STUDY ABOUT OBJECT ORIENTED MODELING

Salini Dev P V
Assistant Professor, Information Technology Department, Viswajyothi College of Engg. & Technology, Kerala, India

ABSTRACT
Software design is an important task in the software development life cycle. These designs should be unambiguous, precise, correct and complete. So, designing is an important activity in the software development life cycle. And also, designs can be easily understandable to both developers and customers. So, these designs are represented in many ways. One of the main representation methods is modeling. Different models are to express different levels of precision. Object Technology is a good method for creating models. No single model is sufficient. Object oriented modeling is a good way of mapping between real world things and system design. Unified Modeling Language (UML) is a standard tool used for modeling. It is used to visualize, Specify, construct and document a system.And these models can be converted into many programming languages like java, c, c++ etc. So, object oriented modeling is good method for system designing.

Keyword: - *Object oriented Technology, model, software, design.*

1. INTRODUCTION

Object Technology is a set of principles which are used for creating different models. Designing is an important activity in the software development life cycle[1]. Models are flexible to change and have well defined architecture. These models can be implemented in software using object oriented programming language. Object technology is not a theory. Implementation of object technology is a powerful tool to develop software. Object technology is used in client/server systems and web development and in real time systems. Model is a simplification of reality [1][10]. Model helps to visualize, specify, construct and document a system. Different models are to express different levels of precision. So, no single model is sufficient. Views are slices of models. Each view has structural and behavioral aspects [8]. Different views are,

- Use case view
- Logical view
- Implementation view
- Deployment view

2. OBJECT ORIENTATION

Object orientation helps to map between the real world and software [2]. That is, object orientation provides a direct mapping between concepts and code.

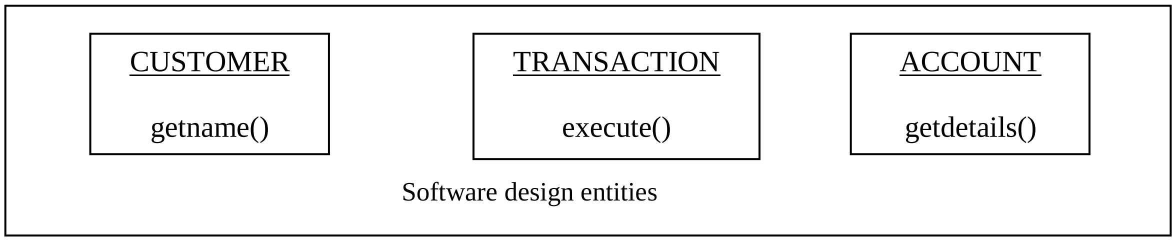

Figure-1 Object orientation

Figure1 shows the mapping between the real world concepts and the software. Basic principles of object orientation are[8]:

- Abstraction:
- Encapsulation
- Modularity
- Hierarchy

2.1 Object

An object is an entity. Object is an instance of a class.

2.2 Class

A class represents a concept. A class is an abstract definition of an object [9].

3. UML (UNIFIED MODELING LANGUAGE)

UML is a standard graphical language for specifying, visualizing, constructing and documenting the system. UML is used for expressing object oriented designs. Most UML designs are called Models [9].

Some of the main UML diagrams are,

3.1 Class diagram:

Classes in UML are represented using rectangles containing the class name. Class name is placed in the first part and the name is in bold face. Detailed version includes attributes, operations etc. If these are included, attributes are represented in the second part in the rectangle and operations are in the third part and comments can be placed in the fourth part. For example, Customer is a class , customer-name, customer-id, salary are attributes of the customer and getname(), getsalary() are some operations for the customer[7][8]. Different classes are connected with associations. Main parts of the class diagrams are ,

- Classes
- Relationship between classes: Association, Aggregation, Dependancy and Inheritance

In UML, this class is represented as,

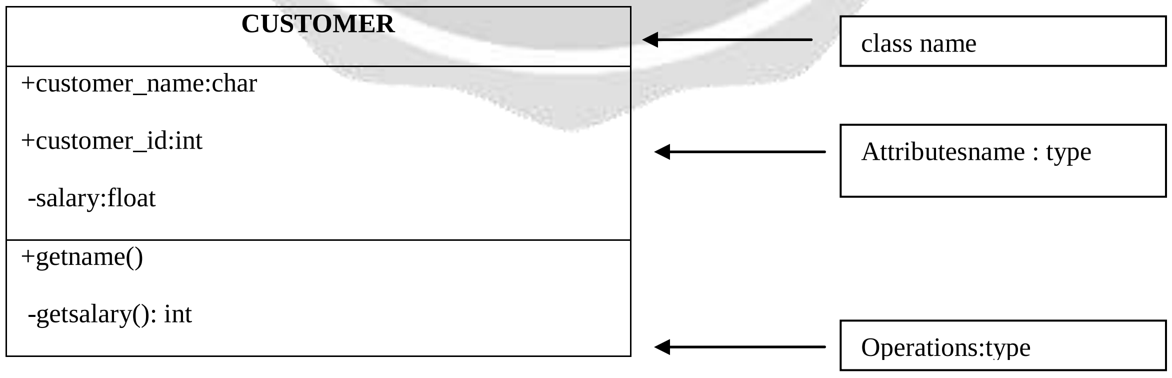

Figure-2 Example for a customer Class notation
+ : public visibility
- : private visibility

3.2 Object diagram:

Objects in UML are represented using rounded rectangles containing the object name. Class name is placed in the first part also in brackets.

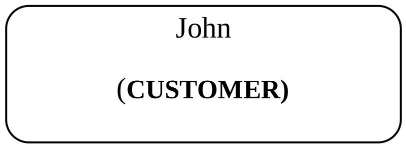

Figure-2 Example for a customer object notation

3.3 Use case Diagram

Use case Diagram is a represents the relationships between user and system. In this main parts are,

- **Actors: A**ctors are two types primary and secondary actors. Primary actors are laced in the top left corner of the use case diagram. Its UML notation is,

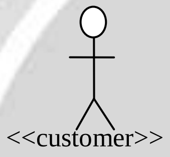

Figure-3 Actor notation

- **Use cases :** Use cases are system functions. User directly interact with the system through these functions. Use cases are represented using ellipse notation.

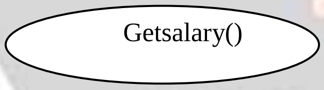

Figure-4 use case notation

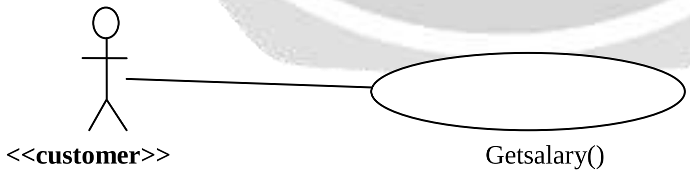

Figure-5 use case diagram

3.4 Activity diagram: Activity diagram represents work flow of an individual activity. Its main parts are activity and data flow[8].

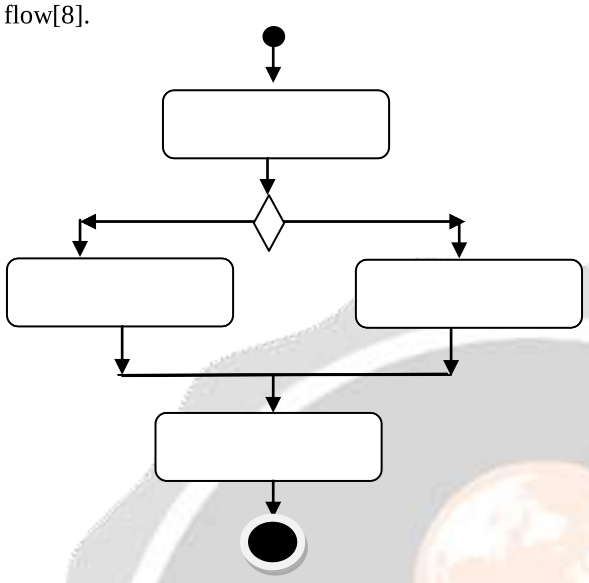

Figure-6 Activity diagram notations

3.5 Sequence diagram: Sequence diagram describes the interactions between objects in a sequential manner. Its main parts are objects and interactions between objects. Interactions between objects are represented using horizontal arrows and life times of objects are represented using vertical lines [2].

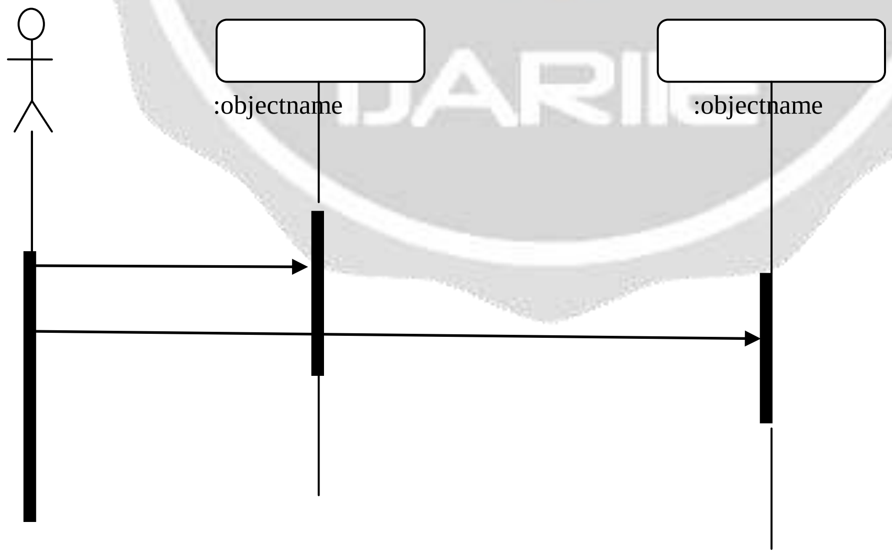

Figure-7 Sequence diagram notations

4. FORWARD ENGINEERING AND REVERSE ENGINEERING

Forward engineering and Reverse engineering is very useful for improving traceability between design and code. It is very helpful to be able to obtain the code element that implements it [4]. Reverse engineering is very useful to obtain the business rules of the object code which it contributes [5,6]. Understanding purposes tractability is very important, so forward and reverse engineering reduces the ambiguity of both the developers and the customers [7]. Therefore, forward and reverse engineering provide reconstruction capability by mapping between the design and its implementation.

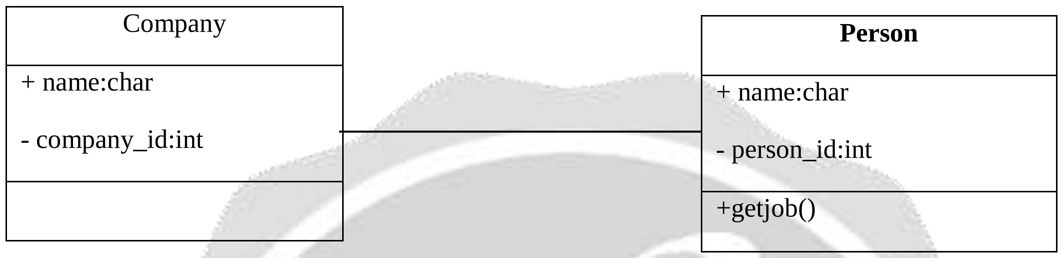

Figure-8 class diagram

The above class diagram in the Figure6 is converted into java language, the code is

```
public class  Company
{
      public char name[10];
      private int customer_id;
.
.
.
}
/*************************/
public class  person
{
      public char name[10];
      private int person_id;
.
.
.
}

      public  void getjob()
      {
                …
      }

}
```

5. CONCLUSIONS

Object Technology is a set of principles which are used for creating different models. Model is a simplification of reality [1]. Model helps to visualize, specify, construct and document a system. Object orientation provides a direct mapping between concepts and code. UML is a standard graphical language for specifying, visualizing, constructing and documenting the system. UML is used for expressing object oriented designs. Most UML designs are called Models [2]. Forward engineering and Reverse engineering is very useful for improving traceability between design and code. Object oriented modeling is good method for mapping real world and system designs. It can improve traceability, understandability and provide easy conveying between customer and developer. Design reconstruction is very easy by reverse engineering[2][3].

6. REFERENCES

[1]. Upasana_Choudhary Review on Reverse Engineering Techniques of Software Engineering, Article *in* International Journal of Computer Applications 119(14):7-10 · June 2015.
[2]. P. Aiken. Data Reverse Engineering: Slaying the Legacy Dragon. McGraw-Hill, 1995.
[3]. L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice. Addison-Wesley, 1997.
[4]. Hausi A. Muller , Jens H. Jahnke, Dennis B. Smith, Reverse Engineering: A Roadmap Draft of paper submitted to ICSE 2000 .
[5]. R. Brooks. Towards a theory of comprehension of computer programs. International Journal of Man-Machine Studies, 18:86–98, 1983.
[6]. R. Kazman, S. Woods, and S. Carri`ere. Requirements for integrating software architecture and reengineering models: CORUM II. In Proceedings of the Fifth Working Conference on Reverse Engineering (WCRE-98), Honolulu, Hawaii, USA, pages 154–163. IEEE Computer Society Press, October 1998.
[7]. R. Brooks. Towards a theory of comprehension of computer programs. International Journal of Man-Machine Studies, 18:86–98, 1983.
[8]. M. Weiser. Program slicing. IEEE Transactions on Software Engineering, SE-10(4):352–357, July 1984.
[9].Eric Braude text book of Sftware Design From Programming to Architecture.
[10]. IBM Rational University Rational Software Essentials of Visual Modeling with UML2.0 student guide.