SHRI RAMDEOBABA COLLEGE OF
ENGINEERING AND MANAGEMENT,
NAGPUR - 440013

DESIGN PATTERNS
(CST324)
V SEMESTER

COURSE COORDINATOR: RINA DAMDOO

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROTOTYPE DESIGN PATTERNS

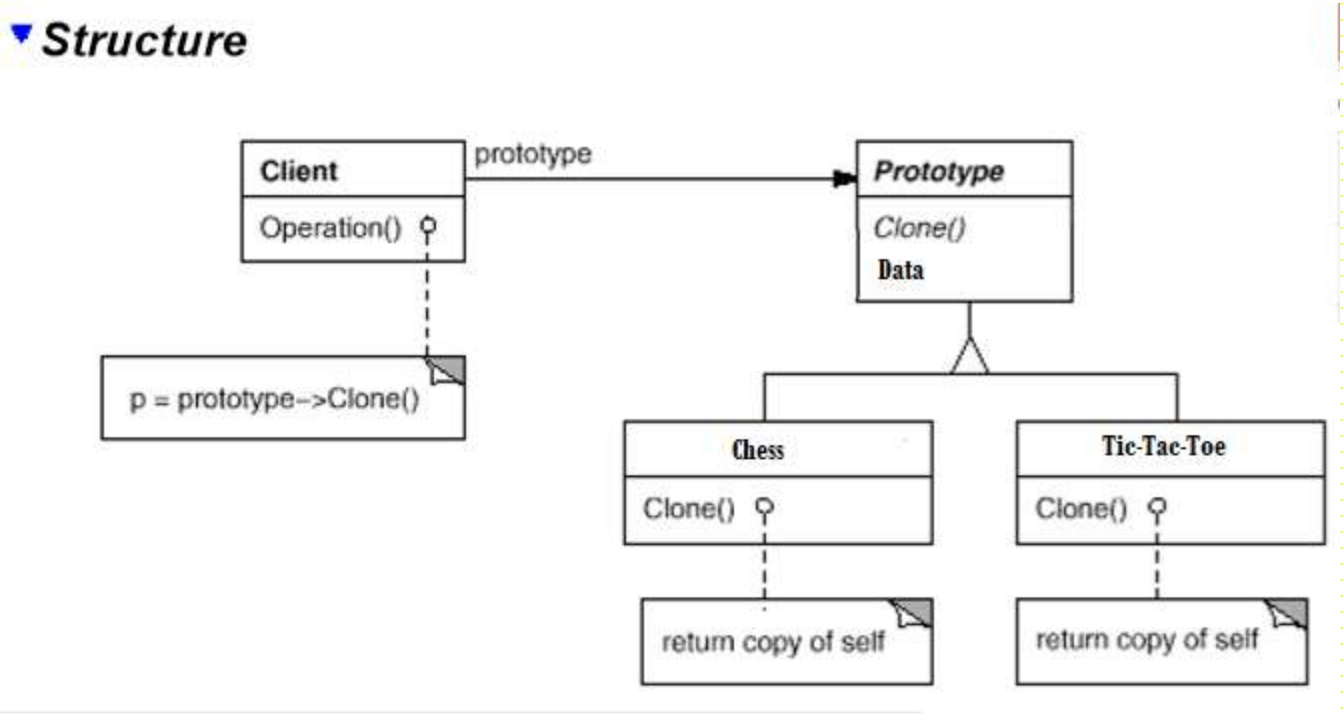
Intent

- Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype

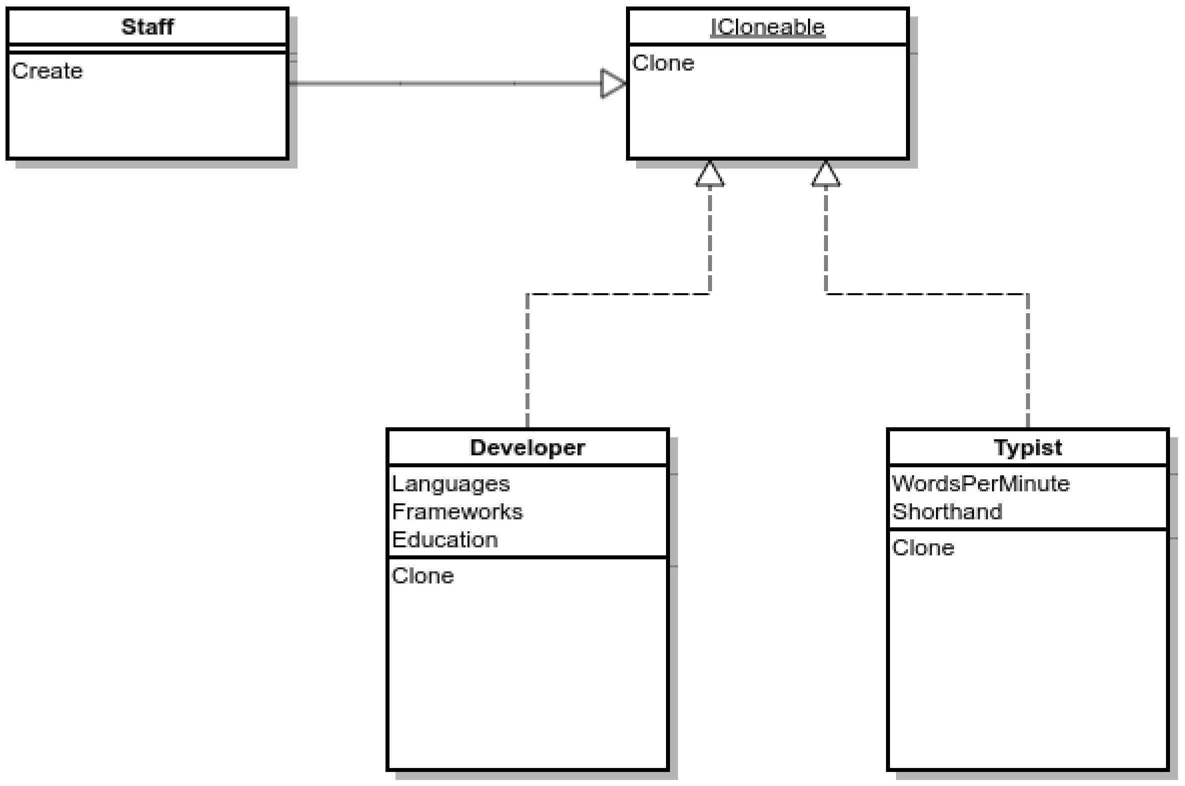
PROTOTYPE DESIGN PATTERNS

Problem Statement :

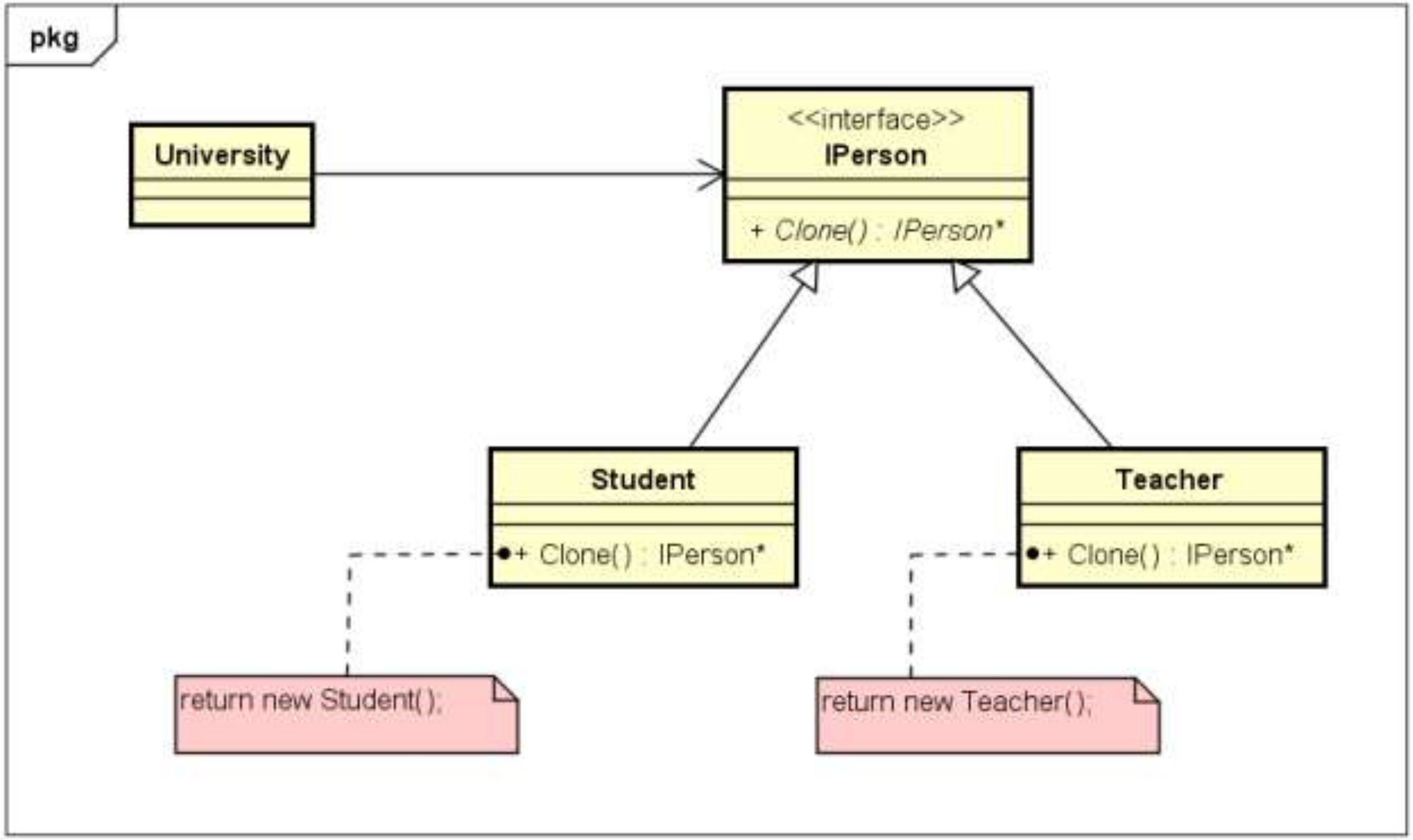
Implement a service to provide two types of prototypical instances Tic-Tac-Toe and Chess for games.



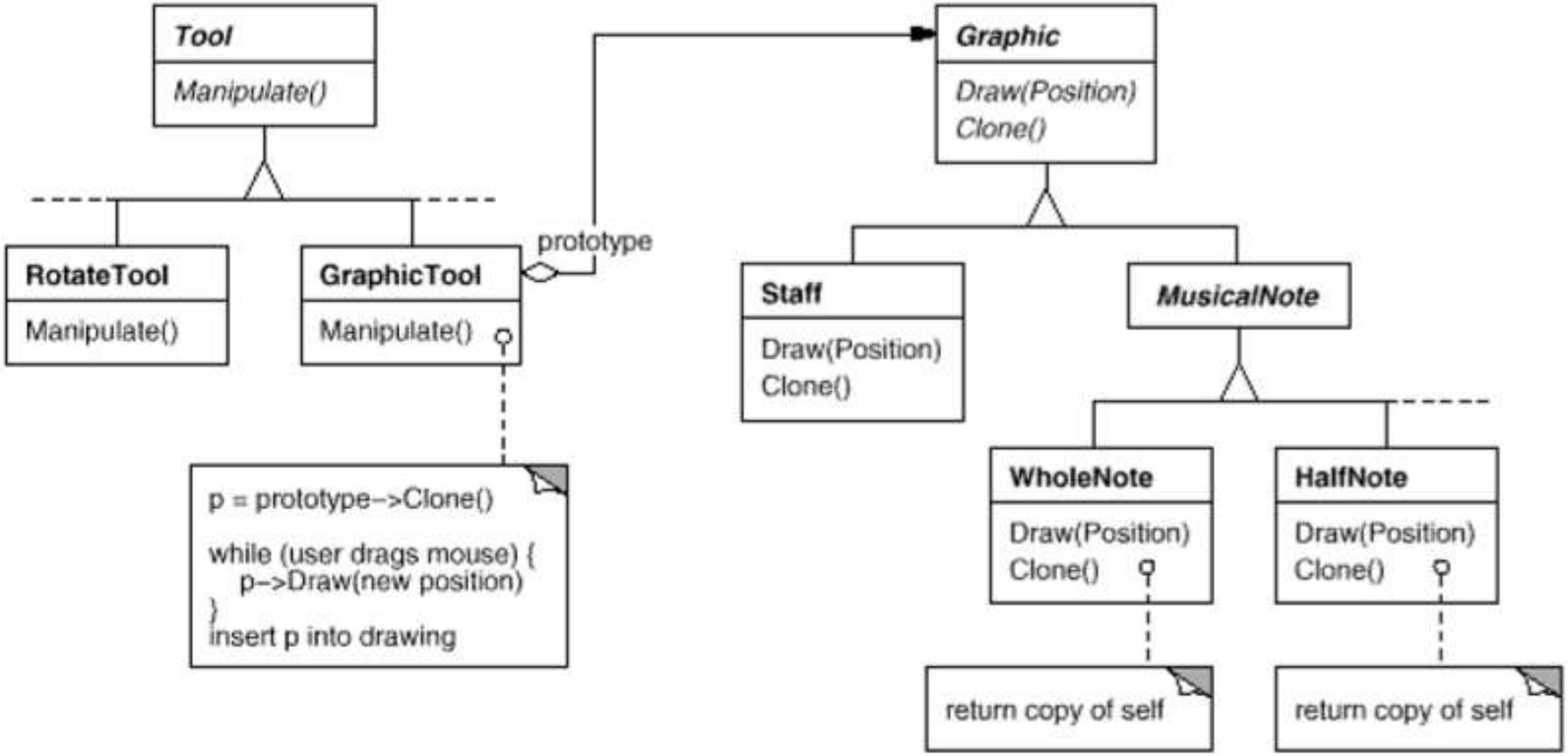
PROTOTYPE EXAMPLE



PROTOTYPE EXAMPLE



MOTIVATION



```
p = prototype->Clone()

while (user drags mouse) {
    p->Draw(new position)
}
insert p into drawing
```

Music editor example

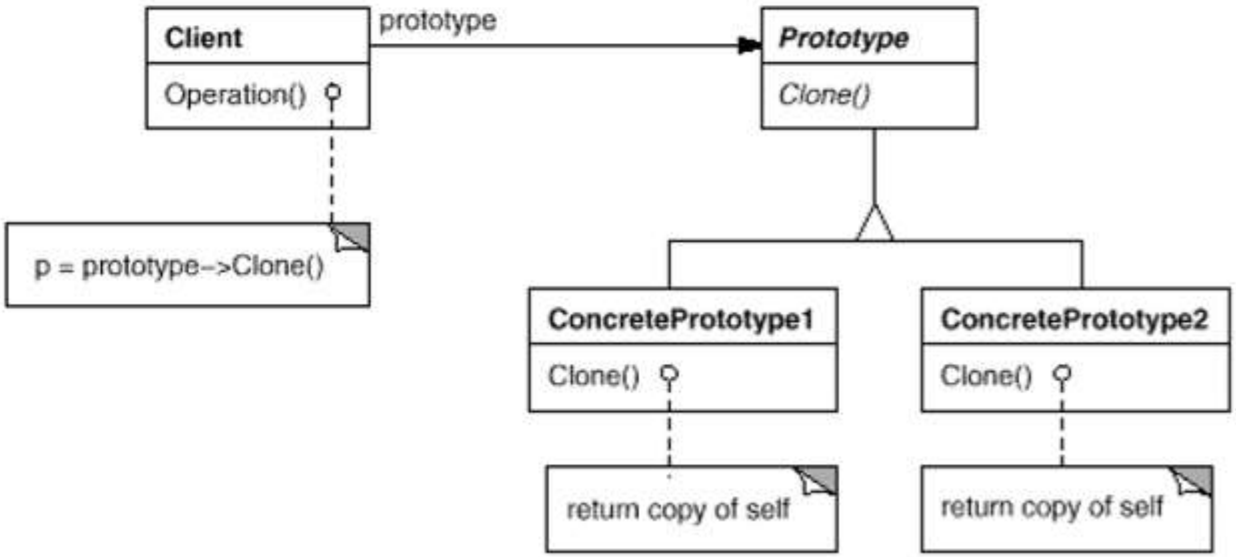
APPLICABILITY

Use the Prototype pattern when:

- A system should be independent of how its products are created, composed, and represented; and
    - When the classes to instantiate are specified at run-time, for example, by dynamic loading; or
    - To avoid building a class hierarchy of factories that parallels the class hierarchy of products; or
    - When instances of a class can have one of only a few different combinations of state. It may be more convenient to install a corresponding number of prototypes and clone them rather than instantiating the class manually, each time with the appropriate state.

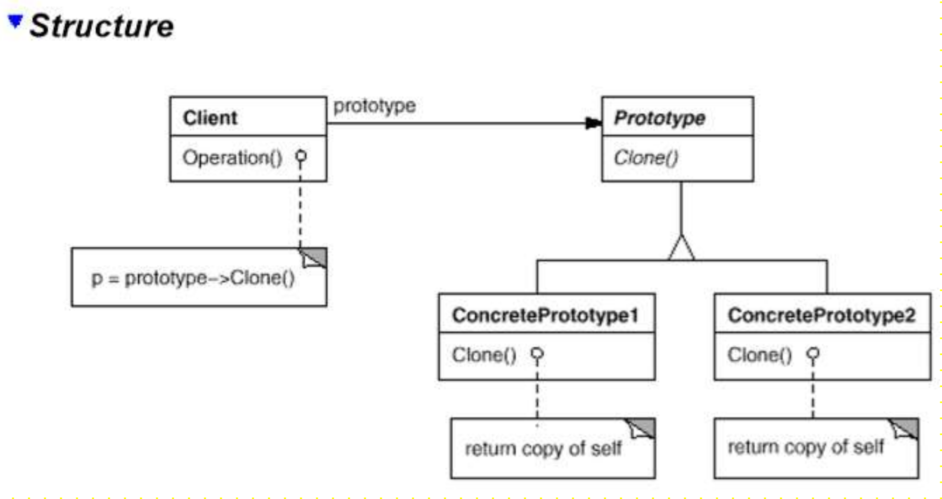
PROTOTYPE DESIGN PATTERNS

▼ *Structure*



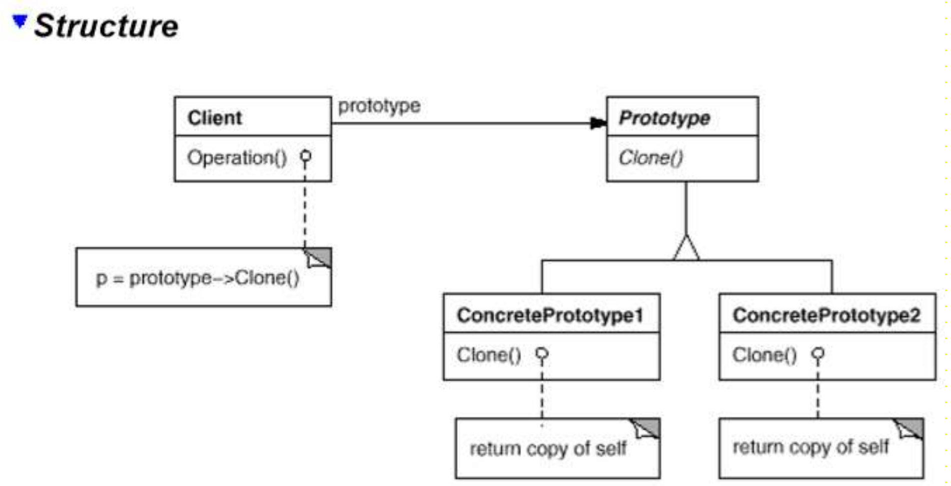
PARTICIPANTS

Participant	Responsibility
Prototype	Declares an interface for cloning itself
ConcretePrototype	Implements an operation for cloning itself.
Client	Creates a new object by asking a prototype to clone itself.



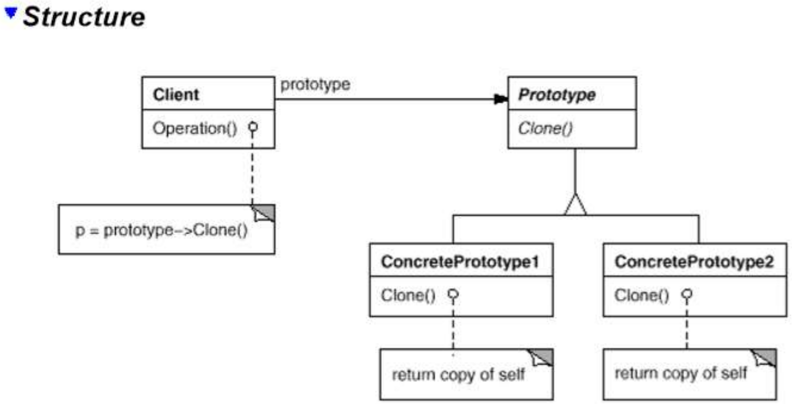
COLLABORATIONS

- A client asks a prototype to clone itself.



CONSEQUENCES

1. Many of the same consequences that Abstract Factory has. It hides the concrete product classes from the client, thereby reducing the number of names clients know about.
2. Adding and removing products at run-time.
3. Specifying new objects by varying values.
4. Specifying new objects by varying structure.
5. Reduced subclassing.

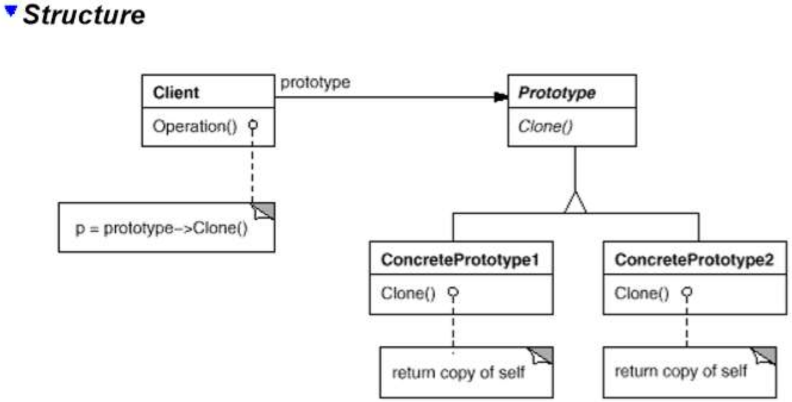


IMPLEMENTATION

Implementation Consider the following issues when applying the Prototype pattern:

1. **Using a prototype manager.** Prototype manager (registry) is an associative store that returns the prototype matching a given key. It has operations for registering a prototype under a key and for unregistering it.
2. **Implementing the Clone operation:**
   - Shallow copy
   - Deep copy

3. Initializing clones. Want to initialize some or all of clone's internal state to values of client's choosing.



KNOWN USES & RELATED PATTERNS

- Class objects act like prototypes in languages (This benefit applies primarily to languages like C++).

- **Abstract Factory** : An Abstract Factory might store a set of prototypes from which to clone and return product objects.

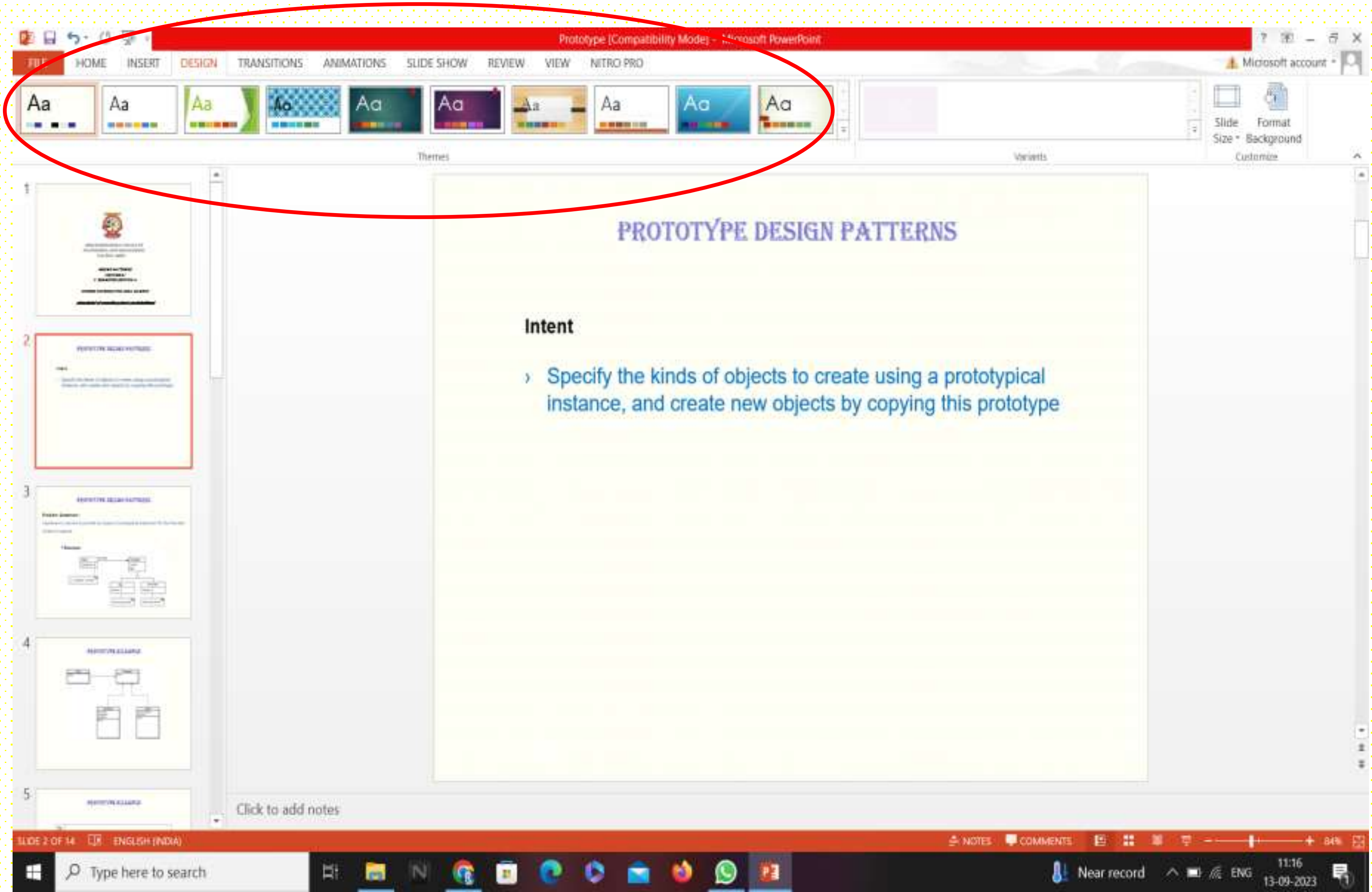
- Designs that make heavy use of the **Composite** (163) and **Decorator** (175) patterns often can benefit from Prototype

WHEN PROTOTYPE PATTERN IS USED……….

- the Object creation is a costly affair and requires a lot of time and resources and you have a similar object already existing.

- A possible real world application: when you need to create a spreadsheet containing many cells. Rather than set the style for each newly created cell to override the default styling, you'd use a Prototype pattern to create a template cell, and clone that cell when creating new cells.

APPLICATION



PROTOTYPE DESIGN PATTERNS

Intent

- Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype

Click to add notes

- Shallow Copy and Deep Copy are types of cloning in Prototype Design patterns.

- In shallow copy, we only cloned the parent object and not its containing objects while in deep copy, we cloned the parent object as well as its containing objects.