ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 7, Issue 2, pp: (120-127), Month: April - June 2019, Available at: www.researchpublish.com

Generation of Code from UML Class diagram using packages in ArgoUML

Renu Dangi

Assistant Professor, Medi-Caps University, Indore (MP), India
renu.dangi@medicaps.ac.in

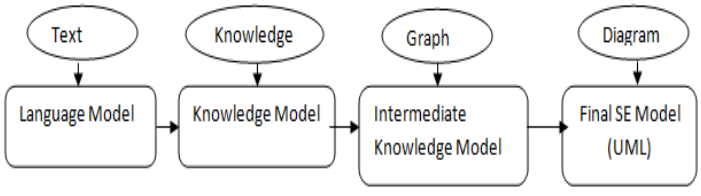
Abstract: **Since the establishment of the Unified Modelling Language (UML) as a standard graphical notation for representing knowledge, new ideas have emerged about tools that can automatically extract knowledge from text and represent it with UML diagrams. The goal of UML can be defined as a simple modelling mechanism to model all possible practical systems in today's complex environment. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard.**

Keywords: **UML, Class diagram, Package, Dependencies on Package.**

1. INTRODUCTION

Knowledge can be represented in variety of forms. In Software Engineering (SE), for example, perhaps the most common way of representing knowledge is with diagrams. This way of representation fits the understanding of a wide range of users. Graphical representation is done with self explanatory shapes, it is semi-formal, and is suitable for subsequent formal processing into program code. This type of knowledge representation is easy to understand and widespread in information technology.

UML (Unified Modelling Language) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. It was initially started to capture the Behaviour of complex software and non-software system and now it has become an OMG standard.UML stands for Unified Modelling Language.UML is different from the other common programming languages such as C++, Java, COBOL, etc.UML is a pictorial language used to make software blueprints.UML can be described as a general purpose visual Modelling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well.



Block Diagram for Basic Conceptual Model

2. RELATED WORK

A Conceptual Model of UML. To understands the conceptual model of UML, first we need to clarify what is a conceptual model? And why a conceptual model is required?

- A conceptual model can be defined as a model which is made of concepts and their relationships.
- A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other.

As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually. The conceptual model of UML can be mastered by learning the following three major elements −

- UML building blocks
- Rules to connect the building blocks
- Common mechanisms of UML

Object-Oriented Concepts

UML can be described as the successor of object-oriented (OO) analysis and design.

An object contains both data and methods that control the data. The data represents the state of the object. A class describes an object and they also form a hierarchy to model the real-world system. The hierarchy is represented as inheritance and the classes can also be associated in different ways as per the requirement.

Objects are the real-world entities that exist around us and the basic concepts such as abstraction, encapsulation, inheritance, and polymorphism all can be represented using UML.

UML is powerful enough to represent all the concepts that exist in object-oriented analysis and design. UML diagrams are representation of object-oriented concepts only. Thus, before learning UML, it becomes important to understand OO concept in detail.

Following are some fundamental concepts of the object-oriented world −

- Objects − Objects represent an entity and the basic building block.
- Class − Class is the blue print of an object.
- Abstraction − Abstraction represents the behaviour of a real world entity.
- Encapsulation − Encapsulation is the mechanism of binding the data together and hiding them from the outside world.
- Inheritance − Inheritance is the mechanism of making new classes from existing ones.
- Polymorphism − It defines the mechanism to exists in different forms.

OO Analysis and Design

OO can be defined as an investigation and to be more specific, it is the investigation of objects. Design means collaboration of identified objects.

Thus, it is important to understand the OO analysis and design concepts. The most important purpose of OO analysis is to identify objects of a system to be designed. This analysis is also done for an existing system. Now an efficient analysis is only possible when we are able to start thinking in a way where objects can be identified. After identifying the objects, their relationships are identified and finally the design is produced.

The purpose of OO analysis and design can describe as −

- Identifying the objects of a system.
- Identifying their relationships.
- Making a design, this can be converted to executables using OO languages.

There are three basic steps where the OO concepts are applied and implemented. The steps can be defined as

OO Analysis → OO Design → OO implementation using OO languages

The above three points can be described in detail as −

- During OO analysis, the most important purpose is to identify objects and describe them in a proper way. If these objects are identified efficiently, then the next job of design is easy. The objects should be identified with responsibilities. Responsibilities are the functions performed by the object. Each and every object has some type of responsibilities to be performed. When these responsibilities are collaborated, the purpose of the system is fulfilled.

- The second phase is OO design. During this phase, emphasis is placed on the requirements and their fulfilment. In this stage, the objects are collaborated according to their intended association. After the association is complete, the design is also complete.

- The third phase is OO implementation. In this phase, the design is implemented using OO languages such as Java, C++, etc.

Role of UML in OO Design

UML is a modelling language used to model software and non-software systems. Although UML is used for non-software systems, the emphasis is on Modelling OO software applications. Most of the UML diagrams discussed so far are used to model different aspects such as static, dynamic, etc. Now whatever be the aspect, the artifacts are nothing but objects.

Hence, the relation between OO design and UML is very important to understand. The OO design is transformed into UML diagrams according to the requirement. Before understanding the UML in detail, the OO concept should be learned properly. Once the OO analysis and design is done, the next step is very easy. The input from OO analysis and design is the input to UML diagrams.

As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually. The conceptual model of UML can be mastered by learning the following three major elements −UML building blocks, Rules to connect the building blocks, Common mechanisms of UML.

The building blocks of UML can be defined as −

Things

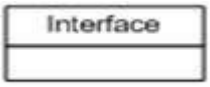
Things are the most important building blocks of UML. Things can be –

Structural Things

Structural things define the static part of the model. They represent the physical and conceptual elements. Following are the brief descriptions of the structural things.

Class − Class represents a set of objects having similar responsibilities.

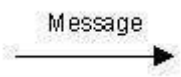
Interface − Interface defines a set of operations, which specify the responsibility of a class.



Behavioural Things

A Behavioural thing consists of the dynamic parts of UML models. Following are the Behavioural things −

Interaction − Interaction is defined as a Behaviour that consists of a group of messages exchanged among elements to accomplish a specific task.



Grouping Things

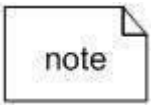
Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available −

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 7, Issue 2, pp: (120-127), Month: April - June 2019, Available at: www.researchpublish.com

Package − Package is the only one grouping thing available for gathering structural and Behavioural things.

An notational Things

An notational Things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. Note - It is the only one a notational thing available. A note is used to render comments, constraints, etc. of an UML element.



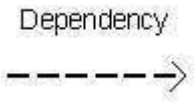
Relationship

Relationship is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships available.

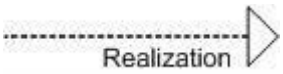
Dependency

Dependency is a relationship between two things in which change in one element also affects the other.



Realization

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility, which is not implemented and the other one implements them. This relationship exists in case of interfaces.



UML Diagrams

UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.

The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete.

UML includes the following nine diagrams:

Class diagram, Object diagram, Use case diagram, Sequence diagram, Collaboration diagram, Activity diagram, State-chart diagram, Deployment diagram, Component diagram.

UML Modelling Tool

ArgoUML:

How ArgoUML is Developed: Jason Elliot Robbins founded the Argo Project and provided early project leadership. While Jason remains active in the project, he has handed off project leadership. The project continues to move forward strongly.

To learn how the project is run and how to contribute to it, go the Argo UML Web Site Developer Zone [http://argouml.tigris.org/dev.html] and read through the documentation there. The Developers' Cookbook was written specifically for this purpose.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 7, Issue 2, pp: (120-127), Month: April - June 2019, Available at: www.researchpublish.com

Figure 1: Argo UML Tool

Class Diagram:

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

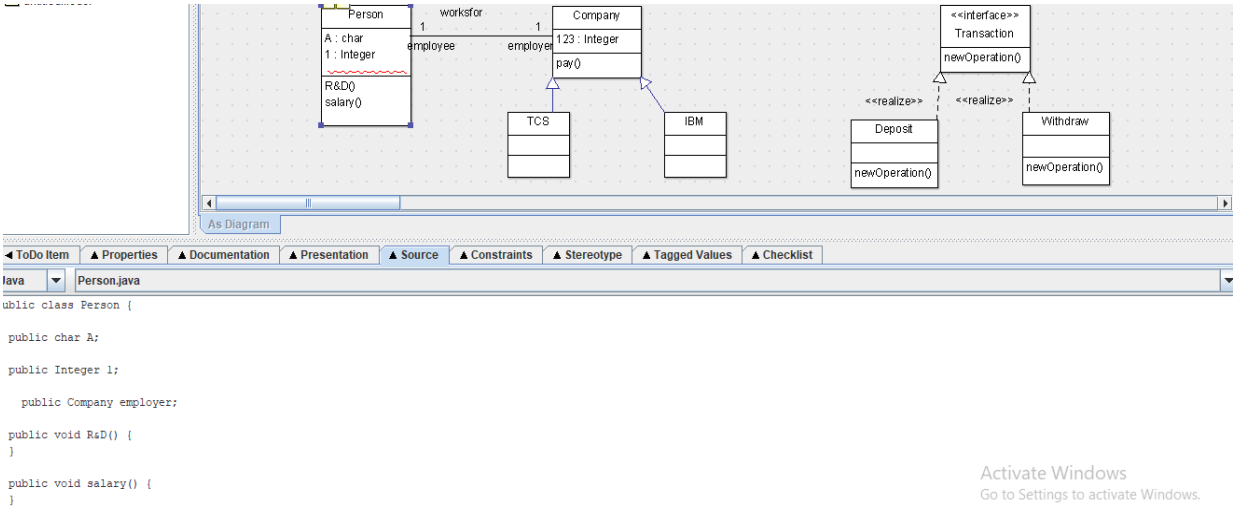
The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application; however class diagram is a bit different. It is the most popular UML diagram in the coder community.

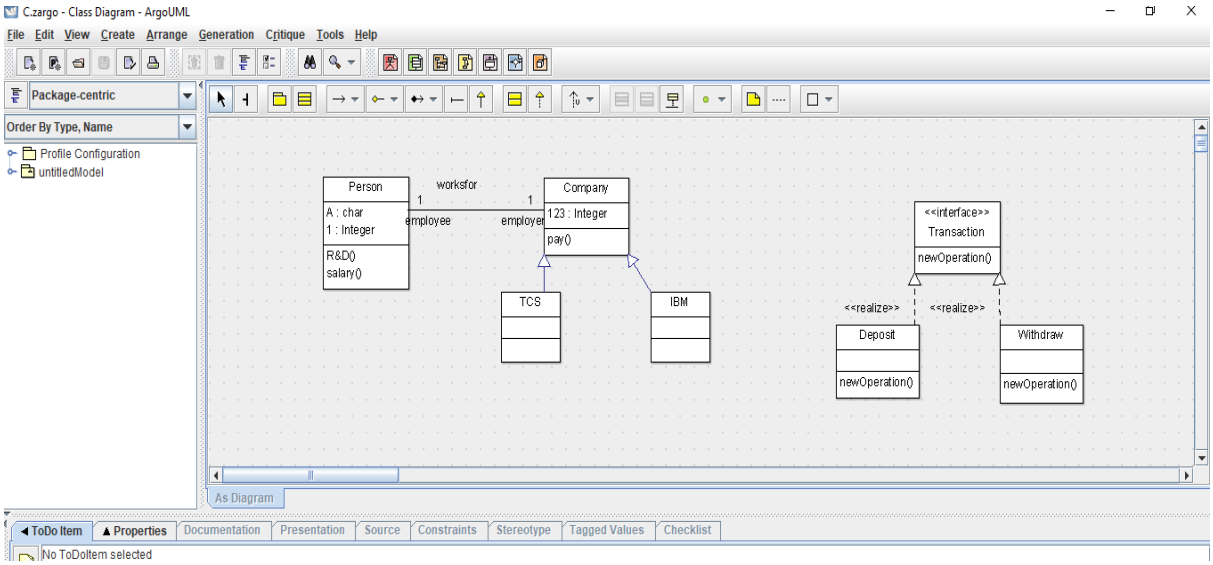
Class diagram is also considered as the foundation for component and deployment diagrams. Class diagrams are not only used to visualize the static view of the system but they are also used to construct the executable code for forward and reverse engineering of any system.

Generally, UML diagrams are not directly mapped with any object-oriented programming languages but the class diagram is an exception.

Class Diagram in Argo UML Tool



ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 7, Issue 2, pp: (120-127), Month: April - June 2019, Available at: www.researchpublish.com



Generating a code from Class Diagram in Argo UML Tool

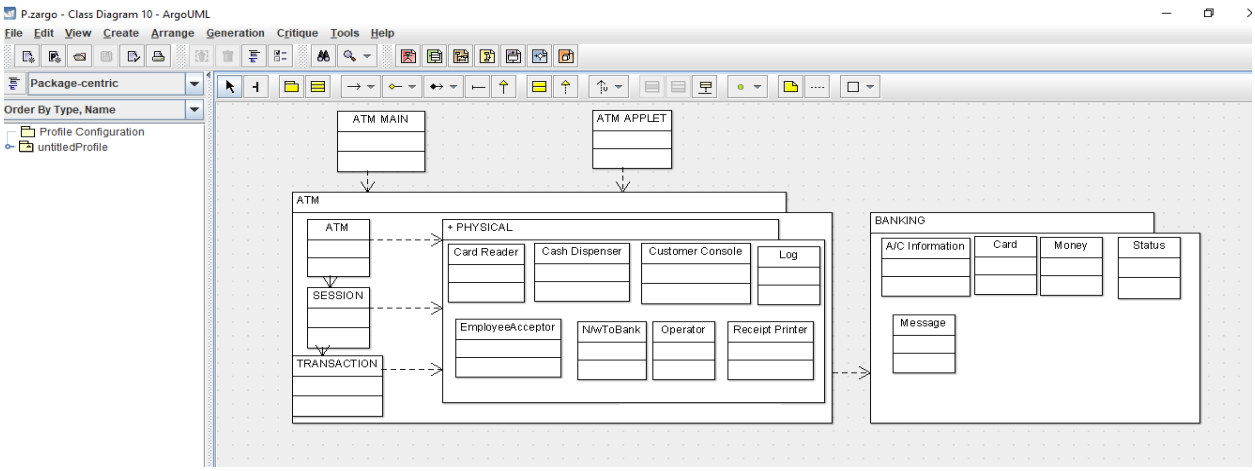
Source Code:

Java Person.java

```
public class Person {

  public char A;

  public Integer 1;

    public Company employer;

  public void R&D() {
  }

  public void salary() {
  }
```

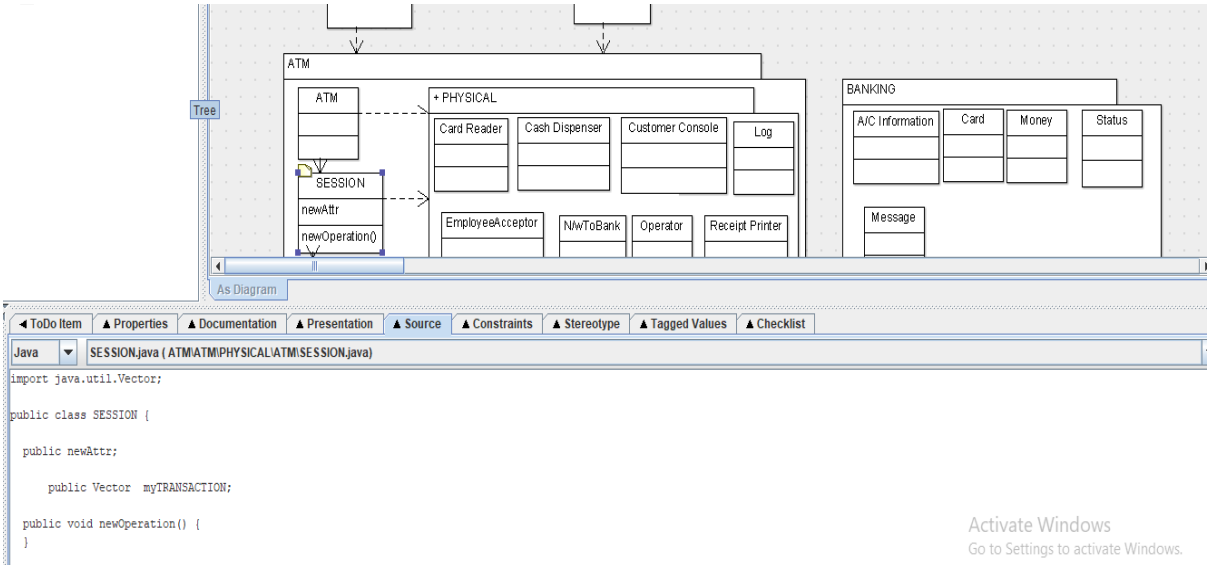
Dependencies on Packages

Class Diagram using Package in Argo UML Tool



ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 7, Issue 2, pp: (120-127), Month: April - June 2019, Available at: www.researchpublish.com

Generating a code from Class Diagram using Package in Argo UML Tool



Source code:

```
Java                SESSION.java ( ATM\ATM\PHYSICAL\ATM\SESSION.java)
import java.util.Vector;

public class SESSION {

  public newAttr;

       public Vector  myTRANSACTION;

  public void newOperation() {
  }
```

3. CONCLUSION

In this paper, the goal of UML can be defined as a simple modelling mechanism to model all possible practical systems in today's complex environment. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard. There are a number of goals for developing UML but the most important is to define some general purpose modelling language, which all modellers can use and it also needs to be made simple to understand and use.UML diagrams are not only made for developers but also for business users, common people, and anybody interested to understand the system. The system can be a software or non-software system. Thus it must be clear that UML is not a development method rather it accompanies with processes to make it a successful system.

REFERENCES

[1] Magda G. Ilieva, Harold Boley, REPRESENTING TEXTUAL REQUIREMENTS AS GRAPHICAL NATURAL LANGUAGE FOR UML DIAGRAM GENERATION, 2008.

[2] Burg, J.F.M. and van de Riet, R.P.: Analyzing Informal Requirements Specifications: A First Step towards Conceptual Modelling, Proc.of the 2nd Int. Workshop on Applications of Natural Language to Information Systems, Amsterdam, 1996.

[3] Fliedl, G.; Kop, Ch.; Mayerthaler, W.; Mayr, H.C.; Winkler Ch.: The NIBA workflow: From textual requirements specifications to UML-schemata In: ICSSEA, Paris, 2002.

[4] D. Richards, K. Böttger, O. Aguilera: A Controlled Language to Assist Conversion of Use Case Descriptions into Concept Lattices. In 15th Australian Joint Conference on AI, 2002

[5] Lee, B.-S., Bryant, B.R.: Automated conversion from requirements documentation to an object-oriented formal specification language. In Proceedings of SAC(ACM), Madrid, Spain, 2002.

[6] Mencl, V.: Deriving Behaviour Specifications from Textual Use Cases. In Proc of 'Workshop Intelligent Technologies for Software Engineering (WITSE, part of ASE), Linz, Austria, 2004.

[7] Unified Modelling Language (UML) https://www.tutorialspoint.com/uml/