PareDox
Education Solutions

Aamir Shabbir Pare

Problem Solving Programming

Design Patters

GRASP Design Principles

GRASP

- Stands for General Responsibility Assignment Software Patterns
- Guides in assigning responsibilities to collaborating objects.
- 9 GRASP patterns
  - Creator
  - Information Expert
  - Low Coupling
  - Controller
  - High Cohesion
  - Indirection
  - Polymorphism
  - Protected Variations
  - Pure Fabrication

Aamir Shabbir Pare

PareDox
Education Solutions

Responsibility

- Responsibility can be:
  - Accomplished by a single object
  - Or Collaboratively by a group of objects
- GRASP helps us in deciding which responsibility should be assigned to which object/class.
- Identify the objects and responsibilities from the problem domain, and also identify how objects interact with each other.
- Define blueprint for those objects – i.e. class with methods implementing those responsibilities.

PareDox
Education Solutions

Creator

- Who creates an Object? Or who should create a new instance of some class?
- “Container” object creates “contained” objects.
- Decide who can be creator based on the objects association and their interaction.

Aamir Shabbir Pare

PareDox
Education Solutions

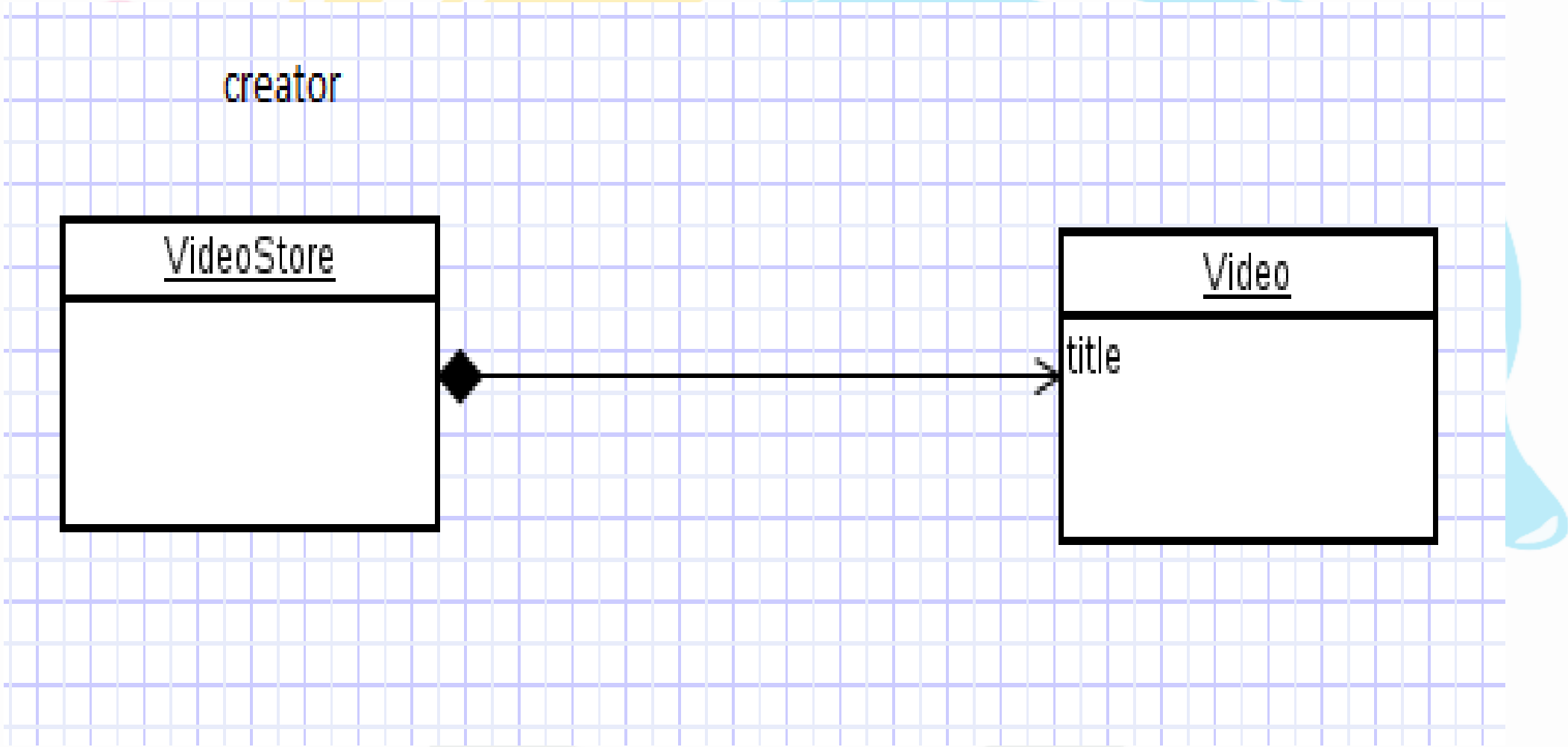
Creator Example

- Consider VideoStore and Video in that store.
- VideoStore has an aggregation association with Video. I.e, VideoStore is the container and the Video is the contained object.
- So, we can instantiate video object in VideoStore class

Aamir Shabbir Pare

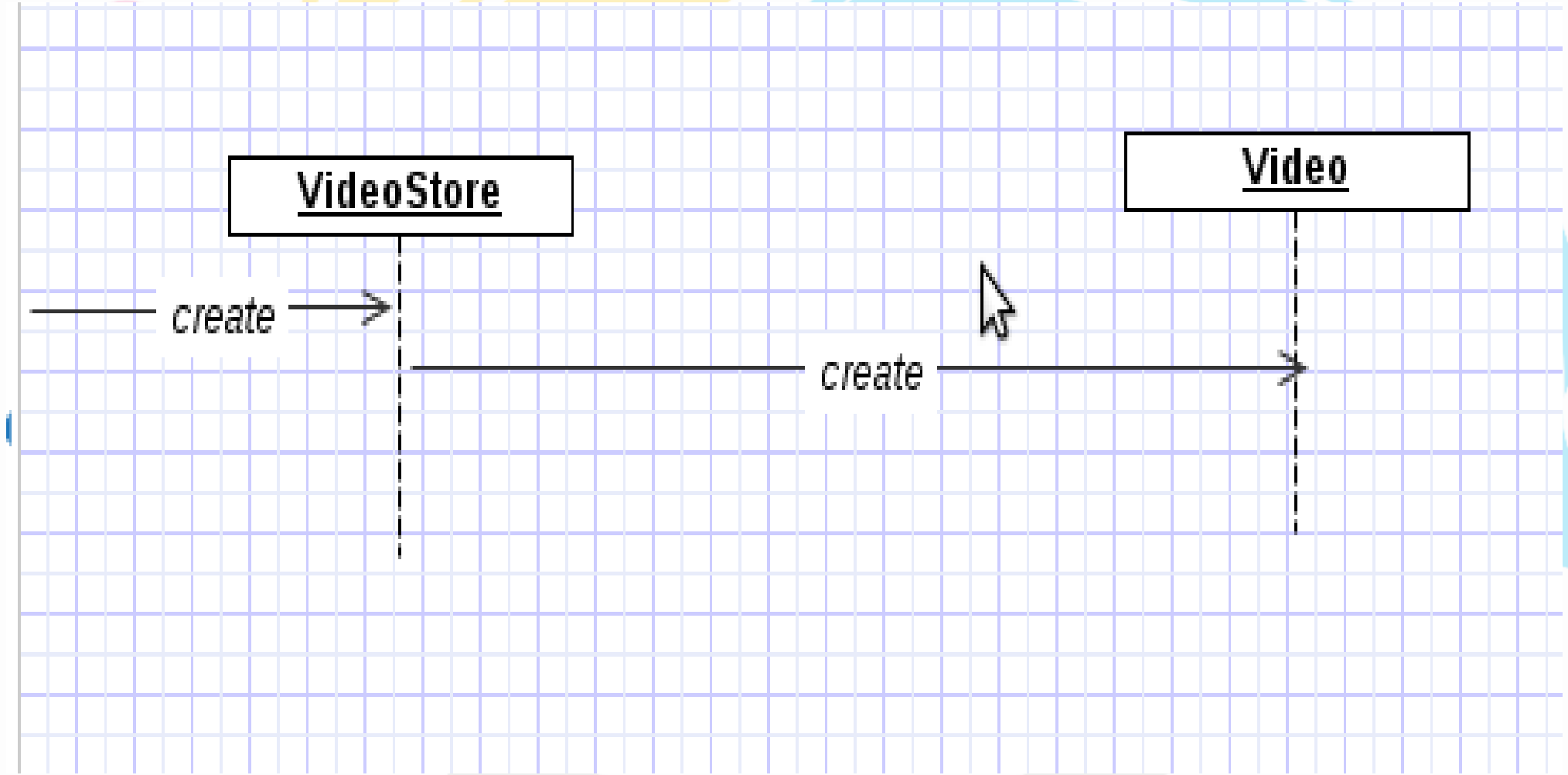
PareDox
Education Solutions

Example Diagram



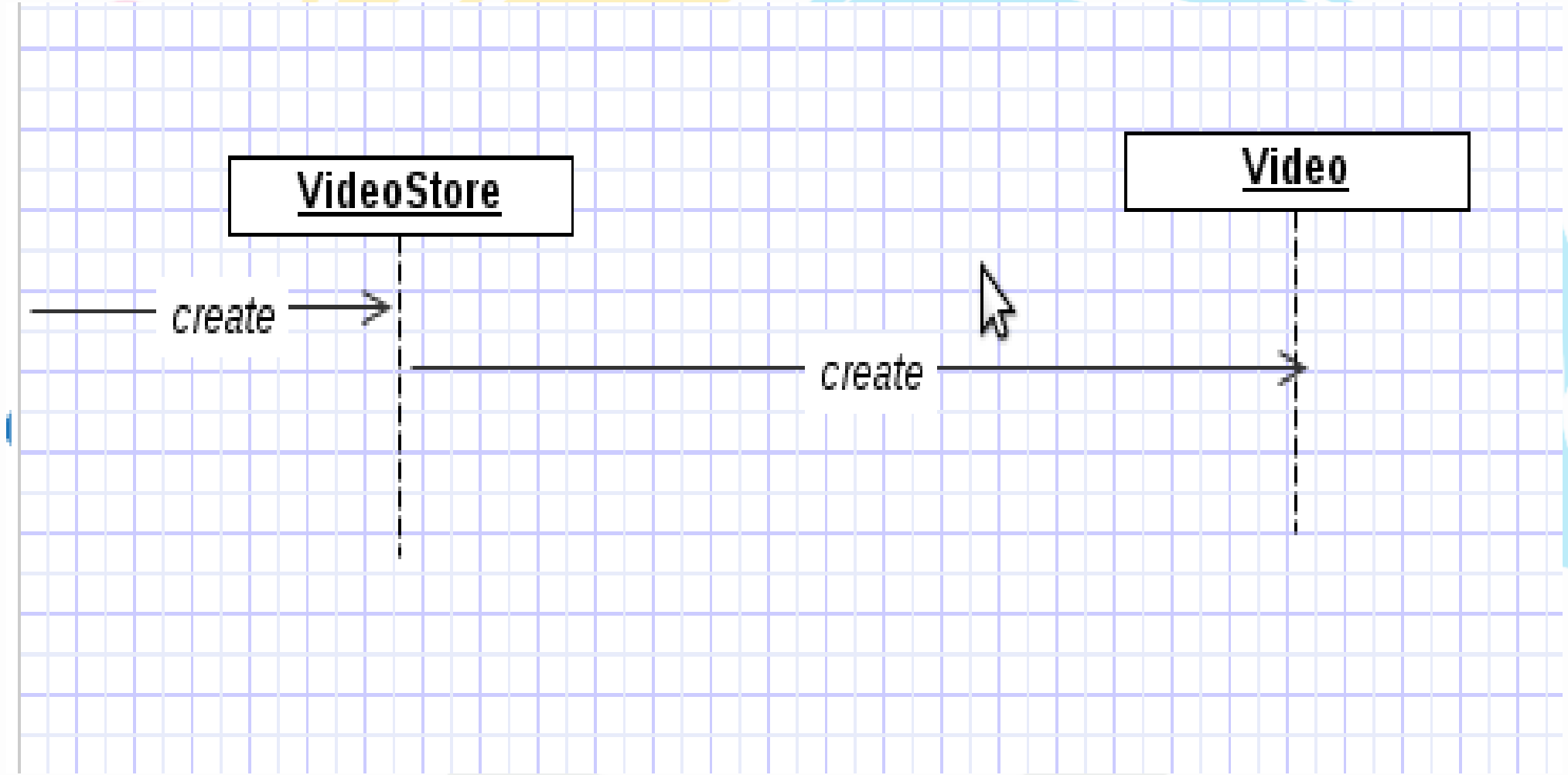
Aamir Shabbir Pare

Creator Example



Aamir Shabbir Pare

Creator Example



Aamir Shabbir Pare

PareDox
Education Solutions

Information Expert

- Given an object o, which responsibilities can be assigned to o?
- Expert principle says – assign those responsibilities to o for which o has the information to fulfill that responsibility.
- They have all the information needed to perform operations, or in some cases they collaborate with others to fulfill their responsibilities.

Aamir Shabbir Pare

PareDox
Education Solutions

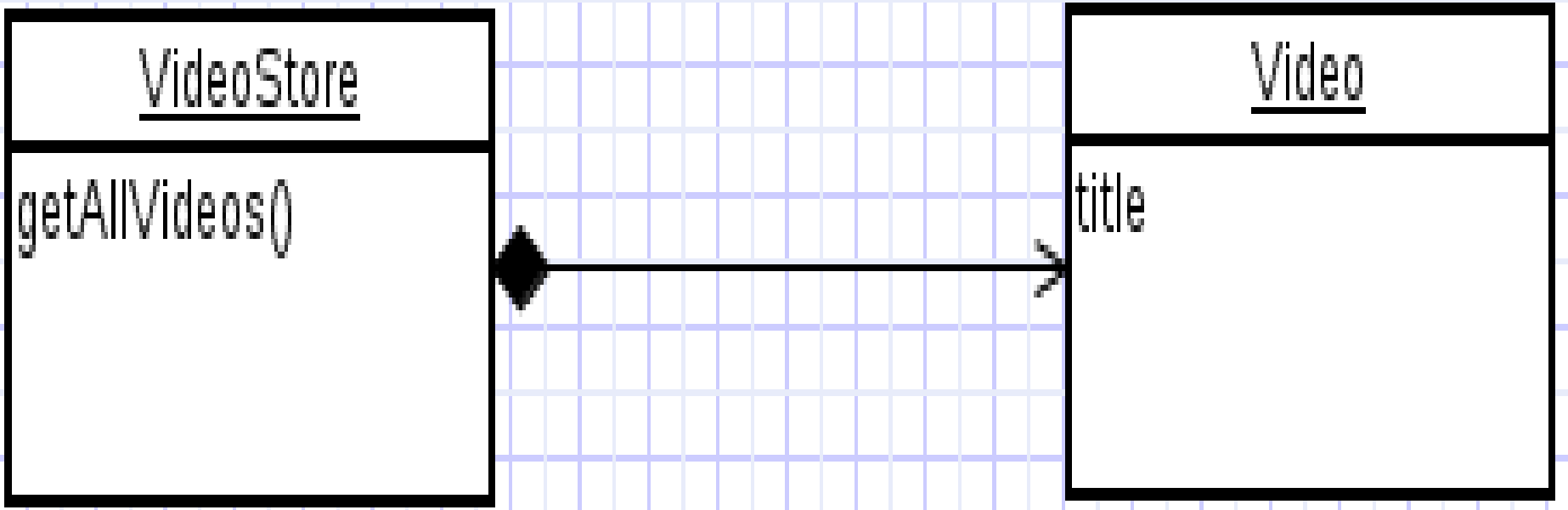
Example Information Expert

- Assume we need to get all the videos of a VideoStore.
- Since VideoStore knows about all the videos, we can assign this responsibility of giving all the videos can be assigned to VideoStore class.
- VideoStore is the information expert.

Aamir Shabbir Pare

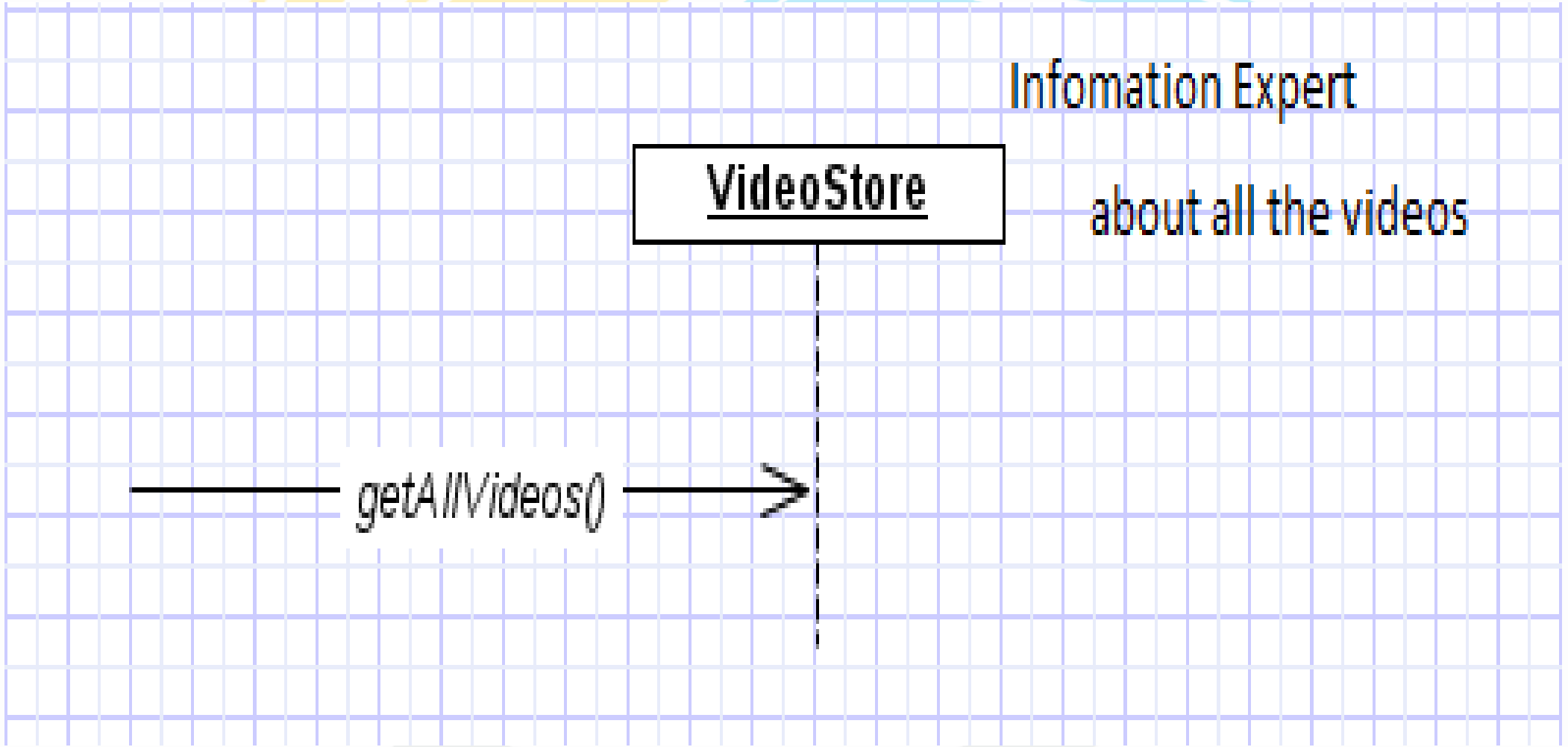
PareDox
Education Solutions

Example Expert



Aamir Shabbir Pare

Example Expert



Aamir Shabbir Pare

PareDox
Education Solutions

Low Coupling

- How strongly the objects are connected to each other?
- Coupling – object depending on other object.
- When depended upon element changes, it affects the dependent also.
- Low Coupling – How can we reduce the impact of change in depended upon elements on dependent elements.
- Prefer low coupling – assign responsibilities so that coupling remain low.
- Minimizes the dependency hence making system maintainable, efficient and code reusable

Aamir Shabbir Pare

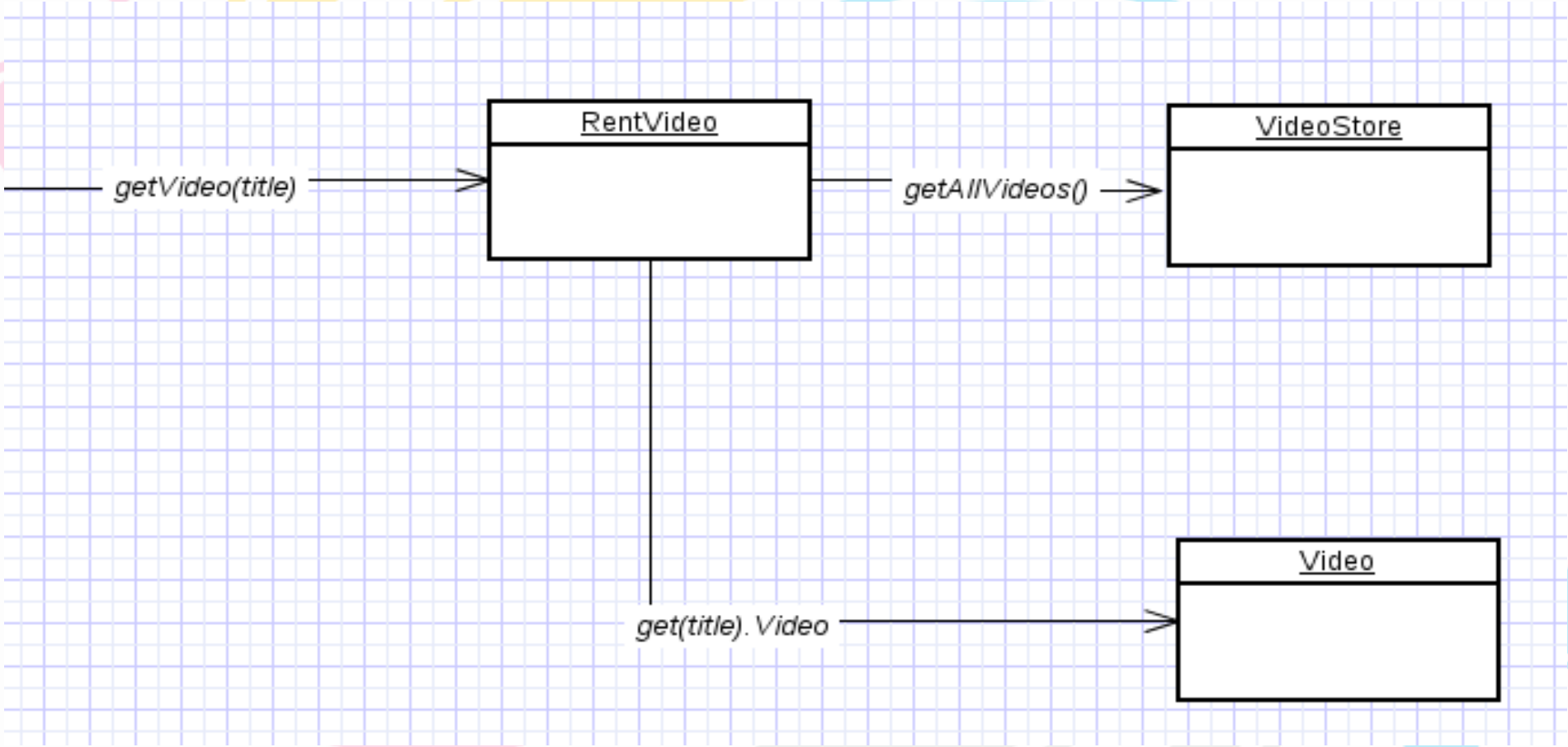
PareDox
Education Solutions

Low Coupling

- Two elements are coupled, if
  - One element has aggregation/composition association with another element.
  - One element implements/extends other element.

Aamir Shabbir Pare

Poor Coupling Example



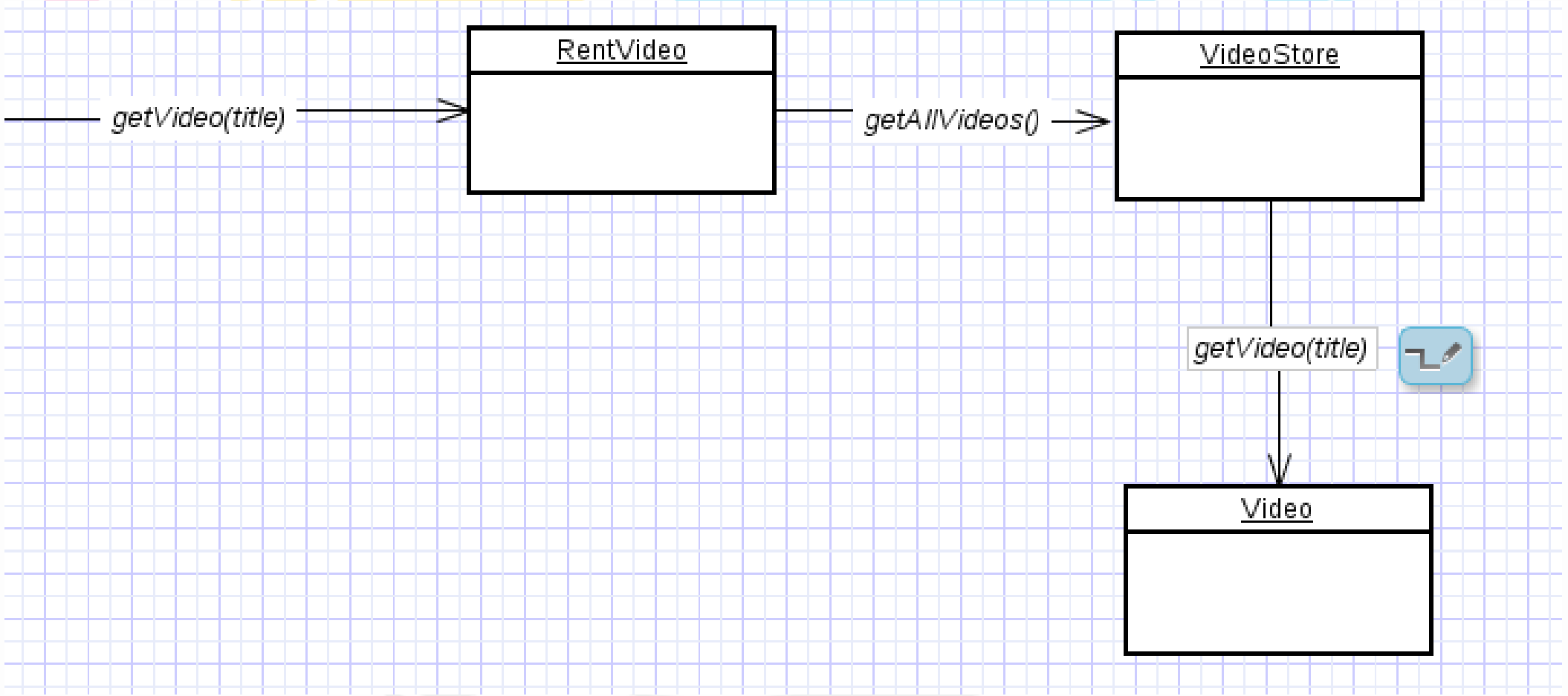
- Here class Rent knows about both VideoStore and Video objects. Rent is depending on both the classes.

Aamir Shabbir Pare

PareDox
Education Solutions

Low Coupling Example

VideoStore and Video class are coupled, and Rent is coupled with VideoStore. Thus providing low coupling.



PareDox
Education Solutions

Controller

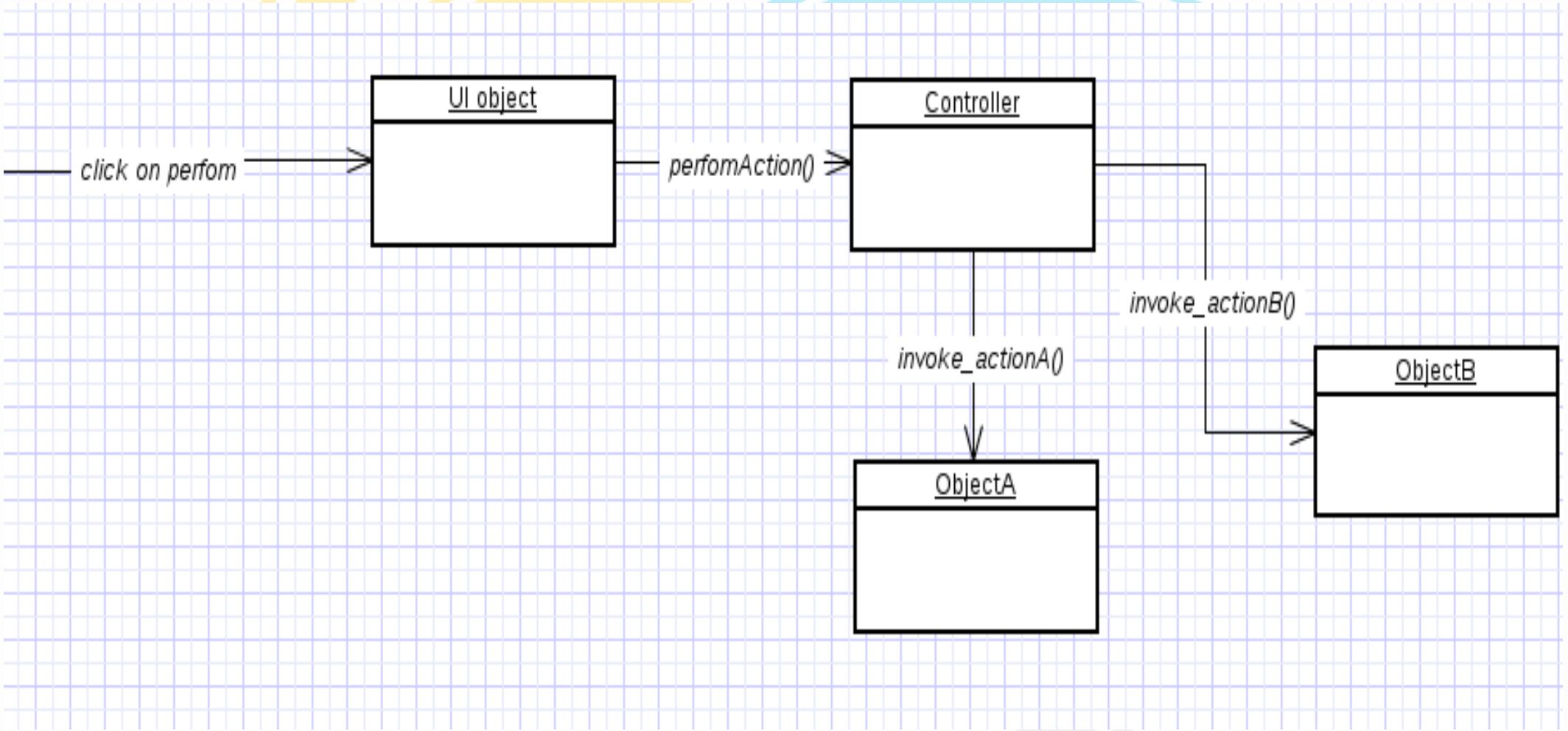
- Deals with how to delegate the request from the UI layer objects to domain layer objects.
- when a request comes from UI layer object, Controller pattern helps us in determining what is that first object that receive the message from the UI layer objects.
- This object is called controller object which receives request from UI layer object and then controls/coordinates with other object of the domain layer to fulfill the request.
- It delegates the work to other class and coordinates the overall activity.

Controller

- **We can make an object as Controller, if**
  - Object represents the overall system (facade controller)
  - Object represent a use case, handling a sequence of operations (session controller).
- **Benefits**
  - can reuse this controller class.
  - Can use to maintain the state of the use case.
  - Can control the sequence of the activities

PareDox
Education Solutions

Controller Example



Aamir Shabbir Pare

Bloated Controllers

Controller class is called bloated, if

- The class is overloaded with too many responsibilities.

  Solution – Add more controllers

- Controller class also performing many tasks instead of delegating to other class.

  Solution – controller class has to delegate things to others.

PareDox
Education Solutions

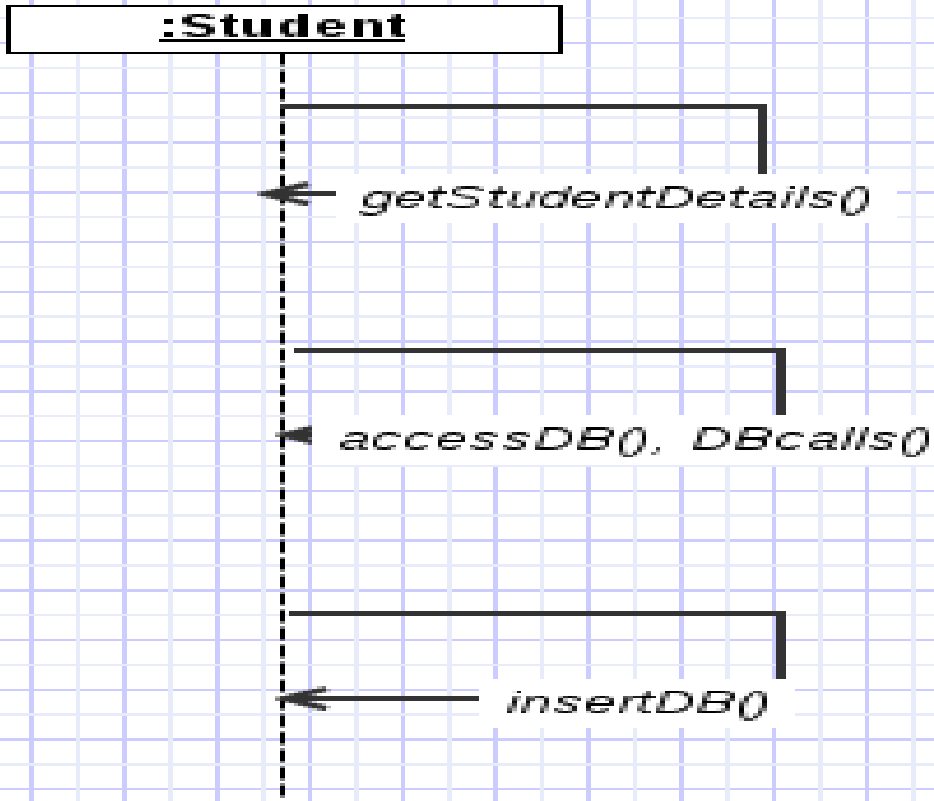
High Cohesion

- How are the operations of any element are functionally related?
- Related responsibilities in to one manageable unit.
- Prefer high cohesion
- Clearly defines the purpose of the element
- Benefits
  - Easily understandable and maintainable.
  - Code reuse
  - Low coupling

Aamir Shabbir Pare

PareDox
Education Solutions

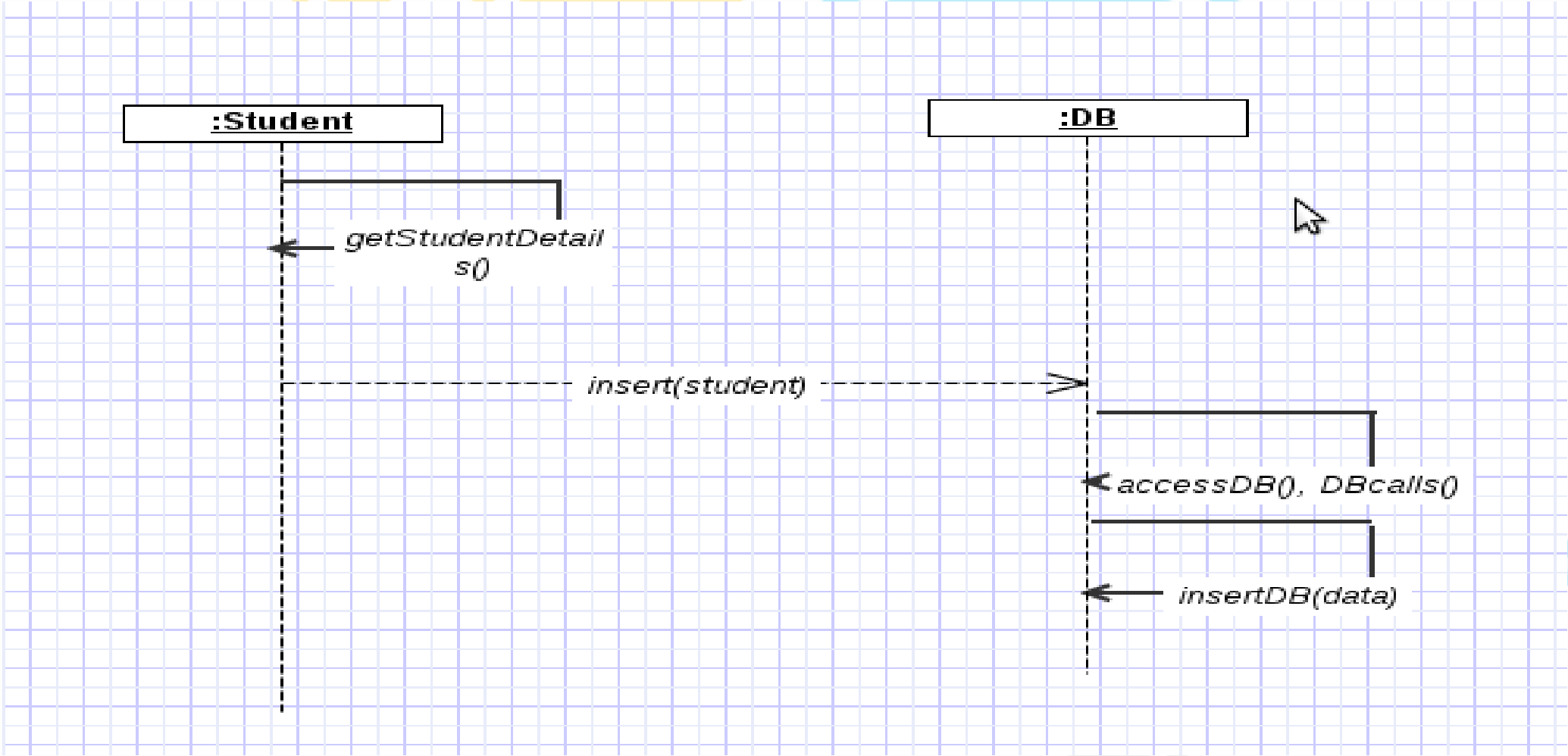
Low Cohesion Example



Aamir Shabbir Pare

PareDox
Education Solutions

High Cohesion Example



Aamir Shabbir Pare

PareDox
Education Solutions

Polymorphism

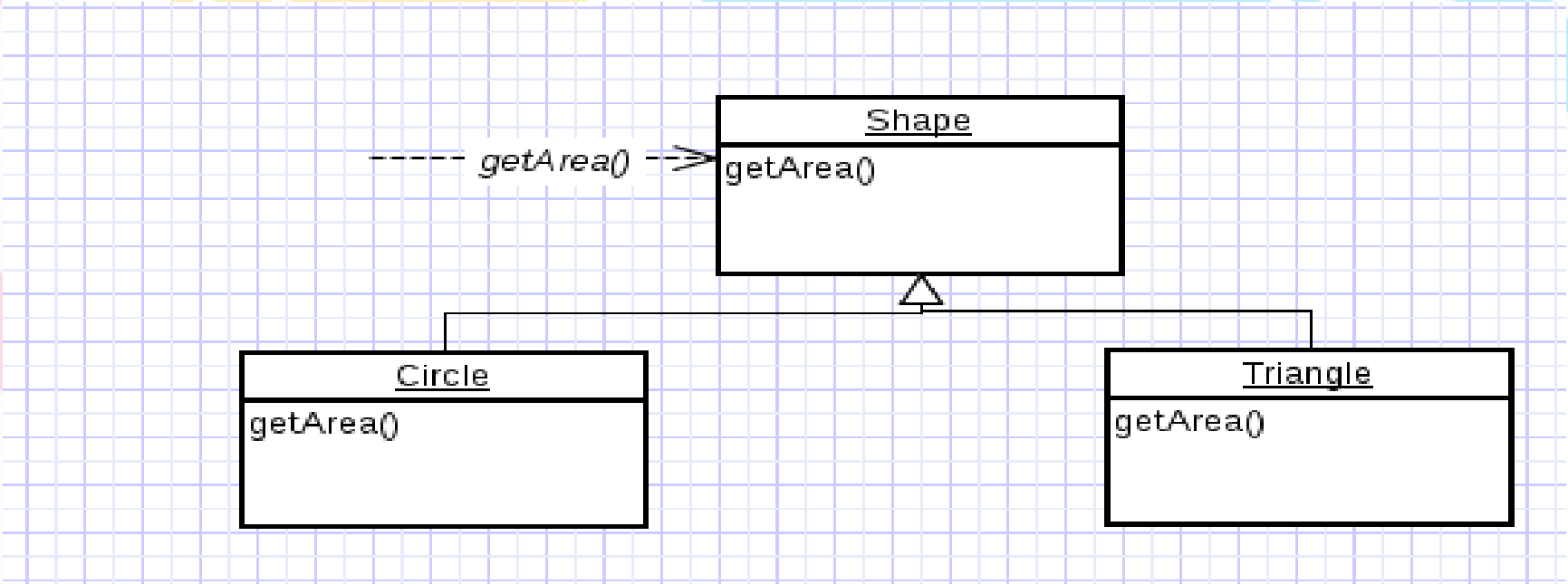
- How to handle related but varying elements based on element type?
- Polymorphism guides us in deciding which object is responsible for handling those varying elements.
- Benefits: handling new variations will become easy.

Aamir Shabbir Pare

PareDox
Education Solutions

Polymorphism Example

- The getArea() varies by the type of shape, so we assign that responsibility to the subclasses.



- By sending message to the Shape object, a call will be made to the corresponding sub class object – Circle or Triangle.

Aamir Shabbir Pare

PareDox
Education Solutions

Pure Fabrication

- Fabricated class/ artificial class – assign set of related responsibilities that doesn't represent any domain object.
- Provides a highly cohesive set of activities.
- Behavioral decomposed – implements some algorithm.
- Examples: Adapter, Strategy
- Benefits: High cohesion, low coupling and can reuse this class.

Aamir Shabbir Pare

PareDox
Education Solutions

Pure Fabrication Example

- Suppose the Shape class, if we must store the shape data in a database.
- If we put this responsibility in Shape class, there will be many database related operations thus making Shape incohesive.
- So, create a fabricated class DBStore which is responsible to perform all database operations.
- Similarly logInterface which is responsible for logging information is also a good example for Pure Fabrication.

Aamir Shabbir Pare

PareDox
Education Solutions

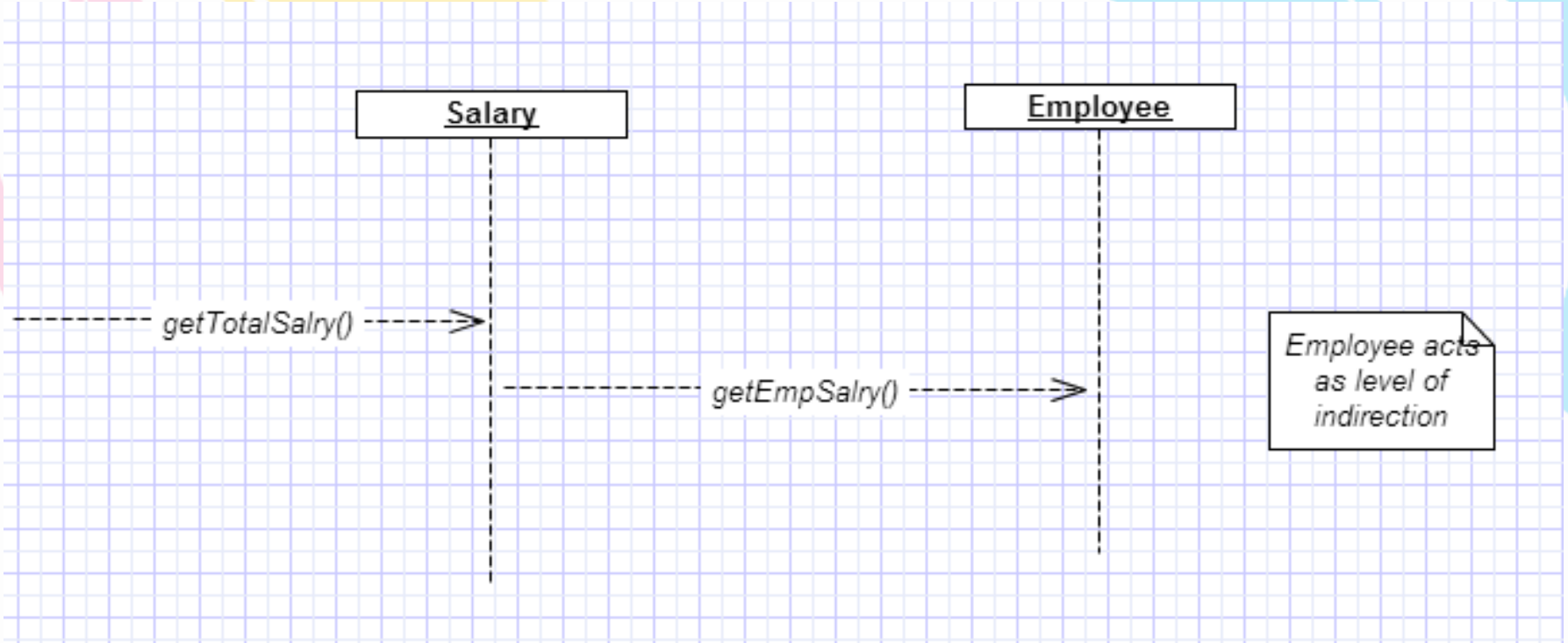
Indirection

- How can we avoid a direct coupling between two or more elements.
- Indirection introduces an intermediate unit to communicate between the other units, so that the other units are not directly coupled.
- Benefits: low coupling
- Example: Adapter, Facade, Obserever

Aamir Shabbir Pare

Indirection Example

- Here polymorphism illustrates indirection
- Class Employee provides a level of indirection to other units of the system.



Aamir Shabbir Pare

PareDox
Education Solutions

Protected Variations

- How to avoid impact of variations of some elements on the other elements.
- It provides a well defined interface so that the there will be no affect on other units.
- Provides flexibility and protection from variations.
- Provides more structured design.
- Example: polymorphism, data encapsulation, interfaces

Aamir Shabbir Pare

PareDox
Education Solutions

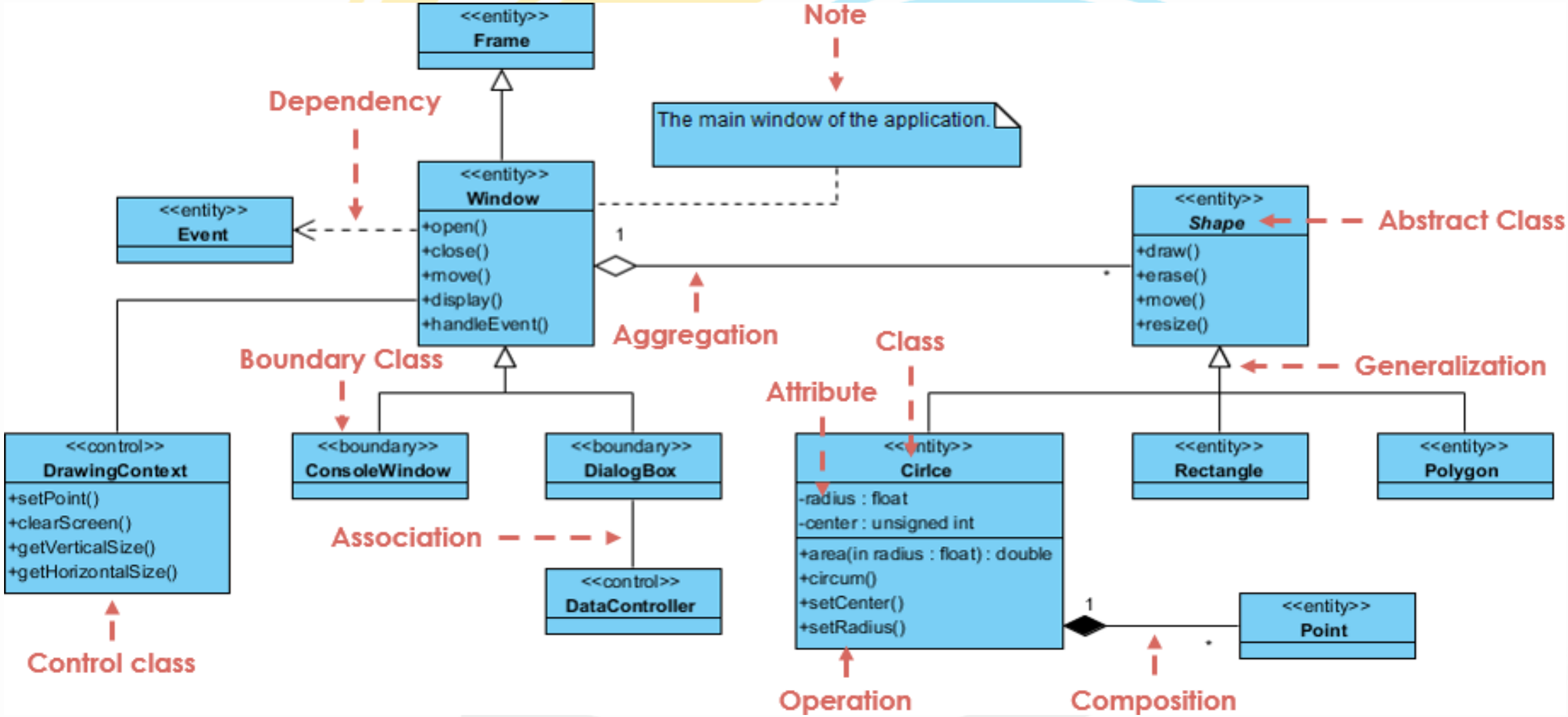
Reference

Applying UML and Patterns, Third Edition, Craig Larman

Aamir Shabbir Pare

PareDox
Education Solutions

Example of UML Class Diagram



Aamir Shabbir Pare