International Journal of Innovative Technology & Adaptive Management (IJITAM)
ISSN: 2347-3622, Volume-1, Issue-2, November 2013

An Approach to Extended Class Diagram Model of UML for Object Oriented Software Design

Bipsha Mallick
Assistant Professor,
Haldia Institute Of Technology
Haldia, Purba Medinipur, W.B., INDIA.
bipasm@gmail.com

Nilanjan Das
Assistant Professor,
Siliguri Institute Of Technology,
Siliguri, W.B, INDIA
nilanjan.das81@gmail.com

Abstract: Unified Modelling Language is a language which defines sets of rules and vocabulary for conceptual and physical representation of system. UML Consists different types of diagram to visualize specify and construct the software system. Class diagram are one of the common type of UML diagram which is used to design static view of the system. So Class diagram play important role to design object oriented software System. In this article we give some ideas to extend features of Class diagram for object oriented software design. There is some extra features are added to the Class diagram and also modify some existing features. So our proposed idea is may be useful to design Object Oriented software.

Keywords: UML, UML diagram Class Diagram, Object Oriented Software, Object Oriented software design.

Introduction:

1.UML: UML is the modelling language .It is used to visualize, specify, construct and document the software intensive system. It includes set of graphical notation, like rectangle, circle, ellipse, line etc. to create visual model of the system.UML used some syntax and semantics same like other languages.UML is used for the purpose of document the object oriented system [1].

2.UML Diagram: There are nine types of diagram in UML and capture five different views of system. The different diagrams of UML are used to provide different perspective for development of the software system [1].

There are five views captures by UML diagrams are

2.1 **Use Case view**: It includes use cases that describe the behaviour of the system. Use Case diagram is used in this view [1].

2.2. **Design view**: Design view shows the structural view of the system. Class diagram and object diagram are used in this view [1].

2.3. **Process view**: The process view describes the dynamic behaviour of the system. State chart diagram, activity diagram, sequence diagram and collaboration are used in this view [1].

2.4. **Implementation view**: This view captures different components of the system and dependencies component diagram used in this view [1].

2.5. **Deployment view:** This view describes how the components are implemented on the different hardware .Deployment view include deployment diagram [1].

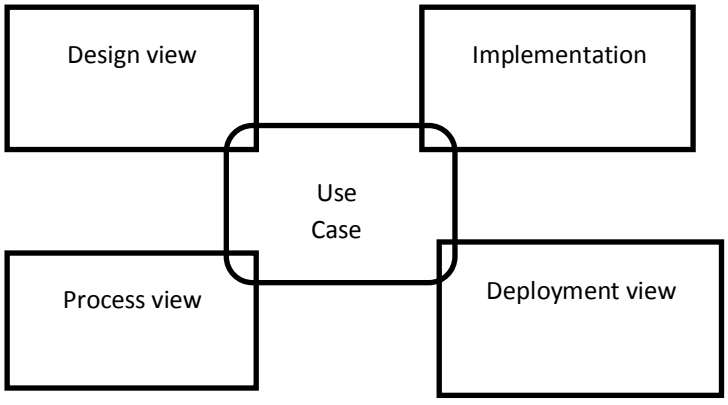


Fig 1: Architectural view of UML

Class Diagram: Class diagram provides set of classes, interfaces and their relationship. Class diagram is the common modelling technique in the object oriented system. This diagram presents the static view of the system.

Classes in the class diagram are basic building block of any object oriented system. Class is the collection of similar types of object. Attributes and operations are the member of class.

Terminology

Class: Graphically class represent by the rectangle in class diagram. This Rectangular diagram of class are compartmentalise by three parts. The different part of the class diagram is Name of class, attributes and operation.

Name: Name of class is the text or string .This part shows at the top of the diagram of class. Name is the first compartment.

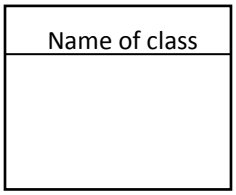


Fig 2: Name of Class Diagram

Attributes: Attributes are the data member of class. There is one or more attribute in the class .Graphically attributes are compartmentalized at below of Name compartment.

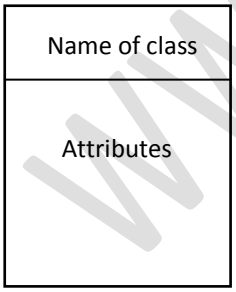
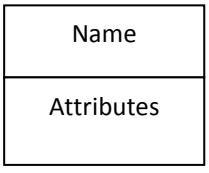


Fig 3: Name and Attribute of Class Diagram

operations: operations are the member function of class. There is one or more operation may be exist in the class .Graphically attributes are compartmentalized at below of Attribute compartment. operations shows the behaviour of the class.



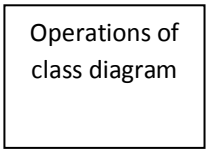


Fig 4: Name, Attribute and Operation of Class Diagram

1. Dependencies: Dependency is the one kind of relationship where one class can depend on another. Graphically dependency represent by dashed line.

2. Association: association is the relationship where one class is connected to other class. Graphically association are represent by solid line.

3. Generalization: Generalization shows the relationship between parent and child classes. So inheritance is the generalization of relationship. Graphically it represent by solid line with triangle shape arrow.

4. Aggregation: One class represents a larger thing which consists of smaller things. This kind of relation is called aggregation. Graphically it represent by solid line with open diamond.

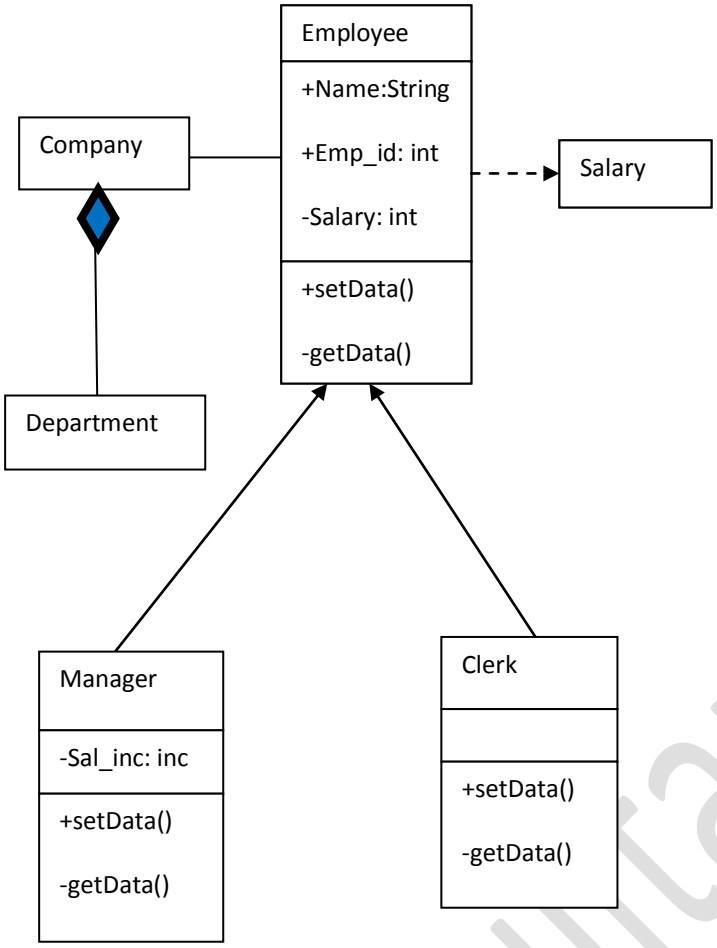


Fig 5: Different relations of class diagram

```
public    String address;
public  staticintrollno;
publicint marks;

intsetMarks(){ marks=50; }

intgetMarks()
{ return marks; }

public static intgetRoll()
{ return rollno; }
}
```

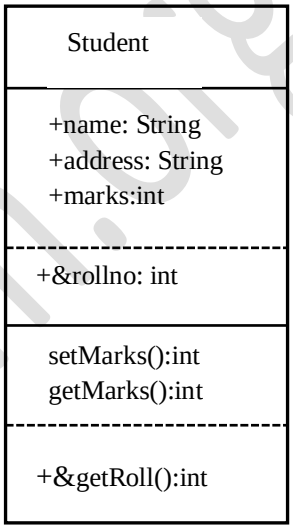


Fig 6: Extended class diagram

Proposed Idea:

Here we added some extra features and also modify existing feature in the class diagram .So we add notation for static member of class. Static member is the important member of class which are not instantiated so these member are called class variable. Here static member are graphically compartmentalized within both member of class as attribute and operations. Static member are separated by dotted line within attribute and operation part of diagram and static member represents graphical symbol &.

Example of class:

```
  Class Student
{
public    String name;
```

Visibility:

The visibility means how members of class are accessible from outside or inside of the class. There are three types of access specifies used in the class such as public, private and protected. public members of class are accessible or visible from outside of the class. private member are not accessible from outside of class ,it is only accessible within the class. protected members are accessible within immediate subclass of super class. Graphically these access specifiers are represented by different notation.

Notation are

1. public represent by +
2. private represent by –

3. protected represent by #.

These are the existing visibilities .In our proposed area we have change the way of presentation of visibilities in the class diagram. In fig we show the presentation of visibilities in different way.

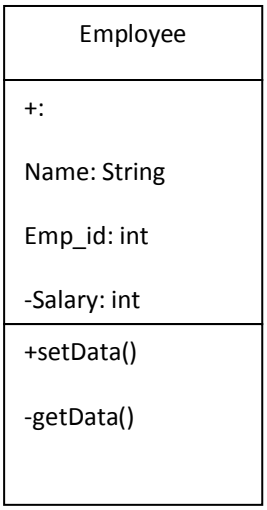


Fig 7: Existing visibilities

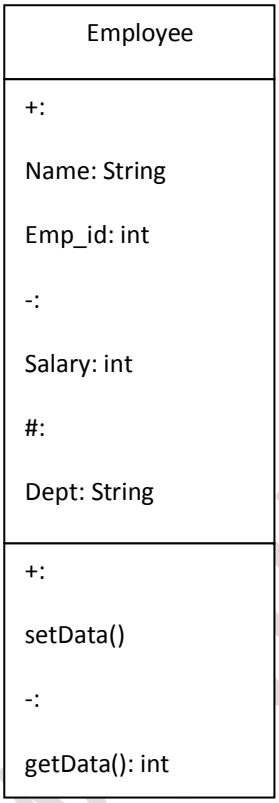


Fig 8: Modified visibilities

In this modified visibility there are separate section for public, private, protected members of class. Graphically visibilities of public member represent by +:, private member by -:, protected by #: .

Interface and Abstract class:

There is no added graphical notation for represent the abstract class and interface. within the class diagram name tag represent the abstract class and interface. So in this paper we add separate graphical notation for abstract class and interface .Here abstract class represent by round rectangle in modified diagram and interface represent by round rectangle with filled circle in name tag section.

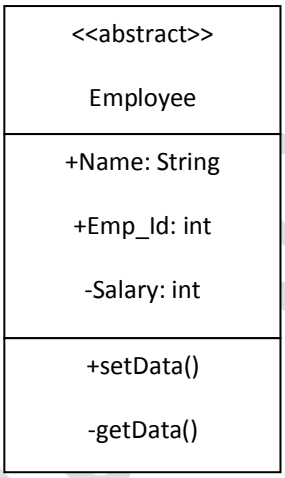


Fig 9: Existing abstract class

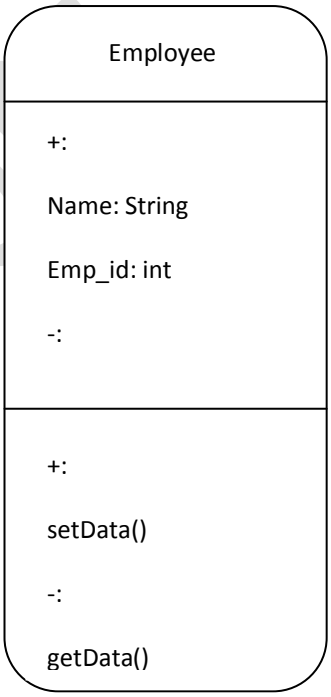


Fig 10: Modified existing class

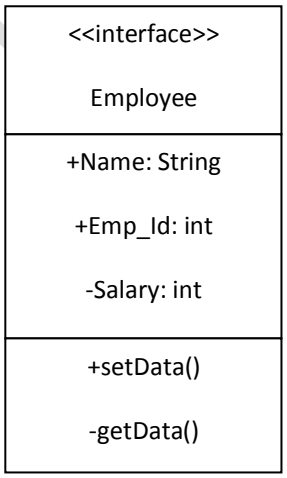


Fig 11: Existing interface

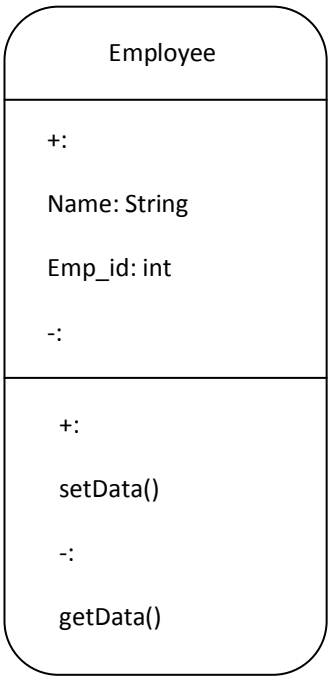


Fig 12: Modified interface

Conclusion:

In this paper we show the possible modification of class diagram. So it is possible to extend the class diagram to create better visual model for object oriented software design. We add some graphical notation and also modify some existing graphical notation to create more user friendly visual model of class diagram for object oriented software system. So this extended model of class diagram may be helpful for developer .So in future we can add more extended feature for class diagram.

Reference:

[1] The Unified Modeling Language User Guide,GradyBooch, James Rumbough,IvarJacobson,Pearson Education .

[2] Software Engineering ,Rajib Mall, Third Edition,PHI Learning Pvt Ltd.