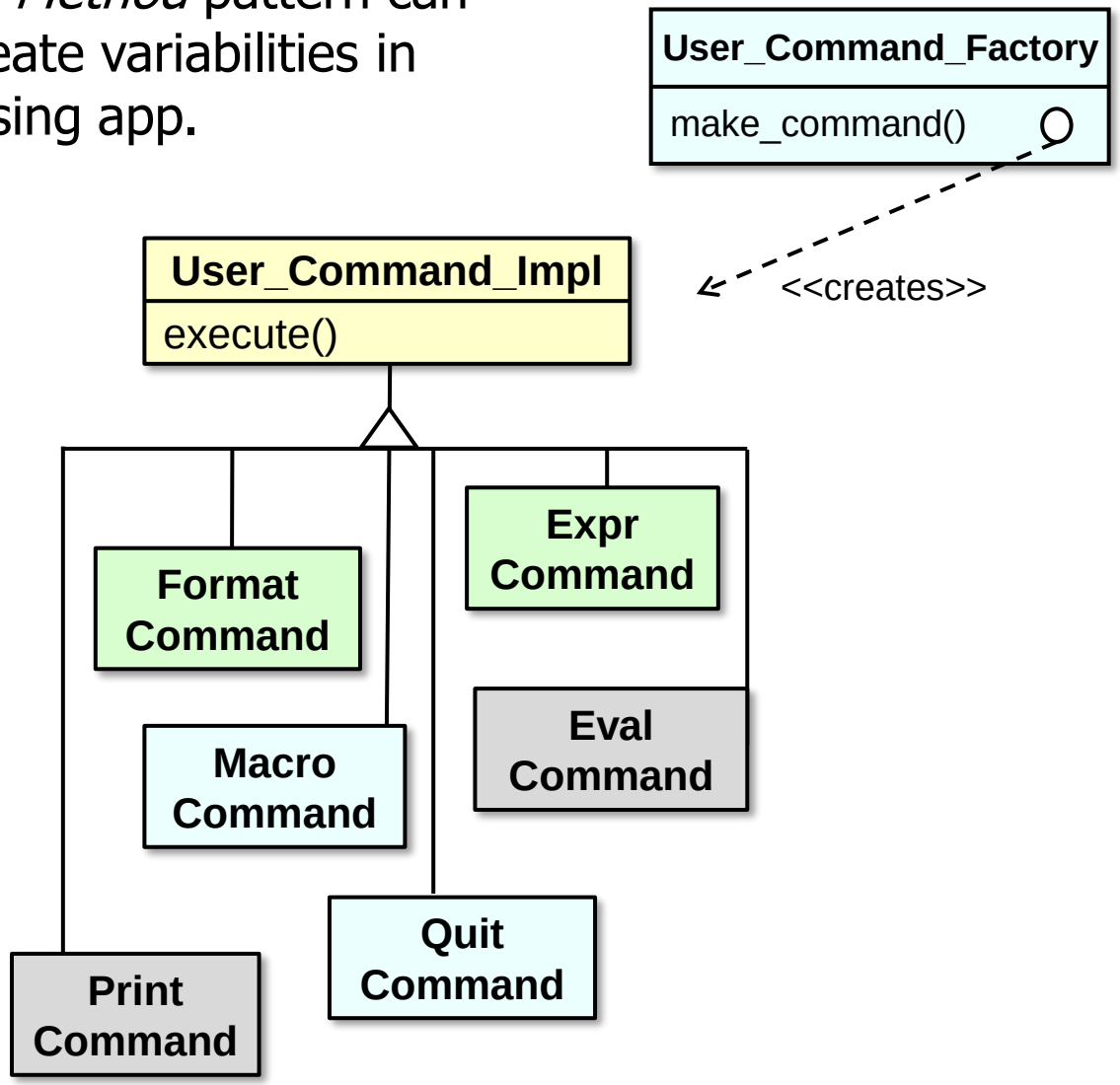
The Factory Method Pattern

Motivating Example

Douglas C. Schmidt

Learning Objectives in This Lesson

- Recognize how the *Factory Method* pattern can be applied to extensibly create variabilities in the expression tree processing app.

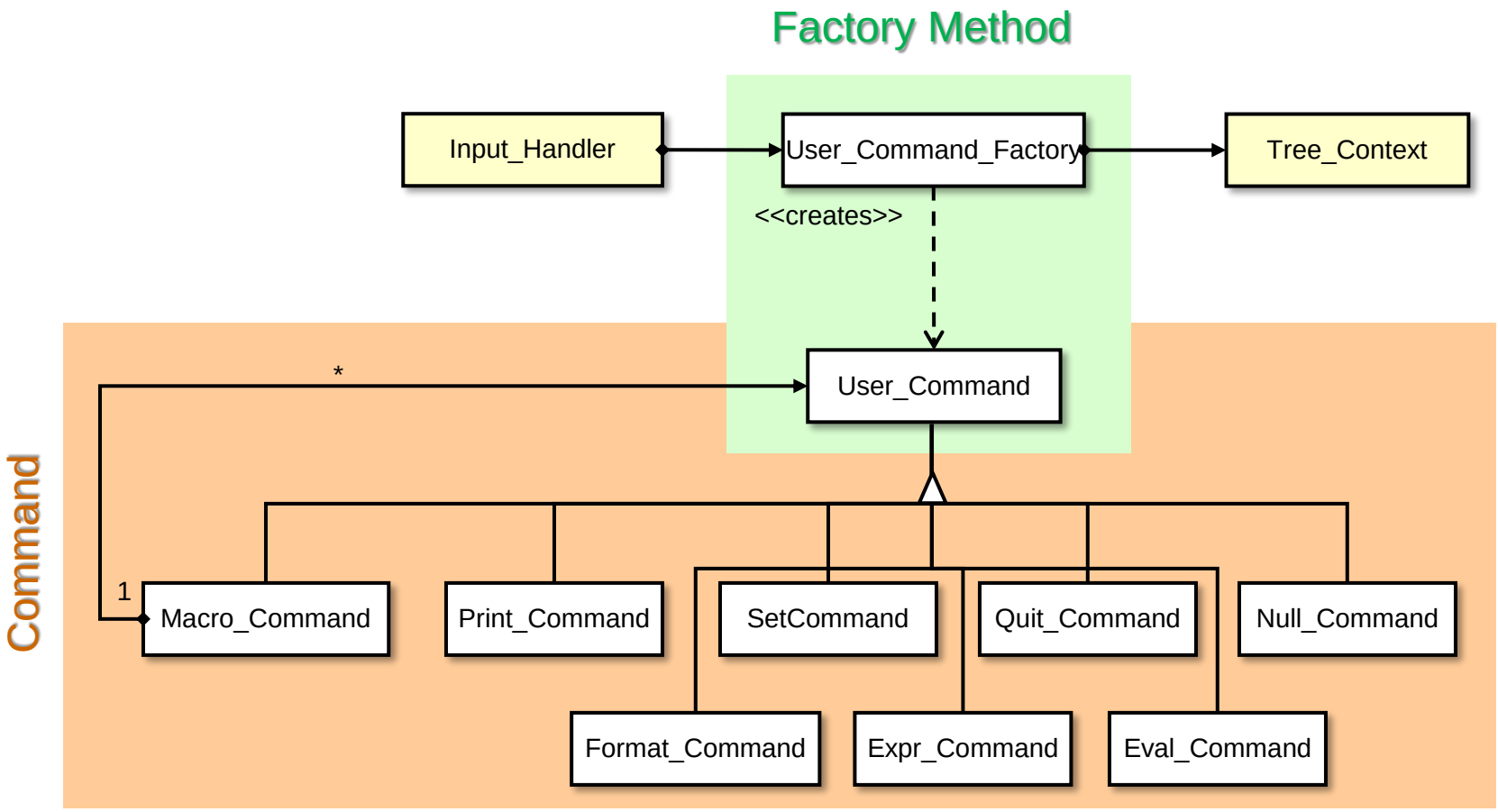


Douglas C. Schmidt

Motivating the Need for the Factory Method Pattern in the Expression Tree App

A Pattern for Abstracting Object Creation

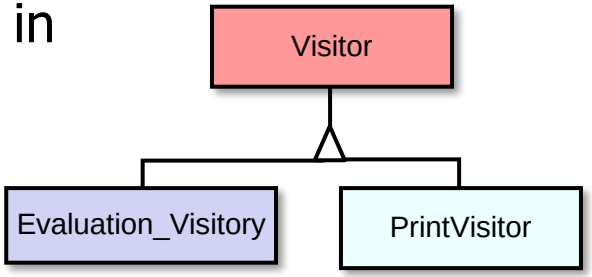
***Purpose**: Enable the extensible creation of variabilities, such as commands, iterators, & visitors.*

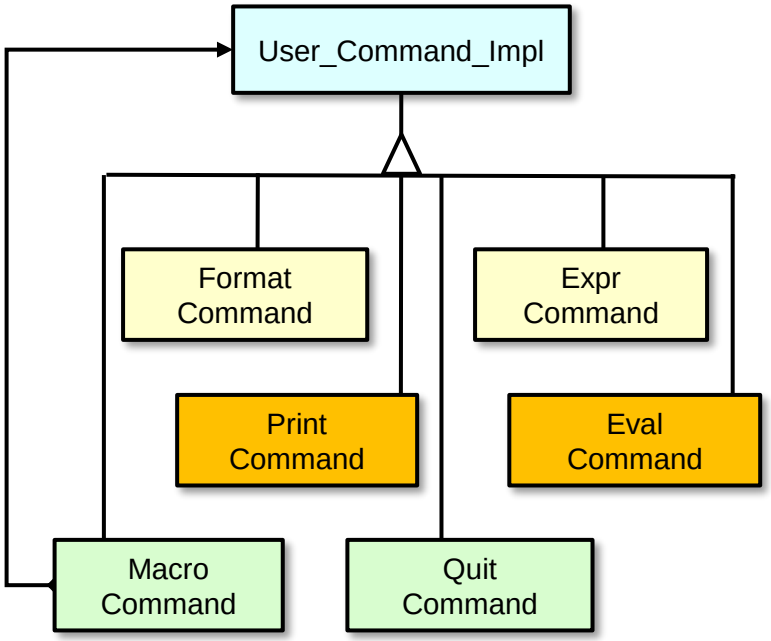


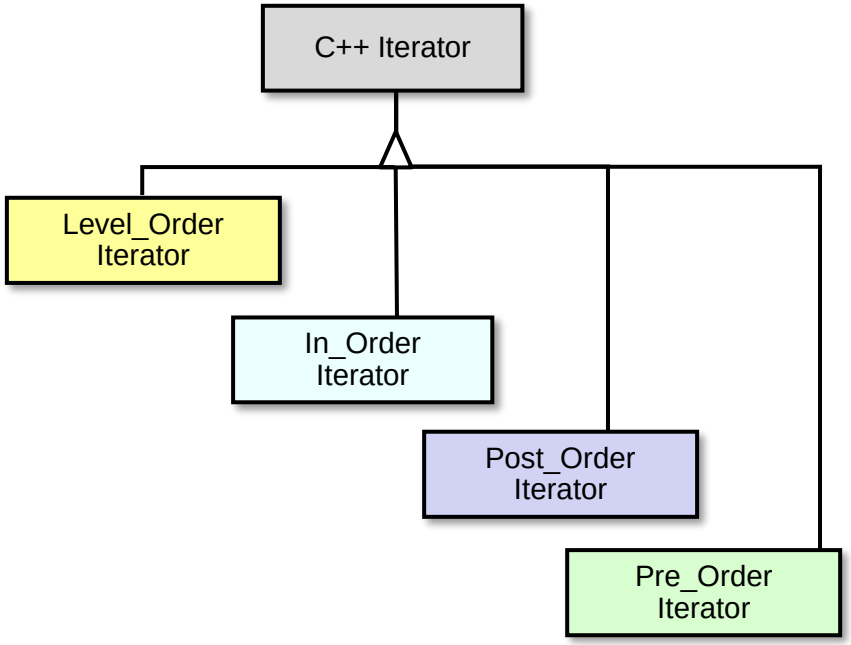
Factory Method decouples the creation of objects from their subsequent use.

Context: OO Expression Tree Processing App

- There are many points of variability in the expression tree processing app.
  - e.g., user commands, traversal strategies, & visitor operations applied on an expression tree

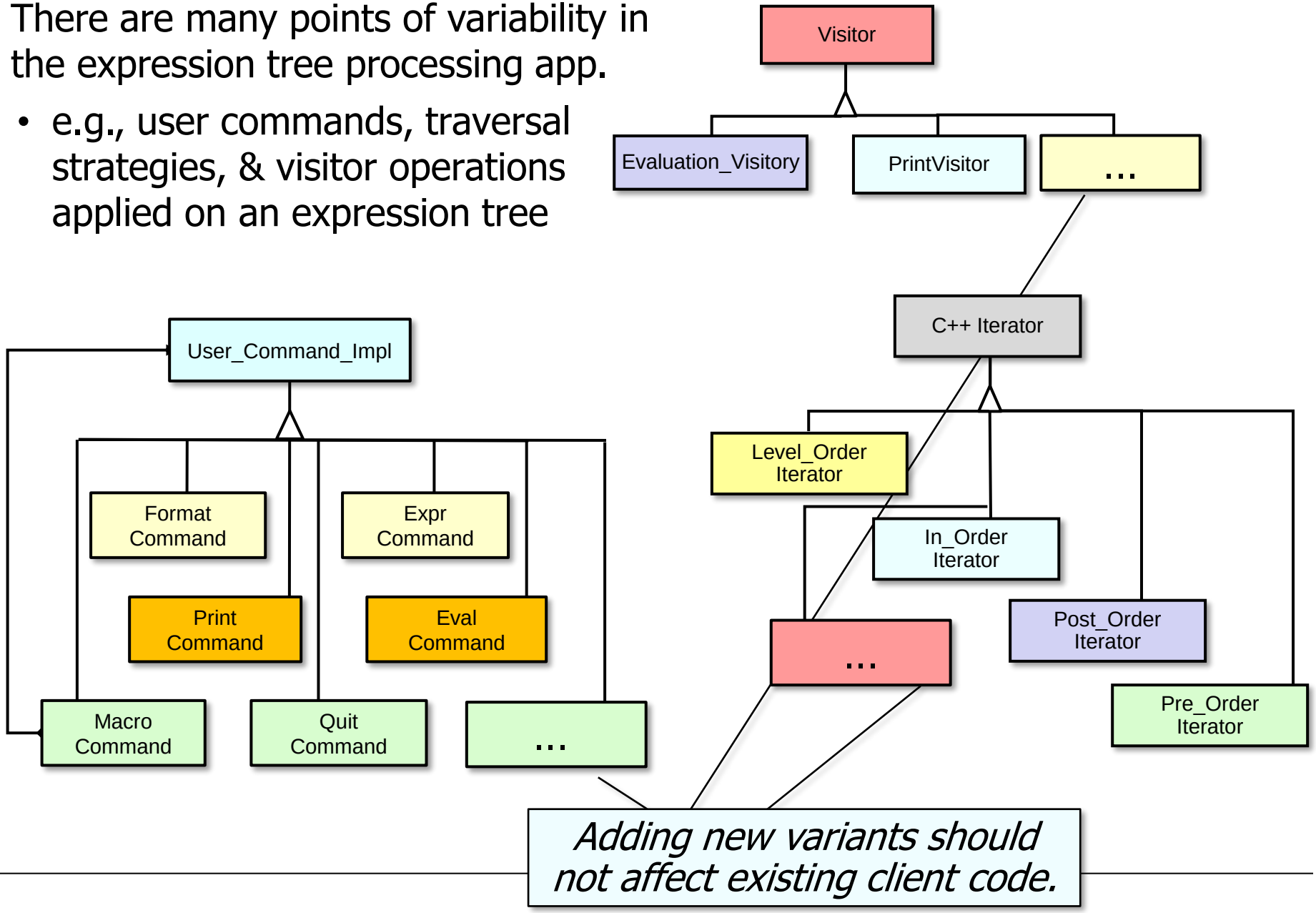






Context: OO Expression Tree Processing App

- There are many points of variability in the expression tree processing app.
  - e.g., user commands, traversal strategies, & visitor operations applied on an expression tree

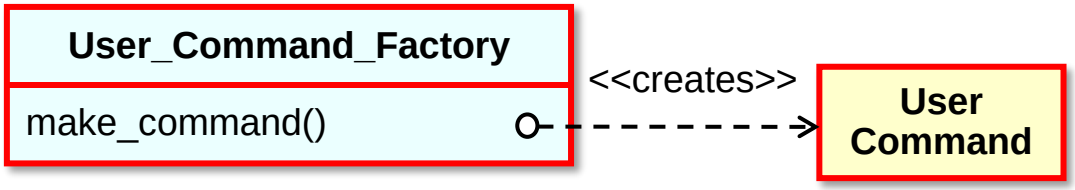


Problem: Inflexible Creation of Variabilities

- Tightly coupling the creation of variabilities with client code is problematic.
  - e.g., hard-coding lexical dependencies on specific derived classes can complicate maintenance & impede extensibility

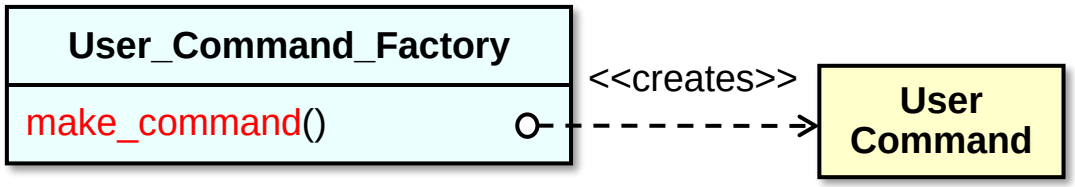
Solution: Abstract Creation of Objects

- Define a **`User_Command_Factory`** class whose **`make_command()`** factory method creates a **`User_Command`** object.



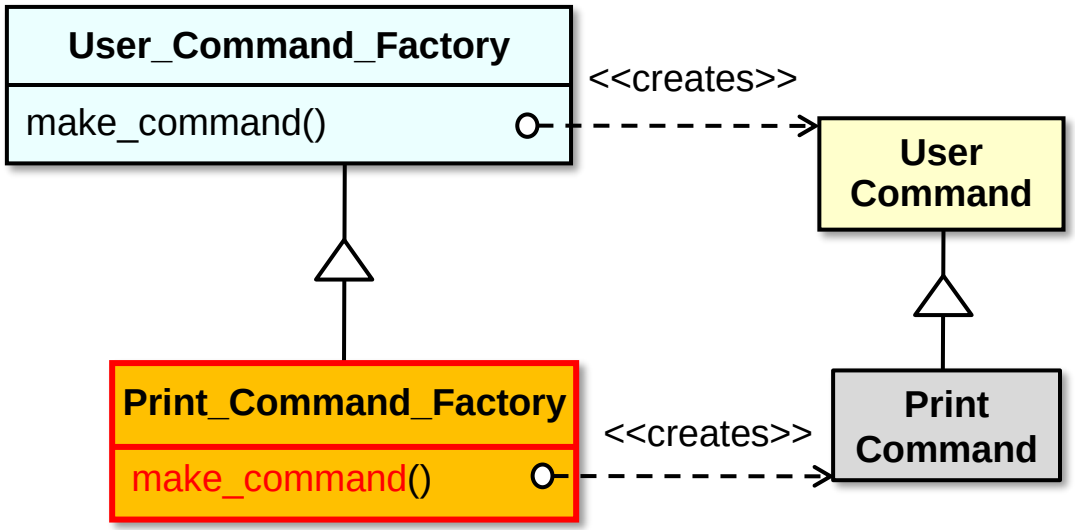
Solution: Abstract Creation of Objects

- Have the **`make_command()`** factory method implement the appropriate derived class of **`User_Command`**



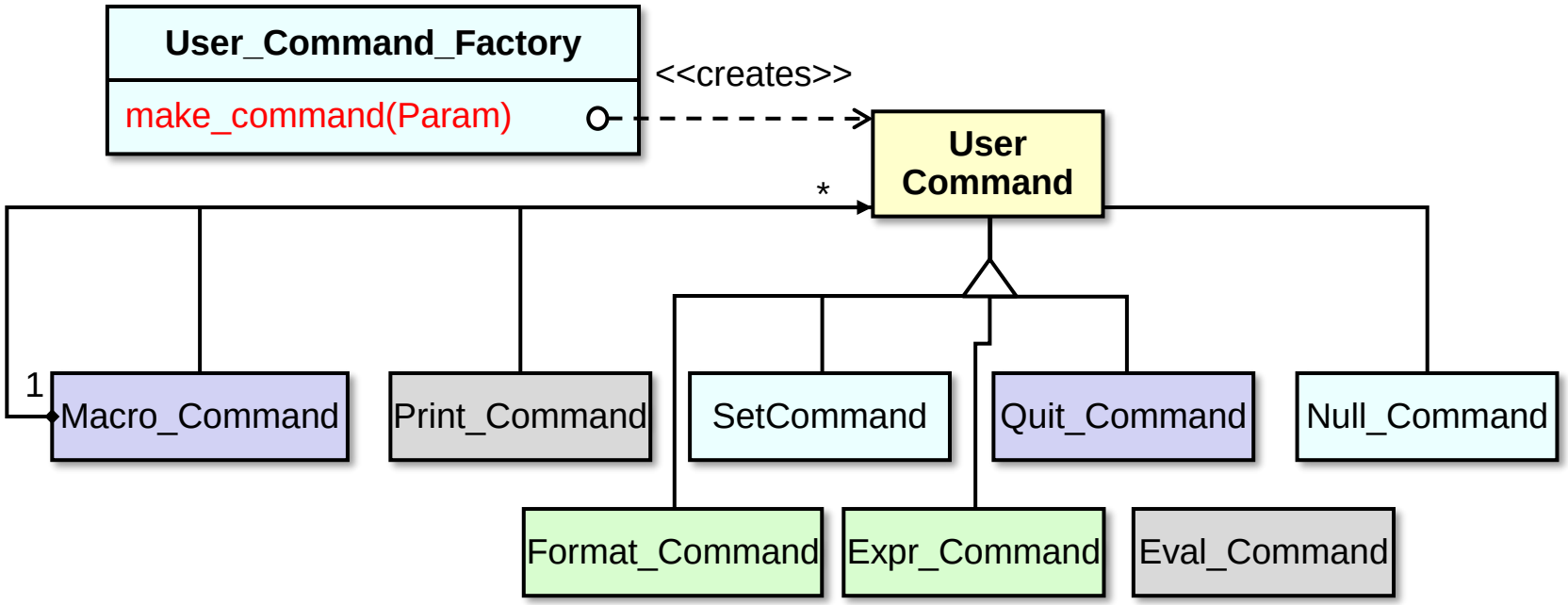
Solution: Abstract Creation of Objects

- Have the **`make_command()`** factory method implement the appropriate derived class of **`User_Command_Impl`**, e.g.,
  - Subclass **`User_Command_Factory`** & override the factory method **`make_command()`**



Solution: Abstract Creation of Objects

- Have the **`make_command()`** factory method implement the appropriate derived class of **`User_Command_Impl`**, e.g.,
  - Subclass **`User_Command_Factory`** & override the factory method **`make_command()`**



- Or pass a parameter to the **`make_command()`** factory method & use it to create the appropriate **`User_Command_Impl`** derived class objects

User_Command_Factory Class Overview

- Create the command corresponding to the user input.

Class methods

```
User_Command make_command(string inputstring)
             ...
```

User_Command_Factory Class Overview

- Create the command corresponding to the user input.

Class methods

```
User_Command make_command(string inputstring)
             ...
```

This is a factory method

User_Command_Factory Class Overview

- Create the command corresponding to the user input.

Class methods

```
User_Command make_command(string inputstring)
             ...
```

- **Commonality**: provides a common API to create commands
- **Variability**: implementations of expression tree command factory methods can vary depending on the requested commands

User_Command_Factory Class Overview

- Create the command corresponding to the user input.