER INTERFACE



The image shows a user interface form with two main sections. The left section is titled 'For Procedural' and contains a text input field and a 'Select' button. The right section is titled 'For Object Oriented' and contains a text input field and a 'Select' button. Below these sections, there is a 'Cancel' button. The form is designed to allow a user to select a language (Procedural or Object Oriented) for their input file.

Figure 10 Selection Form

Chapter XII

User Interface

In this form the user selects his desired language whether Procedural or Object Oriented Language whichever is required. On selecting that the user is redirected to the next form which is the form for the xml input file.

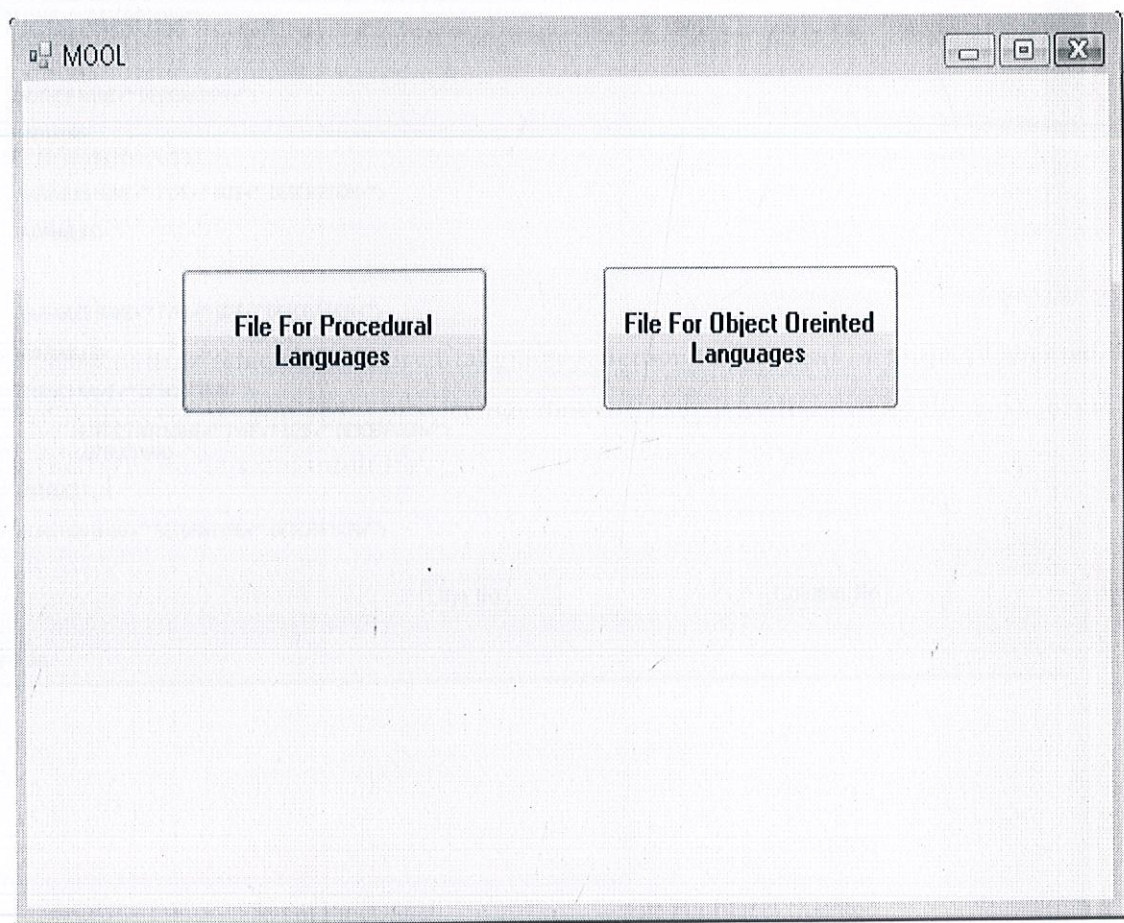


Figure 10: Selection Form

In this form the user inputs his required values in the xml input file template provided. The values are written into the corresponding tags provided for them for eg. The name of the project is written into the PROJECTNAME tag, Name of the author in the AUTHORNAME tag etc.

MOOL

Edit Format Help

Requirements File

```
<PROJECT NAME="" DESCRIPTION="">

  <AUTHOR NAME="" DESIGNATION="">
  </AUTHOR>

  <VARIABLES NAME="" TYPE="" SIZE="" DESCRIPTION="">
  </VARIABLES>

  <VARIABLES NAME="" TYPE="" SIZE="" DESCRIPTION="">
  </VARIABLES>

  <STRUCT NAME="" DESCRIPTION="">
    <STRUCTVAR NAME="" TYPE="" SIZE="" DESCRIPTION="">
    </STRUCTVAR>
  </STRUCT>

  <FUNCTION NAME="" RETURN TYPE="" DESCRIPTION="">
```

Line No Column No

Error File

Figure 11: Notepad Form

In case of errors in the input xml file they are reflected in the Error File list with the corresponding line number where the error has occurred. Errors like missing tag values or some other inconsistencies are shown in the list. On clicking the error in the error list the corresponding line gets highlighted in the input xml file as shown below.

The screenshot shows a window titled 'MOOL' with a menu bar (File, Edit, Format, Help) and a title bar 'Requirements File'. The main text area contains XML code for an 'Employee Management' system. An arrow points to the 'SIZE' attribute in the 'empid' variable tag, which is empty. Below the code area are input fields for 'Line No' and 'Column No'. At the bottom, there is an 'Error File' section with a list of error messages. An arrow points to the first error message, 'Error at line 17 VARIABLESIZE is Empty'. A button labeled 'Enter' is located at the bottom right of the window.

```

<PROJECT NAME="Employee" DESCRIPTION="Employee Management">

  <AUTHOR NAME="mohit" DESIGNATION="Engineer">

    </AUTHOR>

    <CLASS NAME="Employee" ACCESS="private">

      <VARIABLES NAME="empid" TYPE="int" SIZE="" ACCESS="public" DESCRIPTION="stores the value">

        </VARIABLES>

      <VARIABLES NAME="empname" TYPE="" SIZE="10" ACCESS="public" DESCRIPTION="stores the name">

```

Error File

- Error at line 17 VARIABLESIZE is Empty
- Error at line 23 VARIABLETYPE is Empty
- Error at line 23 VARIABLETYPE should be either int,char,string or float
- Error at line 29 FUNCTIONNAME is Empty
- Error at line 85 Base Class access is private.It can not be inherited

Enter

Figure 12: Notepad Form

This is the main form where the actual code generation takes place. In the first Browse button of “Enter the input file name with path”, the path of the input xml file is put. In the second Browse button of “Enter the output file name with path”, the path of the required output file is put. And in the last Browse button of “Input Values” the path of the input value file is entered. This file is for verification of the input values put into the xml input file. The corresponding language button gets enabled according to the output filename entered i.e. if a .cs extension is entered in the output file path then the C Sharp button gets enabled. On clicking the corresponding language button the code gets converted successfully and a user can then exit the application on click of the exit button.

The screenshot shows a window titled "MOOL" with a standard Windows-style title bar (minimize, maximize, close buttons). The main content area is titled "Code Generator for MOOL". It contains three input fields, each with a "Browse" button to its right:

- Input field: "Enter the input file name with path" followed by a text box and a "Browse" button.
- Input field: "Enter the output file name with path" followed by a text box and a "Browse" button.
- Input field: "Input Values" followed by a text box and a "Browse" button.

Below these fields, there are two rows of buttons:

- Row 1: Four buttons labeled "C", "C#", "C++", and "JAVA".
- Row 2: Two buttons labeled "ERROR" and "EXIT".

Figure 13: Main Form

Screen Shots of Output Files:

C Sharp Output

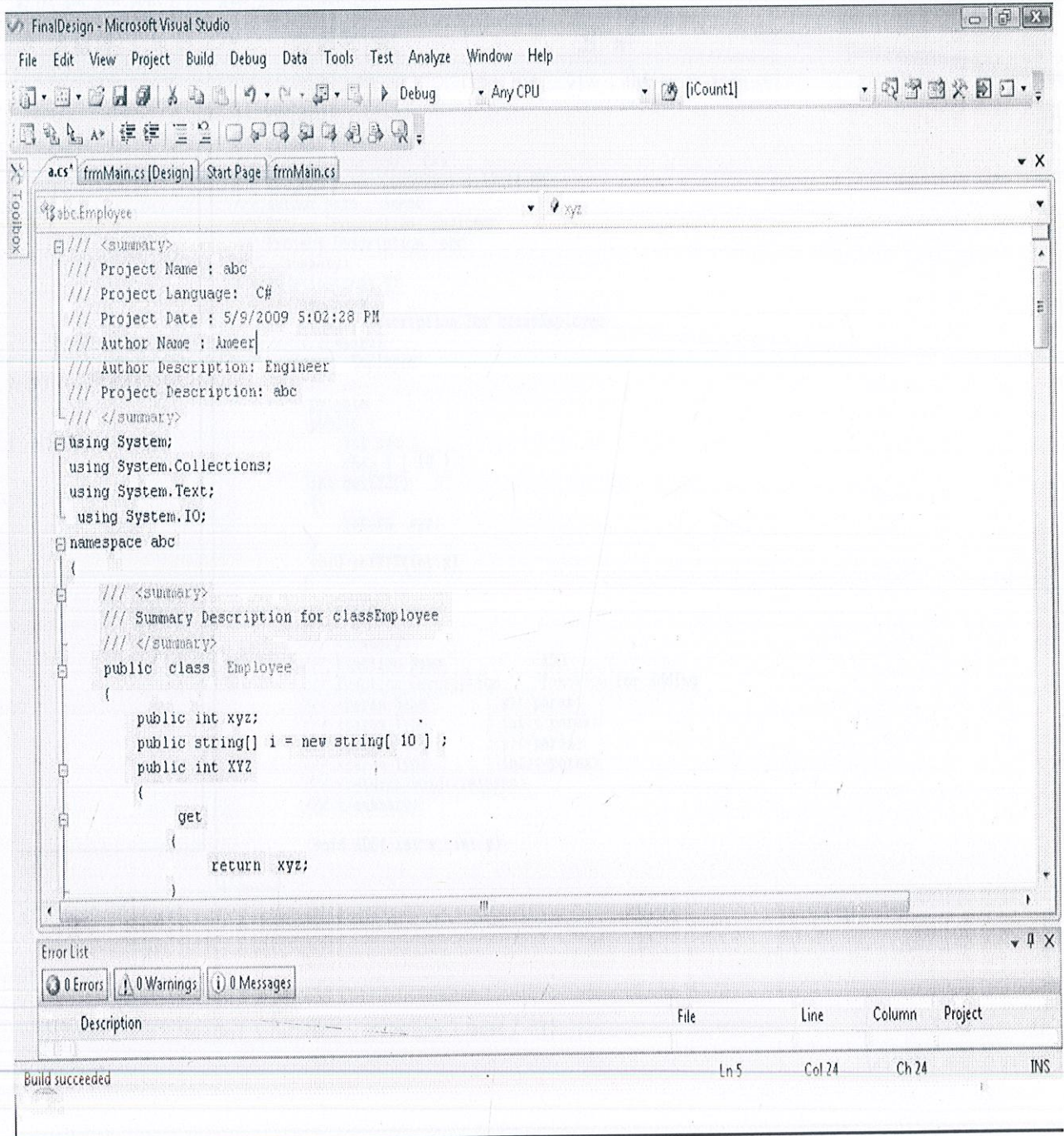


Figure 14: C Sharp Screen shot

Microsoft Visual C++ - [C:\...\a.cpp]

File Edit View Insert Project Build Tools Window Help

Standard toolbar icons: New, Open, Save, Print, Undo, Redo, Cut, Copy, Paste, Find, Replace, etc.

```

/// <summary>
/// Project Name : abc
/// Project Language: C++
/// Project Date : 5/9/2009 4:57:14 PM
/// Author Name : Ameer
/// Author Description: Engineer
/// Project Description: abc
/// </summary>
#include<iostream.h>
/// <summary>
/// Summary Description for classEmployee
/// </summary>
class Employee
{
    private :
    public :
        int xyz ;
        char i [ 10 ] ;
    int getXYZ()
    {
        return xyz;
    }
    void setXYZ(int x)
    {
        xyz=x;
    }
    /// <summary>
    /// Function Name : ADD
    /// Function Description : function for adding
    /// <param Name : x></param>
    /// <param Type : int></param>
    /// <param Name : y></param>
    /// <param Type : int></param>
    /// <returns>void</returns>
    /// </summary>

    void ADD( int x, int y)

```


Build Debug Find in Files 1 Find in Files 2 SQL Debugging Results

Ready Ln 5, Col 24 REC COL OVR READ

Page | 72

Java Output File:

Parent Class File



```
Employee - Notepad
File Edit Format View Help

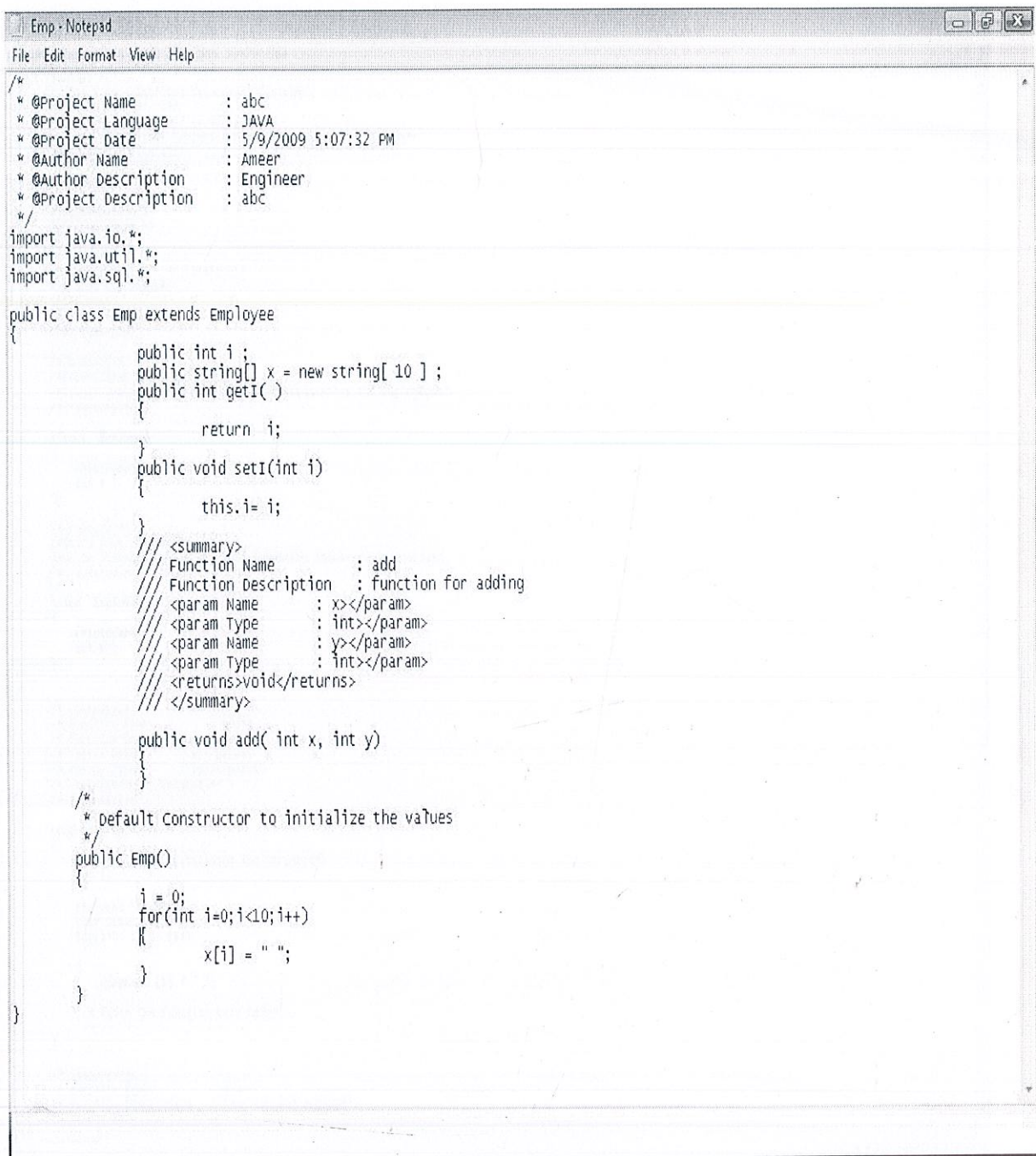
/*
 * @Project Name      : abc
 * @Project Language  : JAVA
 * @Project Date      : 5/9/2009 5:07:32 PM
 * @Author Name       : Ameer
 * @Author Description : Engineer
 * @Project Description : abc
 */
import java.io.*;
import java.util.*;
import java.sql.*;

public class Employee
{
    public int xyz;
    public String [] i = new String [10];
    public int getxyz()
    {
        return xyz;
    }
    public void setxyz(int xyz)
    {
        this.xyz= xyz;
    }
    /// <summary>
    /// Function Name      : ADD
    /// Function Description : function for adding
    /// <param Name       : x></param>
    /// <param Type        : int></param>
    /// <param Name       : y></param>
    /// <param Type        : int></param>
    /// <returns>void</returns>
    /// </summary>
    public void Add( int x, int y)
    {
    }
    /// <summary>
    /// Function Name      : sub
    /// Function Description : function for subtraction
    /// <param Name       : x></param>
    /// <param Type        : int></param>
    /// <param Name       : y></param>
    /// <param Type        : int></param>
    /// <returns>void</returns>
    /// </summary>
    public void sub( int x, int y)
    {
    }
}

/*
 * Default Constructor to initialize the values
 */
```

Figure 16: Java Screen shot

Java's Second Inherited Class File Output:



```
Emp - Notepad
File Edit Format View Help

/*
 * @Project Name      : abc
 * @Project Language  : JAVA
 * @Project Date      : 5/9/2009 5:07:32 PM
 * @Author Name       : Ameer
 * @Author Description : Engineer
 * @Project Description : abc
 */
import java.io.*;
import java.util.*;
import java.sql.*;

public class Emp extends Employee
{
    public int i ;
    public String[] x = new String[ 10 ] ;
    public int getI()
    {
        return i;
    }
    public void setI(int i)
    {
        this.i= i;
    }
    /// <summary>
    /// Function Name      : add
    /// Function Description : function for adding
    /// <param Name       : x></param>
    /// <param Type        : int></param>
    /// <param Name       : y></param>
    /// <param Type        : int></param>
    /// <returns>void</returns>
    /// </summary>
    public void add( int x, int y)
    {
    }

    /*
     * default Constructor to initialize the values
     */
    public Emp()
    {
        i = 0;
        for(int i=0;i<10;i++)
        {
            x[i] = " ";
        }
    }
}
```

Figure 17: Java Screen shot

C Output File:



```
hghk - Microsoft Visual C++ - [C:\Users\401517\Desktop\aid.c]
File Edit View Insert Project Build Tools Window Help

/* (summary) */
/* Project Name : Employee Management System */
/* Project Language : C */
/* Project Date : 5/3/2009 12:00:00 AM */
/* Project Description: Project for employee management system */
/* Author Name : utkarsh */
/* Author Description: Engineer */
/* (summary) */

#include<stdio.h>
#include<conio.h>

/* Global Variable declaration */
/* stores the value */
int iCount;
/* stores the name */
char cName[10];

/* (summary) */
/* Struct Name: Employee */
/* Struct Description: Stores the information about the employees */
/* (summary) */

struct Employee
{
    /* integer */
    int x;
};

/* (summary) */
/* Union Name: Employee */
/* Union Description: Stores the information about the employees */
/* (summary) */

union Employee
{
    /* integer */
    int x;
};

/* (summary) */
/* Function Name : add */
/* Function Description : function for adding */
/* (para Name : x) (para) */
/* (para Type : int) (para) */
/* (returns) void (returns) */
/* (summary) */

void add( int x)
{
    /* variable for initializing the variable */
    int i;


    /* stores the name */
    char cExpanse[10];
    for(i=0; i<10; i++)
    {
        cExpanse[i] = ' ';
    }
    /* Write the function body here */
}

/* (summary) */
/* Function Name : sub */
/* Function Description : function for adding */
/* (para Name : a) (para) */
```

Figure 18: C Screen shot

Output of the Input Values Verification File

This file is used to verify the inputs of the xml input file.



```
Employee
5/7/2009 4:29:10 PM
5/7/2009 4:29:10 PM
Employee Management
mohit
Engineer
Employee
public
empid
int
1
public
stores the value
empname
string
10
public
stores the name
add
void
public
function for adding
x
int
y
int
Emp
public
private
Employee
empname
string
10
public
stores the name
empid
int
1
public
stores the value
add
void
public
function for adding
x
int
y
int
s
abc
x
int
integer
y
int
integer
add
int
adds
x
int
y
int
```

Figure 19: Input Values Screen shot

Chapter XIII

Future Scope

The system that has been introduced is a simple and efficient design which can be further extended to include support for other languages like object languages or any other programming paradigm like a script oriented languages. The support for various built-in modules like I/O module, linked list module can also be added which will reduce the work of the programmer in writing the same code again and again.

Various language constraints especially in inheritance can also be added moreover support for pointers in C and C++ can also be added.

The user interface can also be made more attractive by allowing the auto-generation of the GUI file that acts as input to the system. The tool can also be designed to be deployed on the internet which would make it more accessible.

Finally, it can be said that since an iterative model has been followed the proposed design can be improved further.

Chapter XIII

FUTURE SCOPE

Chapter XIII

Future Scope

The design that has been proposed is a simple and effective design which can be further extended to include support for other languages like Visual Basic or any other programming paradigm like Aspect oriented Languages. The support for various built in reusable components like Login Module, Linked List, Stacks can also be added which will reduce the work of the programmer in writing the same code again and again.

Various Language constraints especially in inheritance can also be added moreover support for pointers in C and C++ can also be added.

The user interface can also be made more innovative by allowing the auto generation of the XML file that acts as input to the system. The tool can also be designed to be deployed on the internet which would make the tool widely available.

Finally, it can be said that since an iterative model has been followed the proposed design can be improved further

Chapter XIV

Conclusion

Model for support business language is an intelligent tool for writing the code in various programming languages like C++, JAVA and C# in a proper commented and indented format.

When the user starts the tool, he is asked to select the programming paradigm and accordingly the sample XML file in which the user inputs his requirements will be provided to him.

The user saves the XML file and loads this XML file in the next form which acts as input to our system.

Chapter XIV

Then the user selects the file name along with the extension for example for a C# file, he would name the file as program.cs. In the third textbox, the user inputs a text file which shows the contents of all the values that he entered for the input xml file.

CONCLUSION

In case of errors, an error file is generated that shows all the errors along with the first number of error.

On the click of corresponding error line the corresponding line on the input XML file is highlighted. So it makes the error rectification very easy for the user. And unless a user doesn't clear all the errors in XML file he can't proceed further to code conversion.

Chapter XIV

Conclusion

Model for object oriented language is an intelligent tool for writing the code in various programming languages like C/C++/JAVA and C# in a proper commented and indented format.

When the user starts the tool, he is asked to select the programming paradigm and accordingly the sample XML file in which the user inputs his requirements will be provided to him.

The user saves the XML file and loads this XML file in the next form which acts as input to our system.

Then the user selects the file name along with the extension for example for a c# file, he would name the file as program.cs. After this in the third textbox the user inputs a text file which shows the contents of all the values that he entered for the input xml file.

In case of errors, an error file is generated that shows all the errors along with the line number of error.

On the click of corresponding error line the corresponding line on the input XML file is highlighted. So it makes the error rectification very easy for the user. And unless a user doesn't clear all the errors in XML file he can't proceed further to code conversion.

- ## Chapter XV

Chapter XV

APPENDIX

XV.I References

XV.II Bibliography

Chapter XV

Appendix

XV.I References

- <http://en.wikipedia.org/wiki/Xml>
- <http://msdn.microsoft.com/en-us/library/system.io.file.aspx>
- [http://msdn.microsoft.com/en-us/library/system.string\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/system.string(VS.80).aspx)
- http://www.w3schools.com/XML/xml_whatIs.asp
- <http://support.microsoft.com/kb/307548>
- <http://msdn.microsoft.com/en-us/library/system.xml.aspx>
- <http://msdn.microsoft.com/en-us/library/system.environment.getfolderpath.aspx>
- <http://msdn.microsoft.com/en-us/library/system.windows.controls.richtextbox.aspx>
- [http://msdn.microsoft.com/en-us/library/system.xml.entityhandling\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/system.xml.entityhandling(VS.71).aspx)
- [http://msdn.microsoft.com/en-us/library/system.windows.forms.textboxbase.selectionstart\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/system.windows.forms.textboxbase.selectionstart(VS.71).aspx)
- <http://msdn.microsoft.com/en-us/library/system.xml.xmlexception.linenumber.aspx>
- <http://msdn.microsoft.com/en-us/library/system.xml.xmlexception.linenumber.aspx>
- <http://msdn.microsoft.com/en-us/library/system.xml.xmltextreader.isstartelement.aspx>

XV.II Bibliography

- C#: The Complete Reference By Schildt, Herbert
- XML: The Complete Reference By Heather, Williamson