Jacob Hreshchyshyn
Rithvik Arun
Alex Altnether
Joseph Hale

Example 1:
https://refactoring.guru/design-patterns/flyweight

Flyweight: This pattern is useful in defining many instances of similar objects. The above link provides a description of the pattern. Additionally, Robert Nystrom's *Game Programming Patterns* provides an example of the flyweight's use in creating a forest of trees. Instead of creating a class that stores the state for every single instance of a tree, one class can contain the shared attributes of a tree, like its mesh and textures, while a separate class can contain attributes that make the tree unique in the forest, like the tree instance's individual height and width characteristics.

Example 2:
https://refactoring.guru/design-patterns/builder

Builder: This pattern is useful for defining complex objects out of nested simple objects. The example in the link above shows an example of different house objects. Rather than creating a new object for the same house with different features, you can use the house as the simple object and add nest other objects inside of it such as a garden or swimming pool. This process can help you make complex objects easily.

Example 3:
https://refactoring.guru/design-patterns/strategy

Strategy: This pattern is useful because it lets you define a group of algorithms to allow them to make their objects interchangeable while putting them in separate classes. This works by defining an interface for the algorithm, and creating implementations of that interface with the different variants of the algorithm. By swapping out the implementation, one can dynamically change the algorithm used at runtime. The link above shows different ways to use transportation to get to the place you need to go, such as the airport.

Discussion Topics:
In this meeting, we discussed the use of the Flyweight, Builder, and Strategy design patterns. In exploring these topics, we shared resources that provide good information and practical examples on structural design patterns.