CHAPTER 1

INTRODUCTION

Code generation (compiler), a mechanism to produce the executable form of computer programs with the help of different computer languages. Codraw uses two UML diagrams, class diagram and activity diagram. Class diagram provides structural view of the java code and Activity diagram gives behavioural view of the java code. Manually translated UML diagrams into java code may produce human errors. But java code obtained from transformation tool eliminates human errors and also provides better accuracy.

Codraw helps to develop small application which can be used to diminish outsourcing. User should have proper knowledge about the different UML diagrams for easily drawing class diagram and activity diagram. UML diagram is a general purpose, collection of modelling techniques for visualizing and documenting the artefacts of a software system. Amongst multiple Model driven architecture (MDA) tools, Codraw belongs to transformation tool.

Unified Modeling Language (UML)

Unified Modeling Language is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created, by the Object Management Group. It was first added to the list of OMG adopted technologies in 1997, and has since become the industry standard for modeling software-intensive systems.

UML includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems. The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artefacts of an object-oriented software-intensive system under development.

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artefacts of an object-oriented software-intensive system under development

Class Diagram

In software engineering, a **class diagram** in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

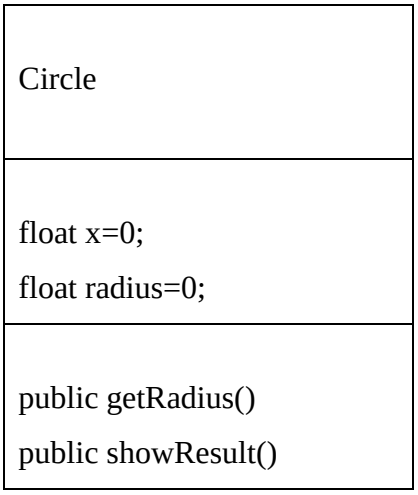

fig 1.1:Example of class diagram

- The upper part holds the name of the class.
- The middle part contains the attributes of the class.
- The bottom part gives the methods of the class can take or undertake.

Classes

A class is the description of a set of objects having similar attributes, operations, relationships and behaviour. A *class* is the blueprint from which individual objects are created. Classes are the fundamental building blocks. A class defines constituent members which enable these class instances to have *state* and *behaviour*

A class is nothing but a blueprint or a template for creating different objects which defines its properties and behaviors. Java class objects exhibit the properties and behaviors defined by its class. A class can contain fields and methods to describe the behavior of an object.

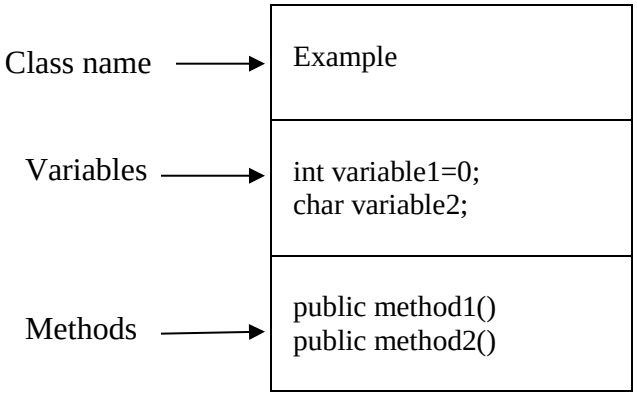

Fig1.2: Class Structure

Attribute: An *attribute* is a named property of a class that describes the object being modelled. In the class diagram, attributes appear in the second compartment just below the name-compartment.

It describes a class that has been selected for inclusion in the analysis model. Attribute are written in the second compartment of a class notation.

Syntax: Attribute_Name: Data_Type

Operation: Operations describe the class behaviour and appear in the third compartment. It defines the behaviour of an Object.

Different types of operations are:

- Operations that manipulate data.
- Operations that perform computation.
- Operations that enquire about the state of an object.
- Operations that monitor an object.

It is written in the third Compartment of a class notation.

Syntax: Operation_Name: Return_Type where Return_Type is option [6]

Model Driven Engineering

Model-driven architecture (MDA) is a technique to develop software. Model-driven architecture supports model-driven engineering of software systems. Model driven engineering Describes an approach for development of software where models are used as the primitive source for documenting, analyzing, designing, constructing, deploying and maintaining a system.

An MDA tool can be one of following types: Simulation, Analysis, Reverse Engineering Transformation, Composition, Test, Creation. MDA tools are used to transform, compare, interpret, align measure, develop, verify, etc. models .A model is a representation or description of system. Model is interpreted as “UML Model”.

The UML is a general-purpose, tool-supported, and collection of modeling techniques for visualizing, constructing, and documenting the artifacts of a software system. Amongst multiple MDA Tools, our tool belongs to Transformation tool named “CoDraw”. Model transformation is the technique for transforming one model to another in the same system. The transformation collaborates the platform independent model with some external information to generate a platform specific model [1].

1.1 Background Motivation

The amount of various software in Latest systems is growing up. Its increase in difficulty combined with timely restrictions has motivated the investigations for software that should be able to reduce costs and accelerate the product delivery. Usually models are used to handle system with complexity through graphical views and abstraction. UML is the standard language for modeling and provides various graphical diagrams to give different views of a system and has been considered significant in the field of model complex systems. When models are used, these must be translated to code to obtain an implementation.

Recently, Model-driven Engineering techniques, used by the software organization, have gained attraction in the software community, promising automation and abstraction for software development [2].

1.2 Problem statement

The proposed system is an approach for code generation from UML models able to generate structural and behavioral code. To validate our approach, a tool was developed, which captures a UML model, composed of a class diagram and activity diagram which generates automated executable java code from it.

1.3 Requirement Analysis

Requirements analysis is critical to the success of a software project. Hence the requirements should be well documented, measurable, testable and related to identified needs of our project, and defined to a level of detail sufficient for system design.

Requirement gathering and Analysis phase started at the beginning of our project, we had formed groups and modularized the project. Important points of consideration were:

1. Define and visualize all the objectives clearly.
2. Gather requirements and evaluate them.
3. Consider the technical requirements needed and then collect technical

specifications of various peripheral components (hardware) required.

4. Analyze the coding languages needed for the project.
5. Define coding strategies.
6. Analyze future risks / problems.
7. Define strategies to avoid these risks else define alternate solutions to these risks.
8. Check financial feasibility.
9. Define Gantt charts and assign time span for each phase.