*Creational Patterns*

This design patterns is all about class instantiation. This pattern can be further divided into class-creation patterns and object-creational patterns. While class-creation patterns use inheritance effectively in the instantiation process, object-creation patterns use delegation effectively to get the job done.

* [Abstract Factory](http://sourcemaking.com/design_patterns/abstract_factory)Creates an instance of several families of classes
* [Builder](http://sourcemaking.com/design_patterns/builder)Separates object construction from its representation
* [Factory Method](http://sourcemaking.com/design_patterns/factory_method)Creates an instance of several derived classes
* [Object Pool](http://sourcemaking.com/design_patterns/object_pool)Avoid expensive acquisition and release of resources by recycling objects that are no longer in use
* [Prototype](http://sourcemaking.com/design_patterns/prototype)A fully initialized instance to be copied or cloned
* [Singleton](http://sourcemaking.com/design_patterns/singleton)A class of which only a single instance can exist

**Prepared by**: Daniel Santos

**Date**: 10 September 2017

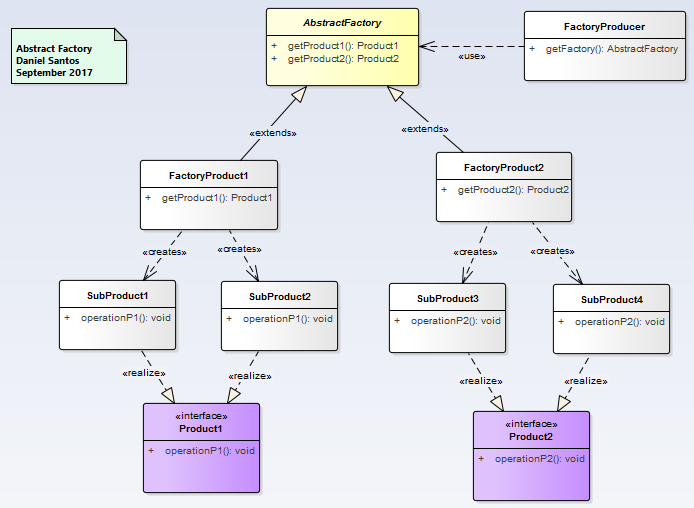
# **Abstract Factory**

# **Creates an instance of several families of classes**

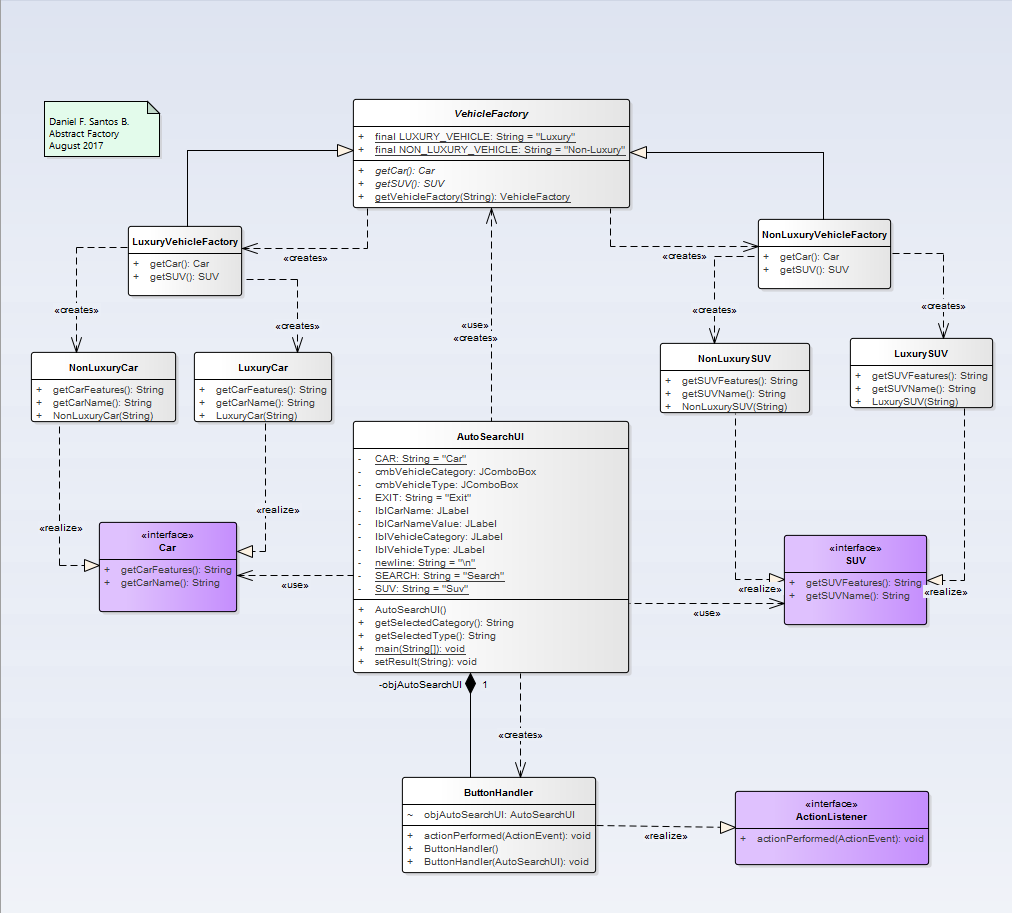
* Provide an interface for creating families of related or dependent objects without specifying their concrete classes.
* A hierarchy that encapsulates: many possible "platforms", and the construction of a suite of "products".

The new operator considered harmful.

General Model



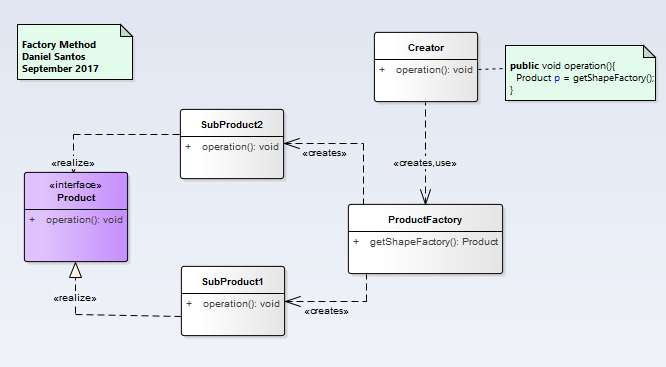
Example model



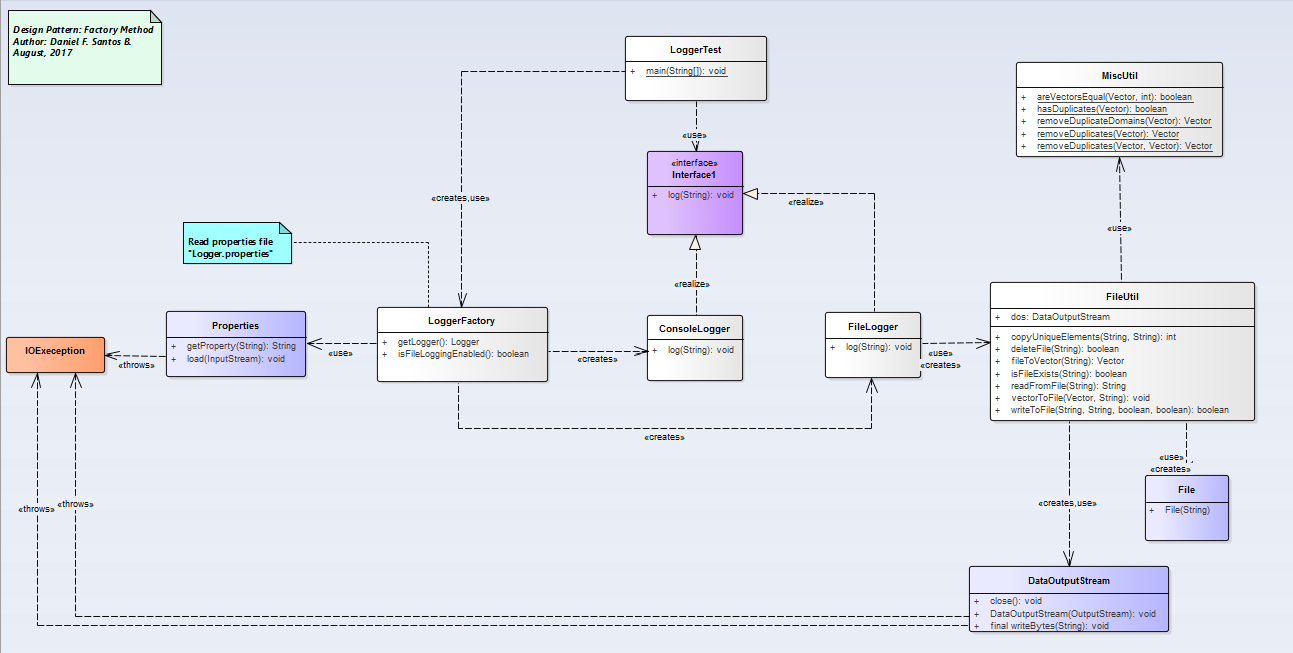
# **Factory Method: Creates an instance of several derived classes**

* Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.
* Defining a "virtual" constructor.
* The new operator considered harmful.

General Model



Example Model

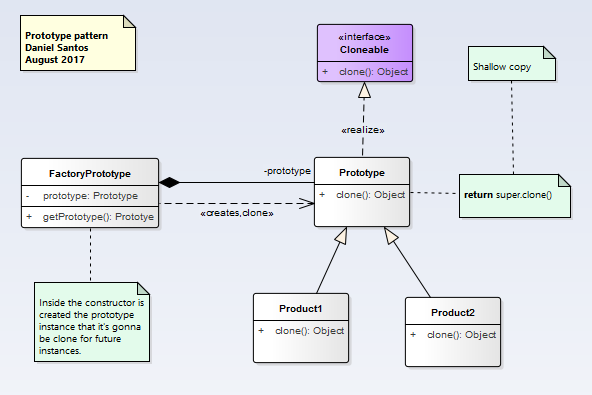


# **Prototype: A fully initialized instance to be copied or cloned**

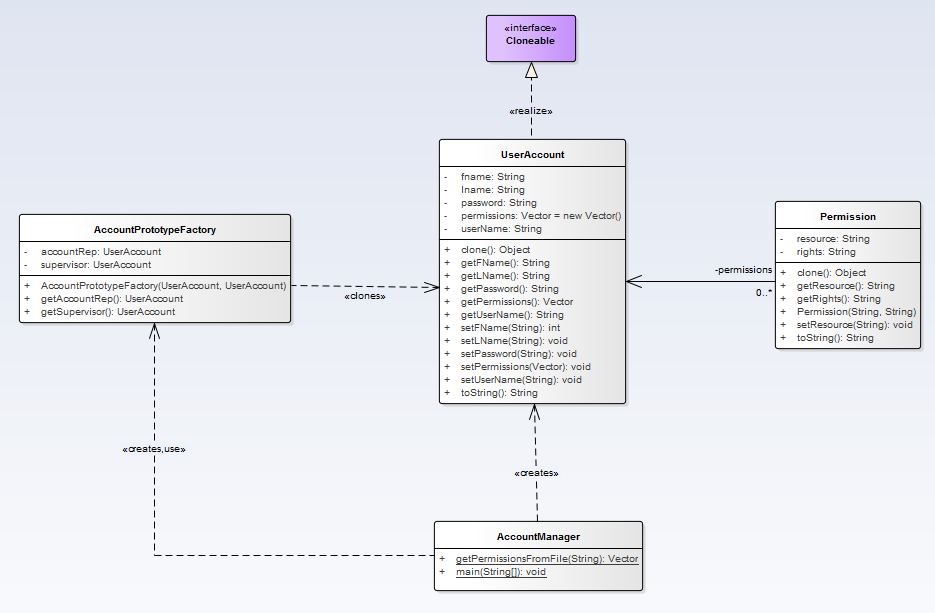
* Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.
* Co-opt one instance of a class for use as a breeder of all future instances.

The new operator considered harmful.

General Model



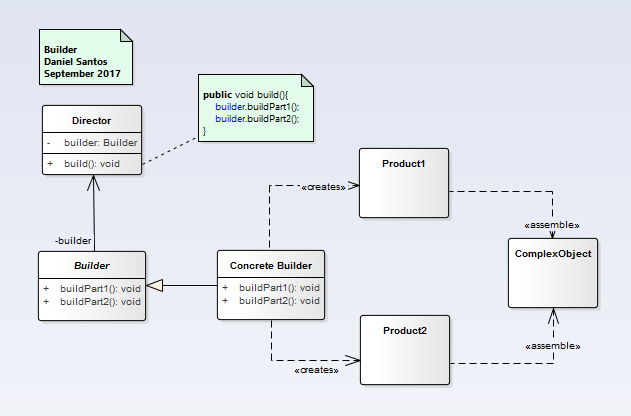
Example model



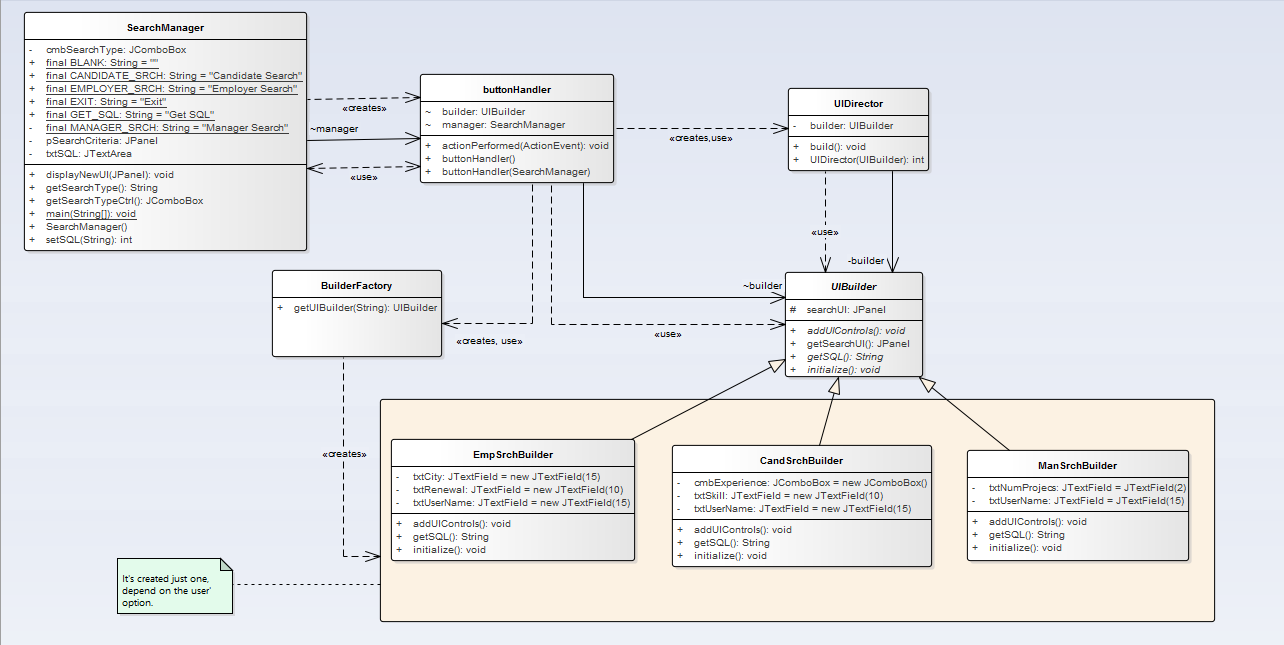
# **Builder**

* Separate the construction of a complex object from its representation so that the same construction process can create different representations.
* Parse a complex representation, create one of several targets.

General Model

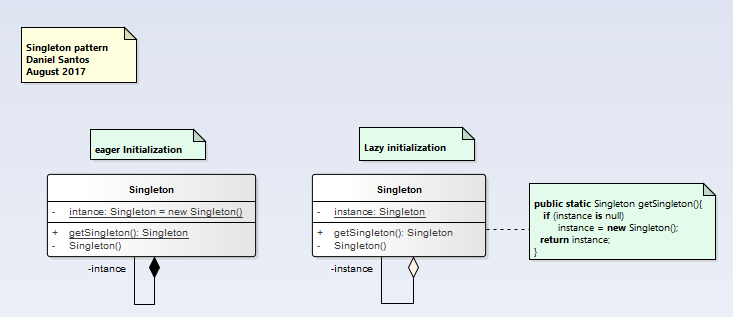


Example Model

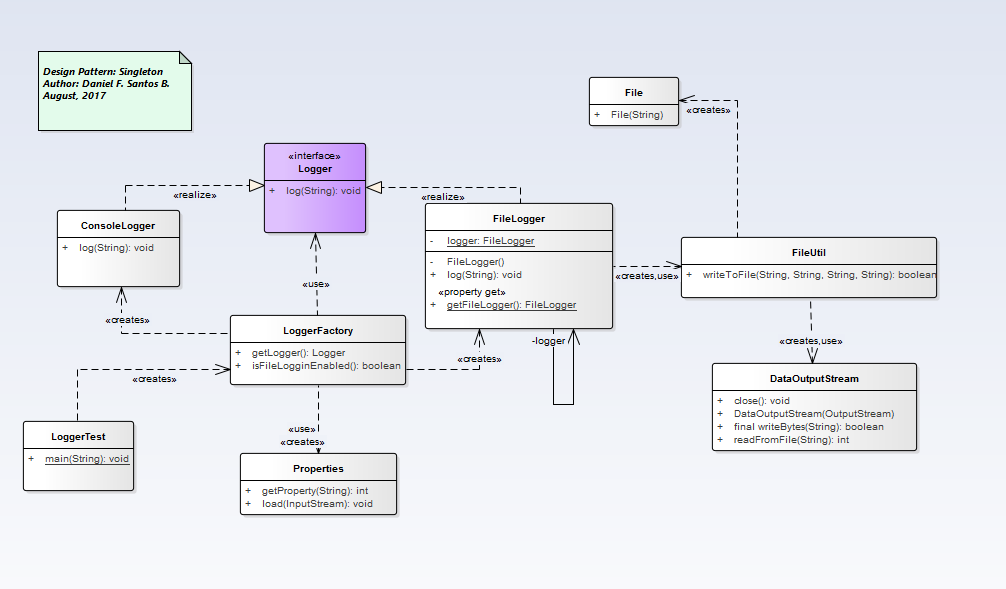


# **Singleton**

* Ensure a class has only one instance, and provide a global point of access to it.
* Encapsulated "just-in-time initialization" or "initialization on first use".

General Model

Example Model



**Object Pool: Avoid expensive acquisition and release of resources by recycling objects that are no longer in use**

# Object pooling can offer a significant performance boost; it is most effective in situations where the cost of initializing a class instance is high, the rate of instantiation of a class is high, and the number of instantiations in use at any one time is low.

