Use the ***Flyweight Pattern*** when one instance of a class can be used to provide many “virtual instances”.

***Scenario: Landscape design for huge planned communities:***

House

Instead of having thousands of Tree objects, you could redesign your system so that you have got only one instance of Tree, and a client object that maintains the state of ALL your Trees.

All the state for ALL of your virtual Tree Each tree instance maintains its own state.

Objects, os stored in this 2D-way One, single, state-free Tree object

TreeManager

treeArray

displayTrees() {

// for all trees {

//get array row

display(x, y, age);

}

}

Tree

display(x, y, age) {

// use X-Y cords

// & complex age

// related calcs

}

***Benefits:***

* Reduces the number of object instances at runtime, saving memory.
* Centralizes the state for many “virtual” objects into a single location.

***Uses and Drawbacks:***

* The Flyweight is used when a class has many instances, and they can all be controlled identically.
* A drawback of the Flyweight pattern is that once you have implemented it, single, logical instances of the class will not be able to behave independently from the other instances.