SHRI RAMDEOBABA COLLEGE OF
ENGINEERING AND MANAGEMENT,
NAGPUR - 440013

DESIGN PATTERNS

V SEMESTER

COURSE COORDINATOR: RINA DAMDOO

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BRIDGE DESIGN PATTERNS

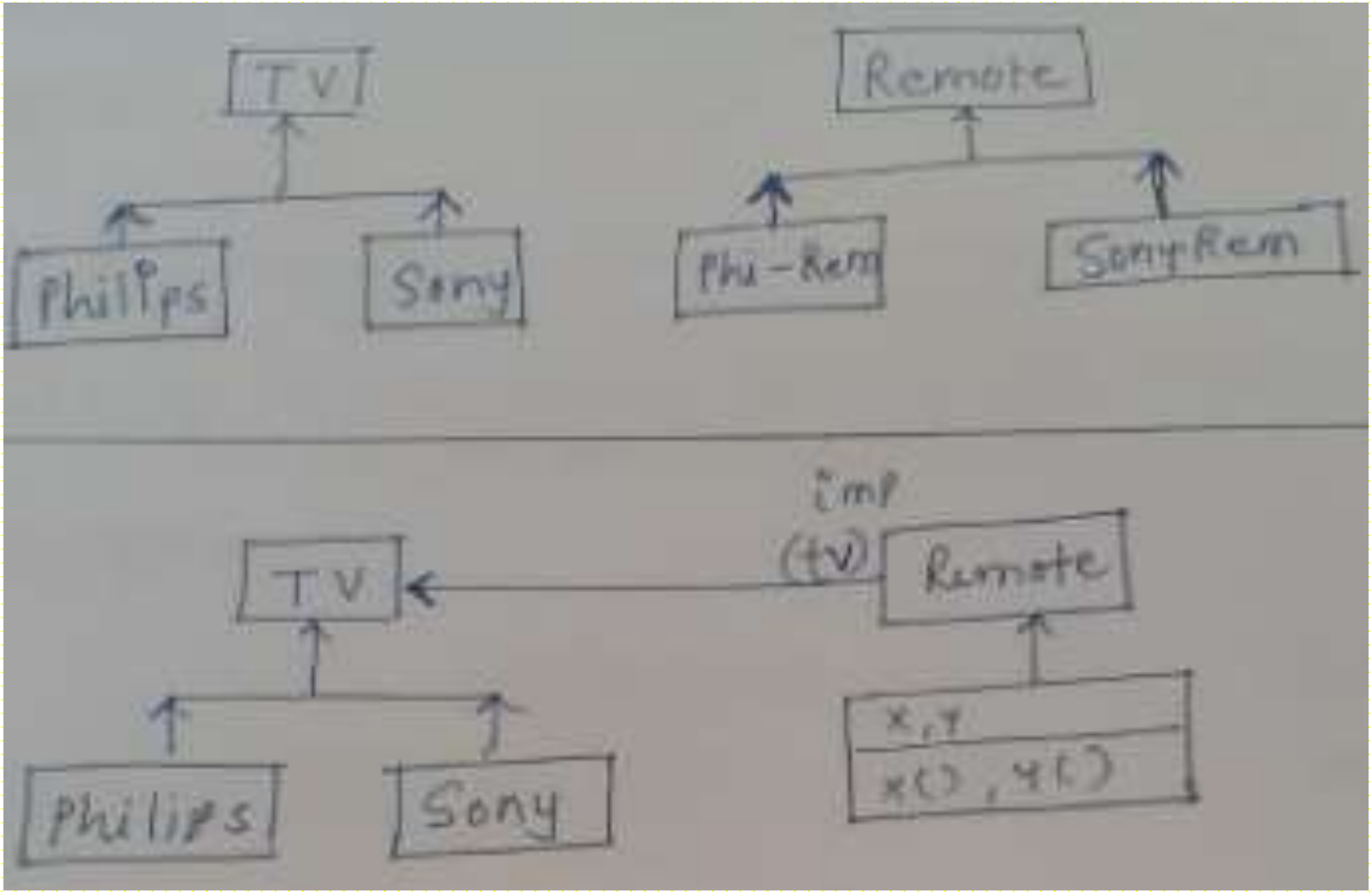
- **Intent**

  Decouple an abstraction from its implementation so that the two can vary independently.

- **Also known As**

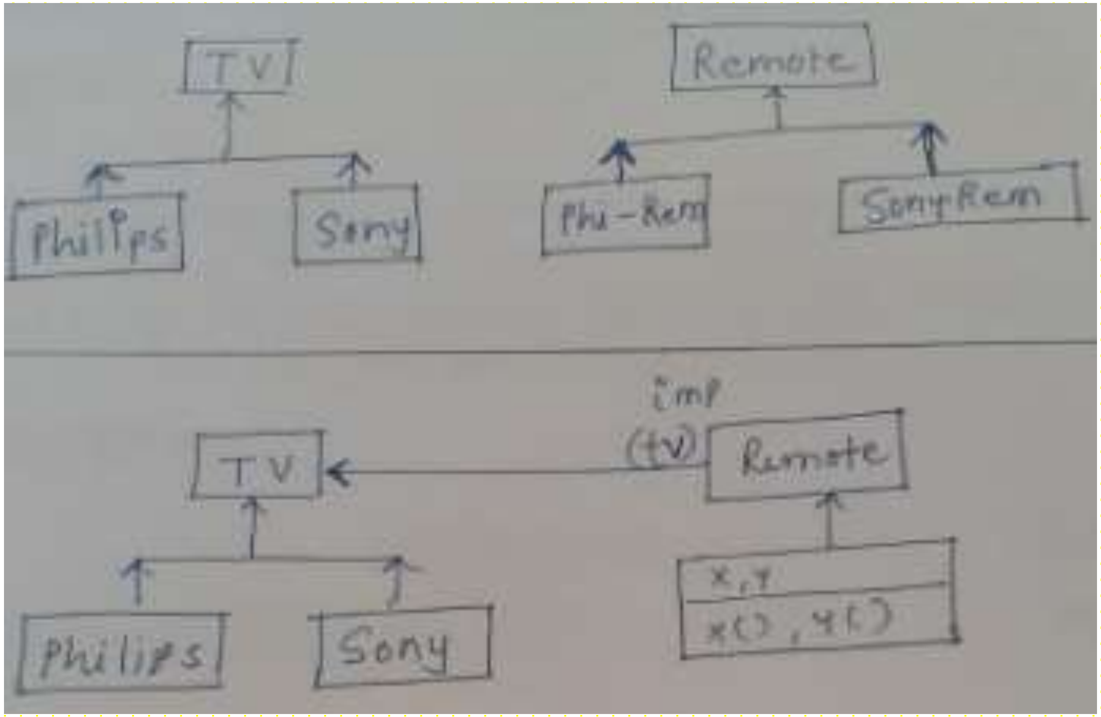
  Handle / Body

Bridge Example

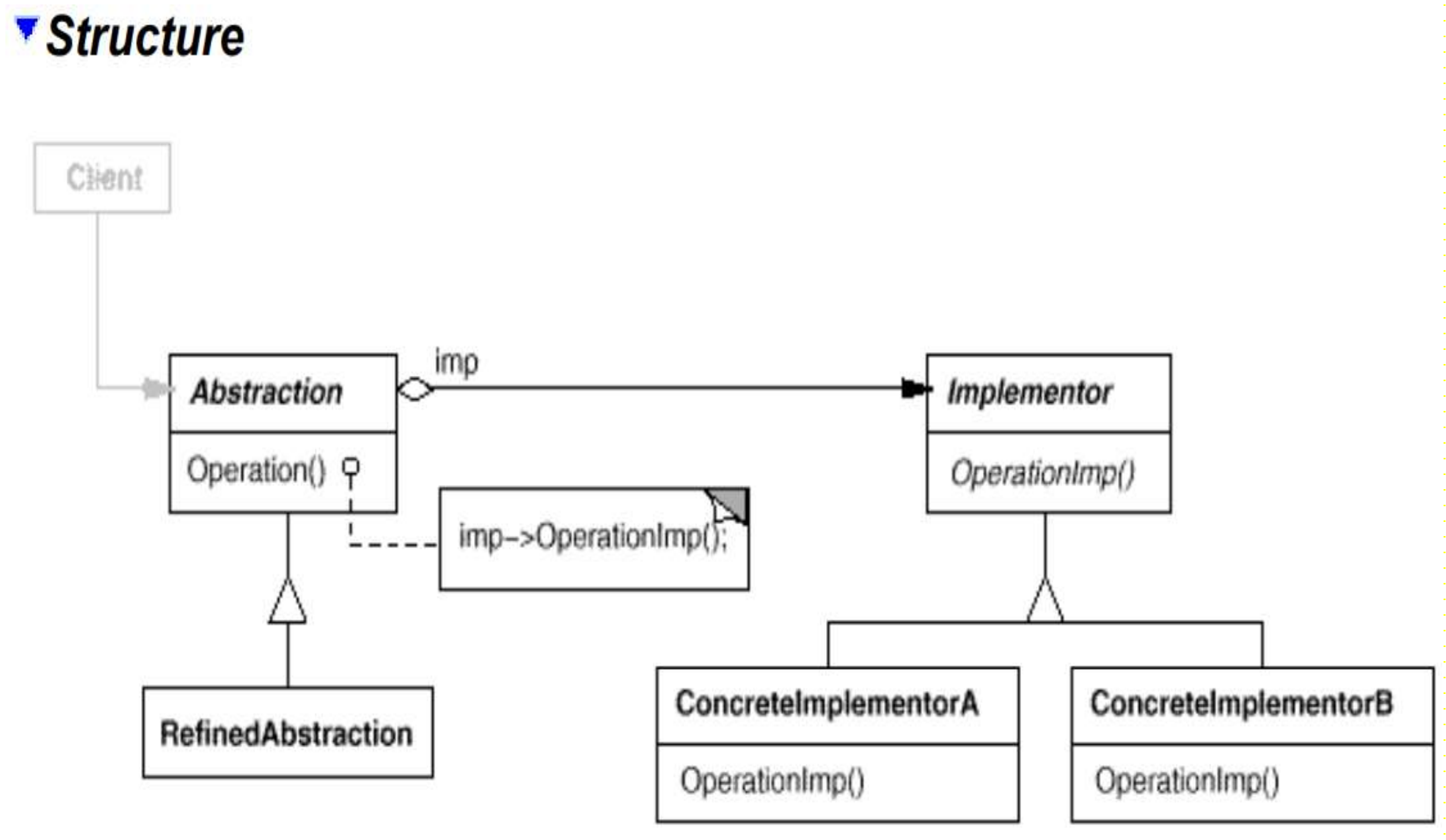


BRIDGE DESIGN PATTERNS

Problem Statement:
Create a common remote for two TV companies. Two TV companies have totally different platforms.

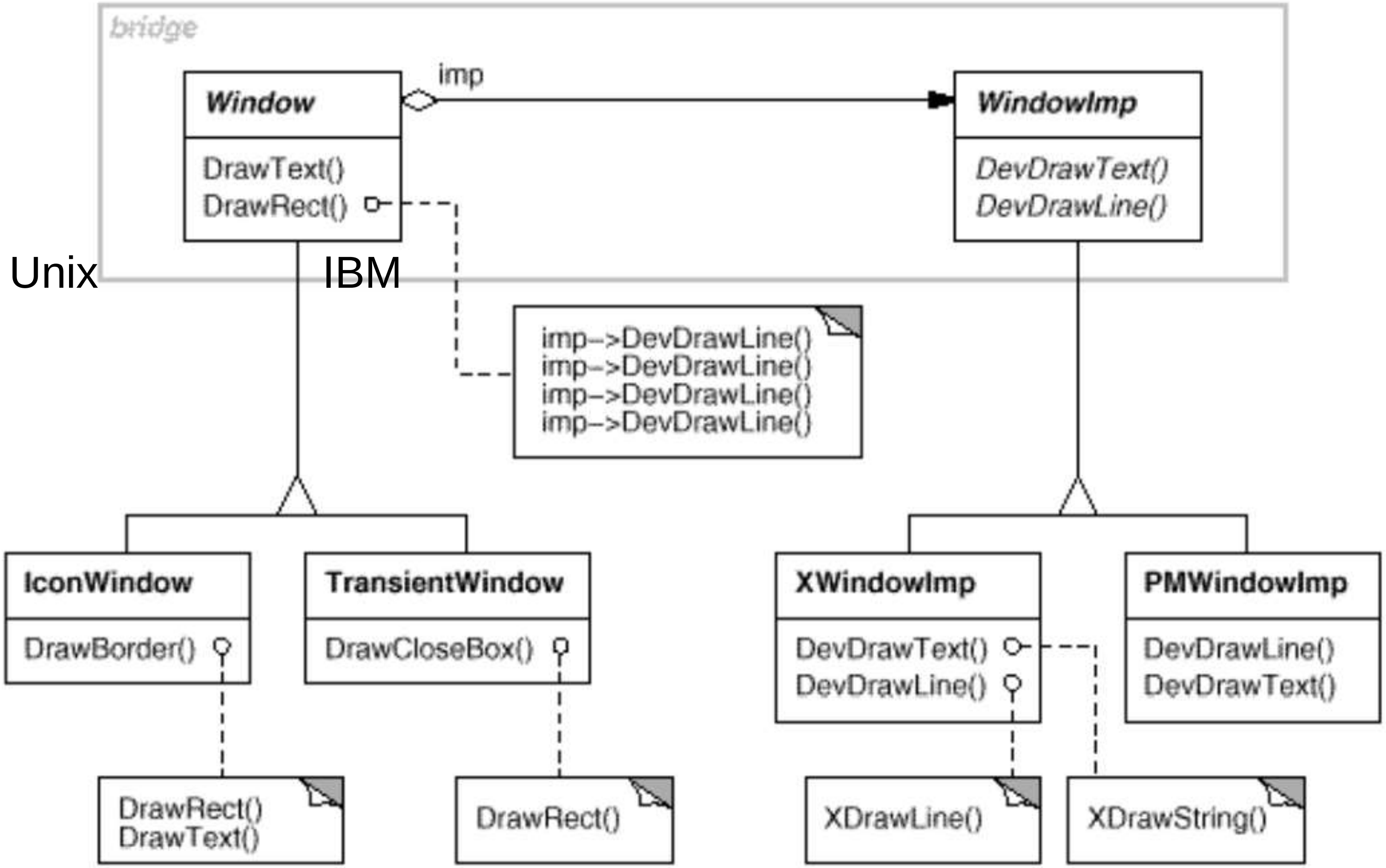


Bridge Design Patterns



Structure

MOTIVATION



We refer to the relationship between Window and WindowImp as a bridge, because it bridges the abstraction and its implementation, letting them vary independently.

APPLICABILITY

Use the Bridge pattern when:

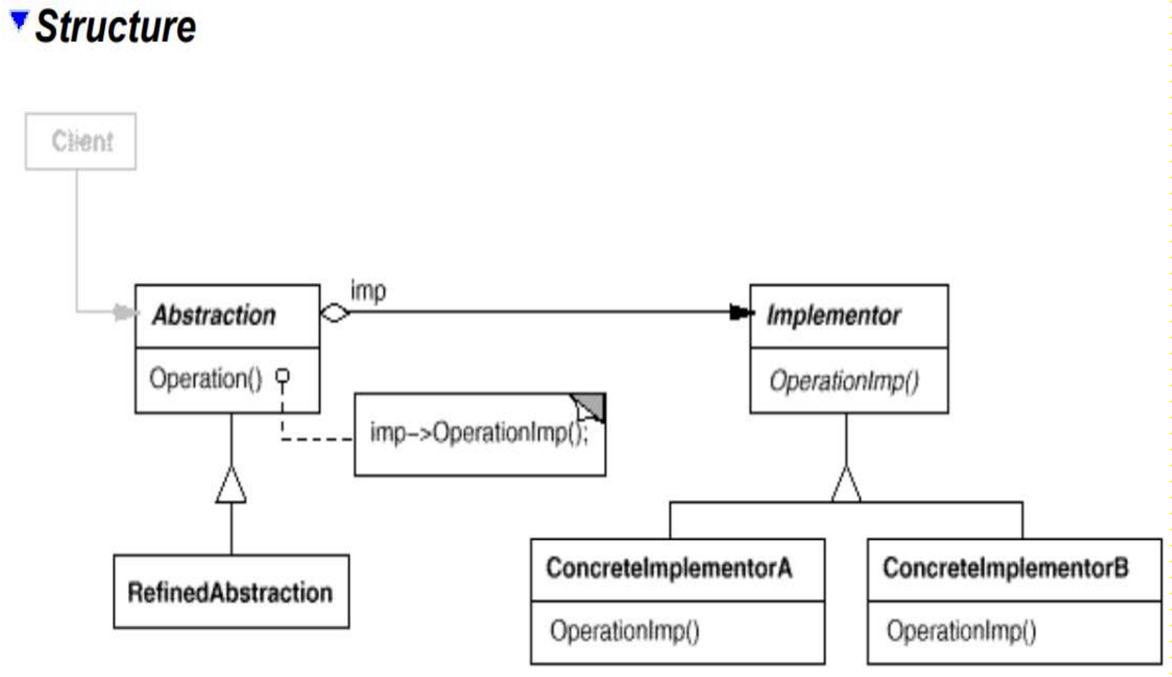
- You want to avoid a permanent binding between an abstraction and its implementation.
- both the abstractions and their implementations should be extensible by subclassing.
- changes in the implementation of an abstraction should have no impact on clients
- you want to share an implementation among multiple objects

PARTICIPANTS

Participant	Responsibility
Abstraction	• defines the abstraction's interface. • maintains a reference to an object of type Implementor.
Refined Abstraction	• Extends the interface defined by Abstraction.
Implementor	• defines the interface for implementation classes. This interface doesn't have to correspond exactly to Abstraction's interface • In fact the two interfaces can be quite different. Typically the Implementor interface provides only primitive operations, and Abstraction defines higher-level operations based on these primitives.
Concrete Implementor	• implements the Implementor interface and defines its concrete implementation.

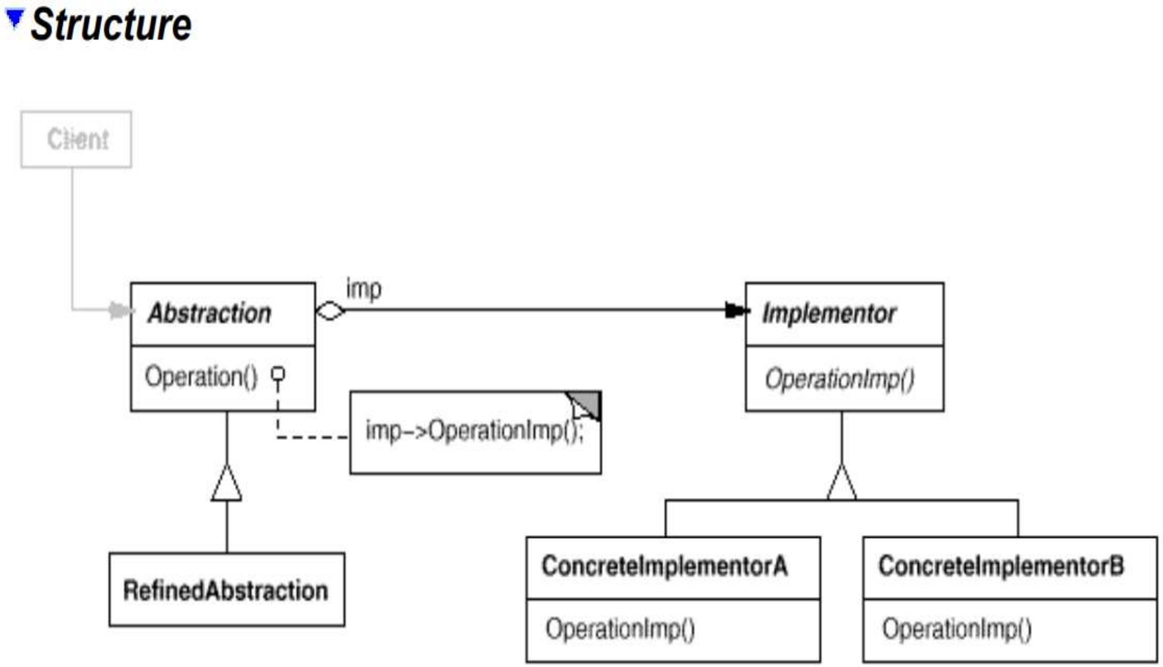
COLLABORATIONS

- Abstraction forwards client requests to its Implementor object.



CONSEQUENCES

- Decoupling interface and implementation.
- Improved extensibility.
- Hiding implementation details from clients.



IMPLEMENTATION

Implementation Consider the following issues when applying the Bridge pattern:

- **Only one Implementor.** In situations where there's only one implementation, creating an abstract Implementor class isn't necessary. This is a **degenerate case of the Bridge pattern;** there's a one-to-one relationship between Abstraction and Implementor
- **Creating the right Implementor object.** How, when, and where do you decide which Implementor class to instantiate when there's more than one?
  - If Abstraction knows about all Concrete Implementor classes, then it can instantiate one of them in its constructor; it can decide between them based on parameters passed to its constructor.
- **Sharing implementors.**

KNOWN USES & RELATED PATTERNS

- NeXT's AppKit [Add94] uses the Bridge pattern in the implementation and display of graphical images.
- An image can be represented in several different ways. The optimal display of an image depends on the properties of a display device, specifically its color capabilities and its resolution. Without help from AppKit, developers would have to determine which implementation to use under various circumstances in every application.

- **Abstract Factory**
- **Adaptor**