UML

By Somenath Mukhopadhyay

som@som-itsolutions.com

What is the UML?

- Stands for unified modelling language
- Is the successor of OOAD methods
- It unifies the methods of Booch, Rumbaugh and Jacobson
- Now a standard with Object Management Group (OMG)

Why Should I use UML?

- When software development means producing codes, why will I produce pretty diagrams?

The Answers are here

- Communication - it allows the developers to communicate clearly
- Natural language gets tangled when it comes to more complex concepts
- Code is precise but too detailed
- So I use UML to get some amount of precision but to avoid too much detailing

Contd..

The Answers are here

- Without going through the nitty gritty of the code, I can look at the class diagram and the sequence diagram to get an overall view of the system
- Object languages allow advantages but don't provide them. To use these advantages, we have to make the infamous paradigm shift
- UML is the tool which can help us in making our learning curve a bit smooth
- Through UML, we can link a piece of OOAD code to see whether it resembles to any design pattern

Use Case – What is it?

- A Use case is a description of a real life scenario
- Buy a Product Scenario –

The customer browses the catalog and adds desired items to the shopping basket. When the customer wishes to pay, the customer describes the shipping and credit card information and confirms the sale. The system checks the authorization on the credit card and confirms the sale both immediately and with a follow-up email.

Contd…

Use Case – What is It?

Buy a Product

- Customer browses through catalog and selects items to buy
- Customer goes to check out
- Customer fills in shipping information
- System presents full pricing information, including shipping
- Customer fills in credit card information
- System authorizes purchase
- System confirms sale immediately
- System sends confirmation email to customer

Contd…

Use Case – What is It?

- Exceptions

– At step 6, system fails to authorize credit purchase. Allow customer to re-enter credit card info and re-try

Use case – What is it?

Class Diagrams

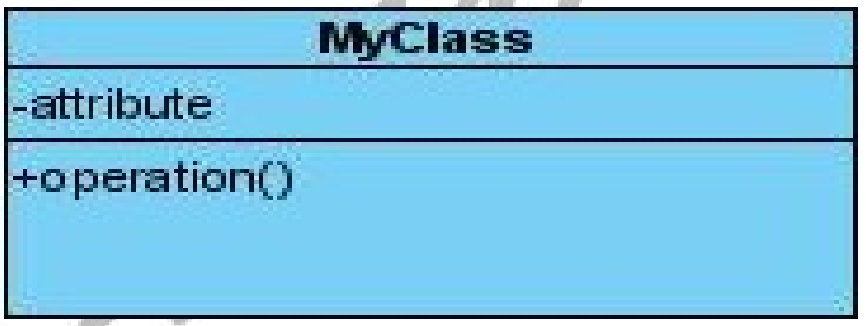
- Describes the static relationships among the different types of objects in a system

- Different characteristics of a class

- Attributes
- Operations
- Relationships with other classes

Class Diagrams

- Class Icon



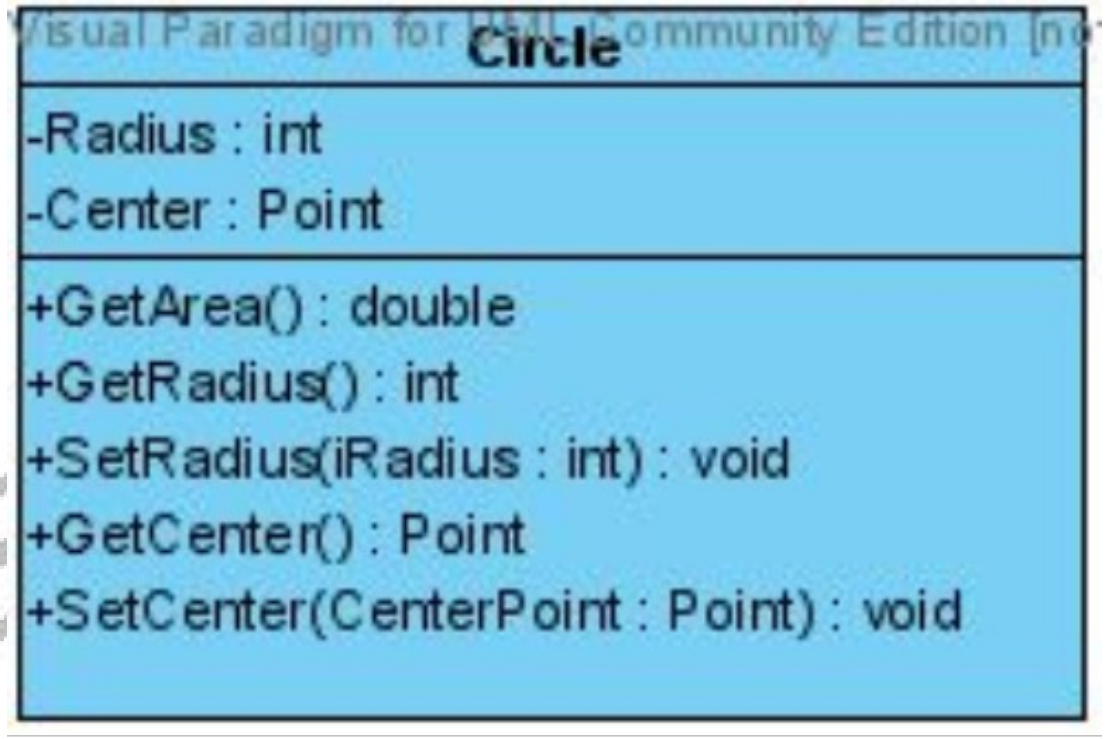
Class Diagrams

Visibility of the members of a class:

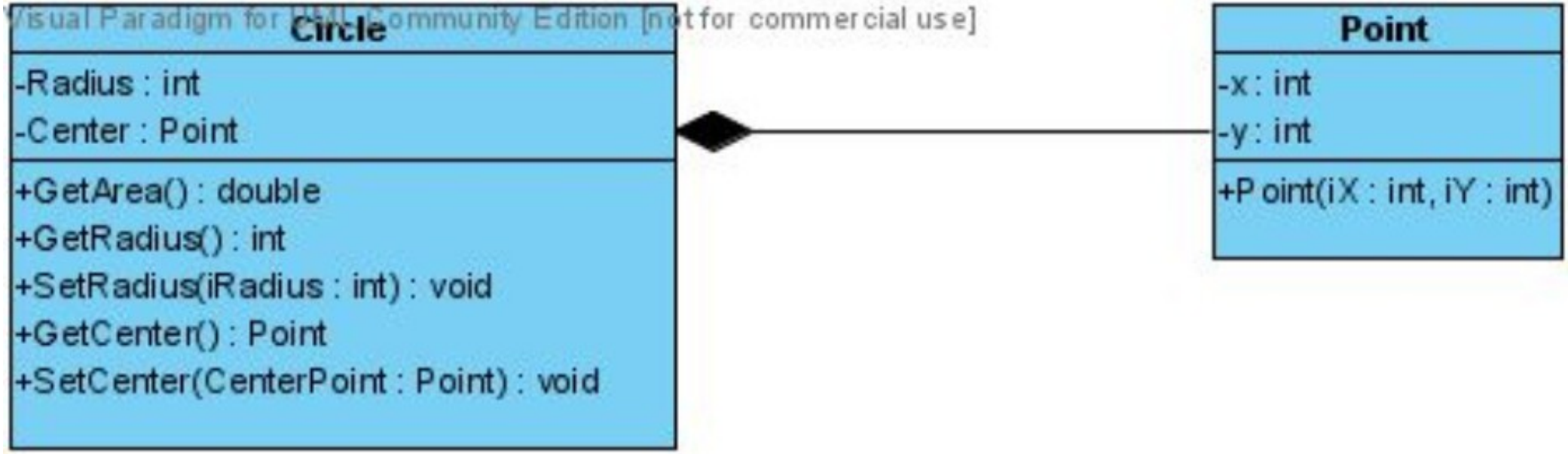
- + public
- # protected
- - private

Class Icon

- Example - Circle Class



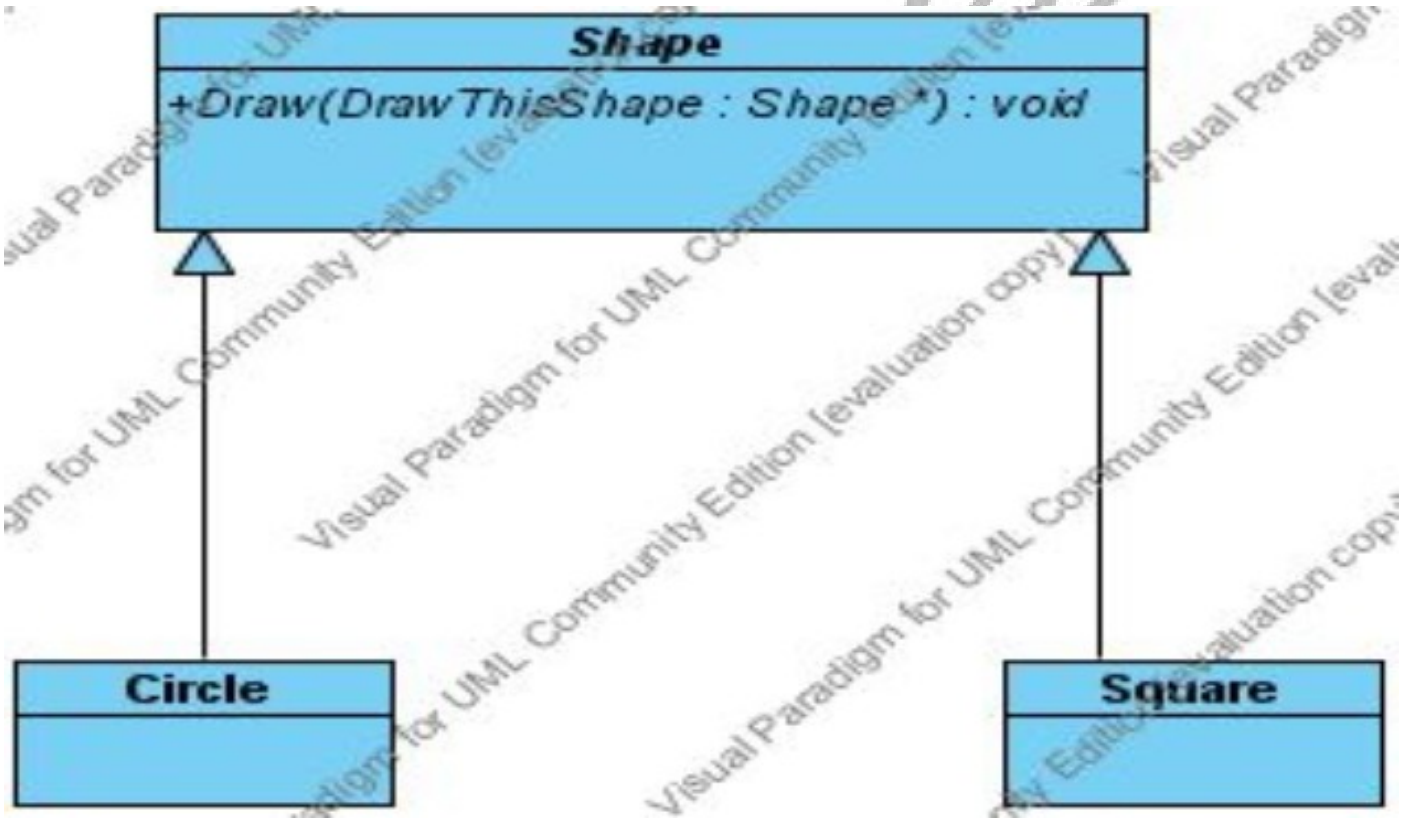
Relationship Between Classes - Composition



Relationship between Classes - Composition

- Circle has Point
- It's an Has-a relationship
- It is depicted by the black diamond

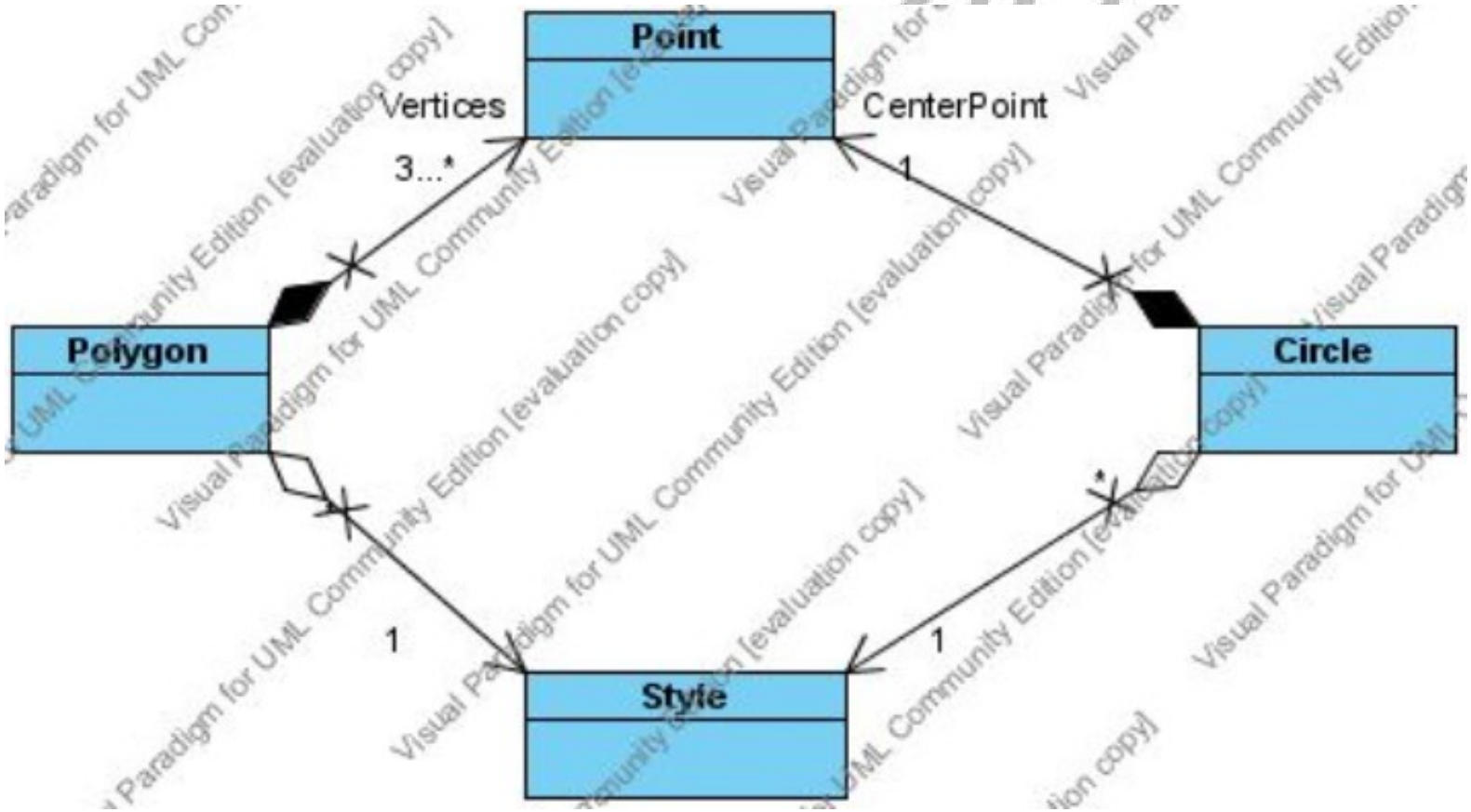
Relationships between classes - Generalization/Inheritance



Relationships between classes - Generalization/Inheritance

- Base class Shape
- It has got a virtual function Draw which will be overridden in the derived classes
- The italic tells us that it is a virtual function
- Two inherited classes – Circle and Square
- Circle and Square classes will inherit Shape's Characteristics. They may have their own characteristics

Relationships between classes – Composition and Aggregation



Relationships between classes

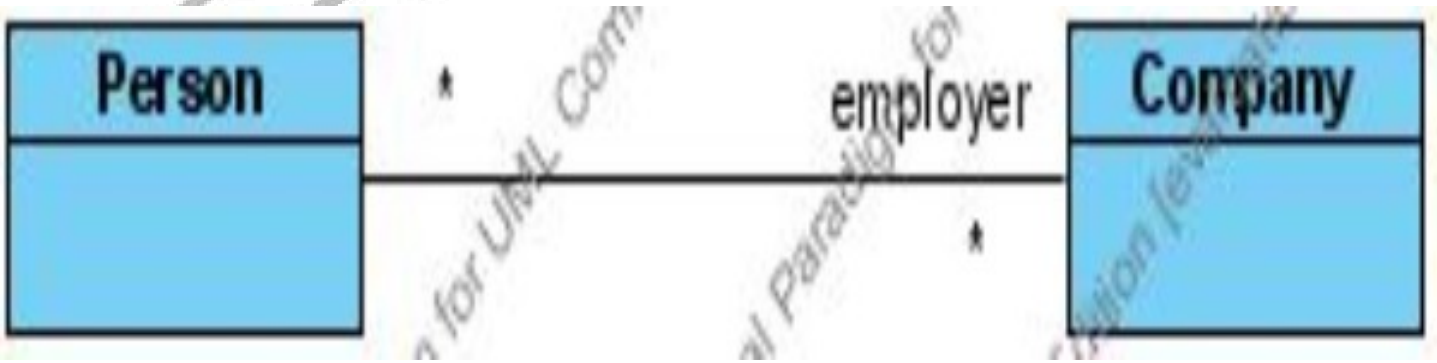
- There are differences between Composition and Aggregation
- Aggregation by value is composition. It is represented by the black diamond
- In Composition the life time of the composed object is same as the container
- Aggregation may be by pointer or by reference. It is represented by the white diamond
- In aggregation the life time of the part is not dependent on the whole

Relationships between classes

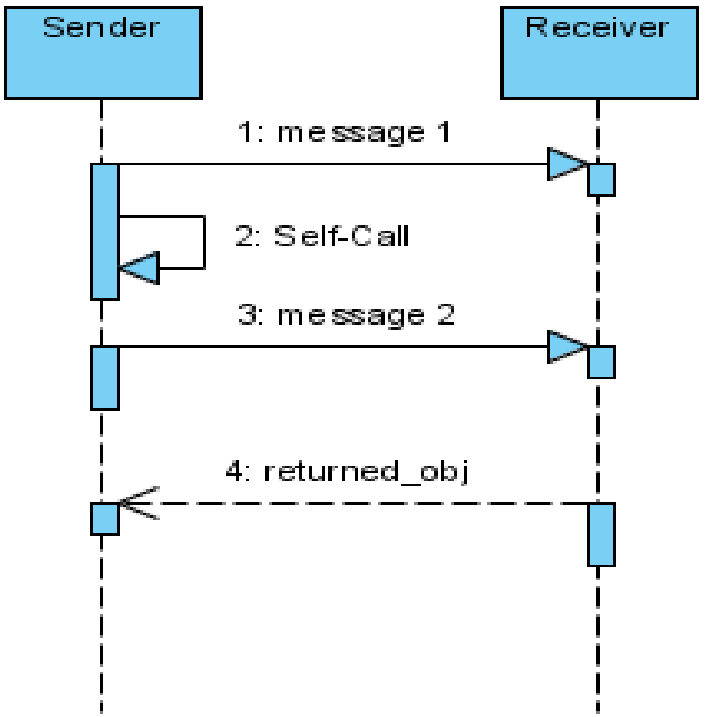
- Navigability- the direction of message flow
- The message flows from Circle to Point but not vice versa
- X indicates non-navigable

Relationships between classes - Association

- In case of association relationship, the objects of the associated class are passed as parameters in messages.



Sequence Diagram



Sequence Diagram

Vertical lines represent Objects – not classes

The arrow represents messages

The dashed line represents return