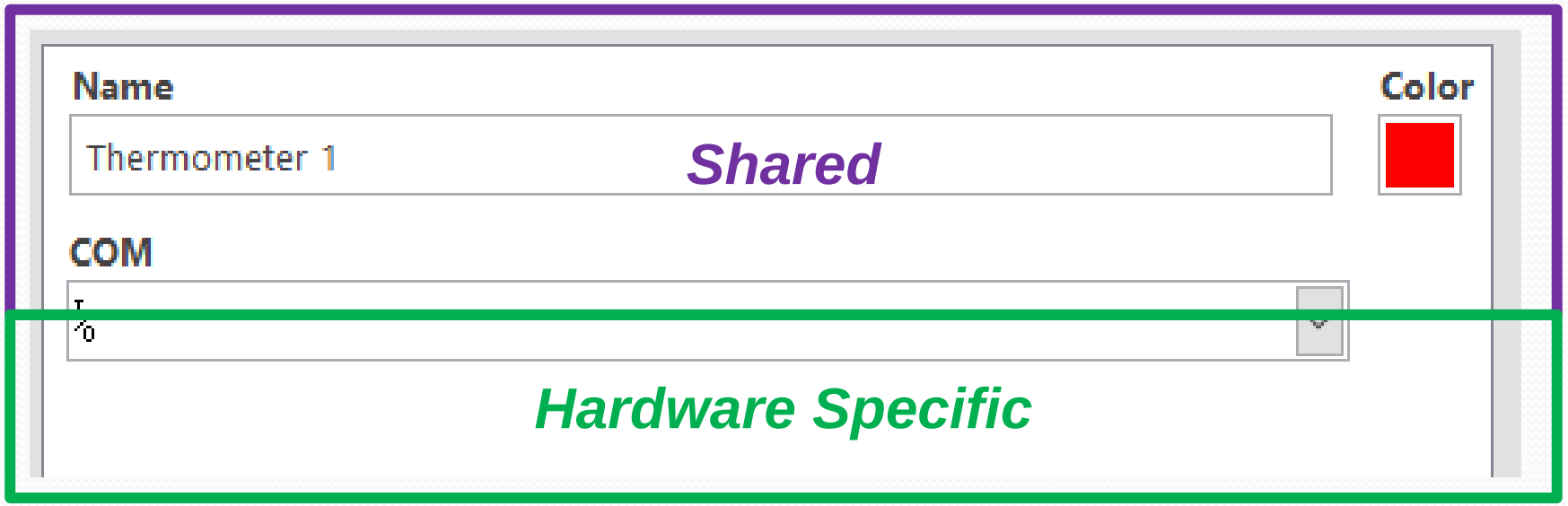
A Practical Demonstration on Using Interfaces for Class Configuration

Hope Harrison

Moore Good Ideas, Inc.

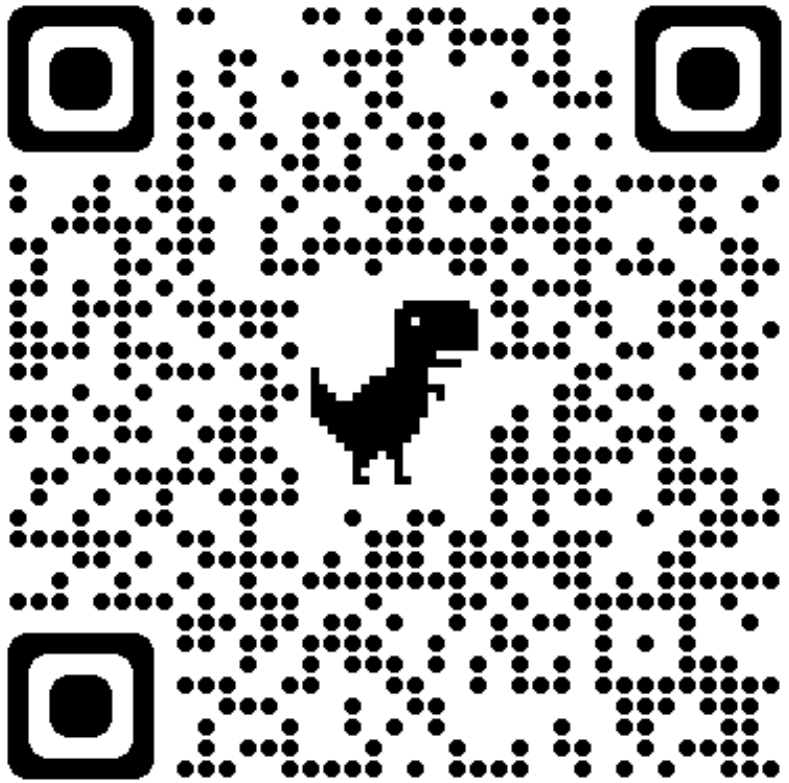
Demonstration

- **What we want to achieve**
  - Save and load different types of classes
  - Tiered configuration – meaning some sibling classes will share some configuration
  - Not required to have a common ancestor between all “configurable” classes



Source Code

https://gitlab.com/mgi/configuration-framework



Background

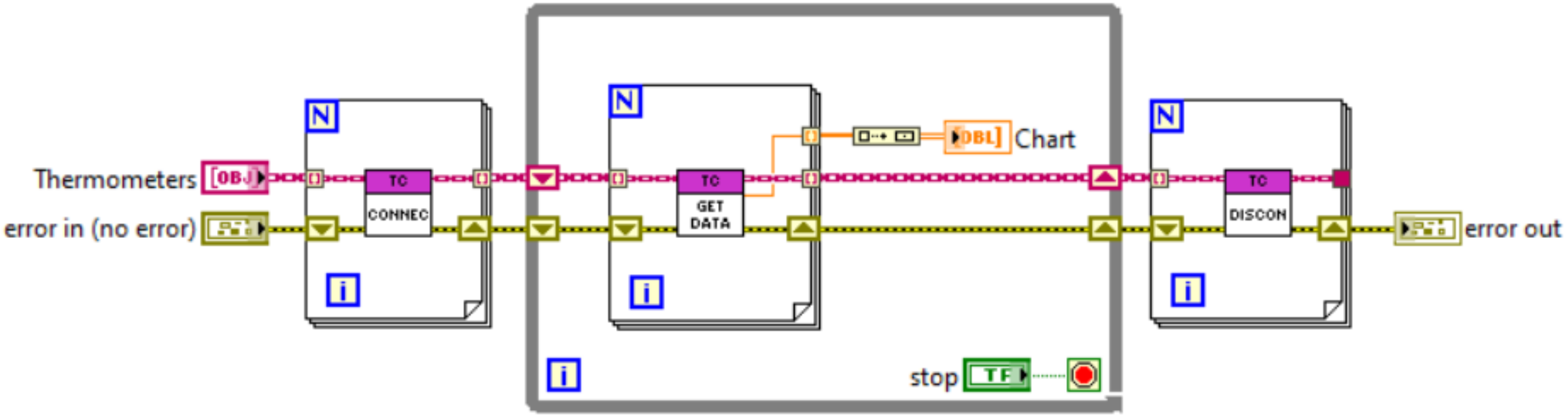
This presentation uses actors heavily, if you are not familiar you can learn more here: https://www.mooregoodideas.com/actor-framework/basics/AF-basics-part-1/

We also use Panel Actors which you can learn more about here: https://www.mooregoodideas.com/mgi-library/panel-manager/panel-actors/index.html

Quick overview: *Actors* are objects that run asynchronously and can send and respond to messages. *Panel Actors* are actors that have a UI and can easily be placed in a window or subpanel.

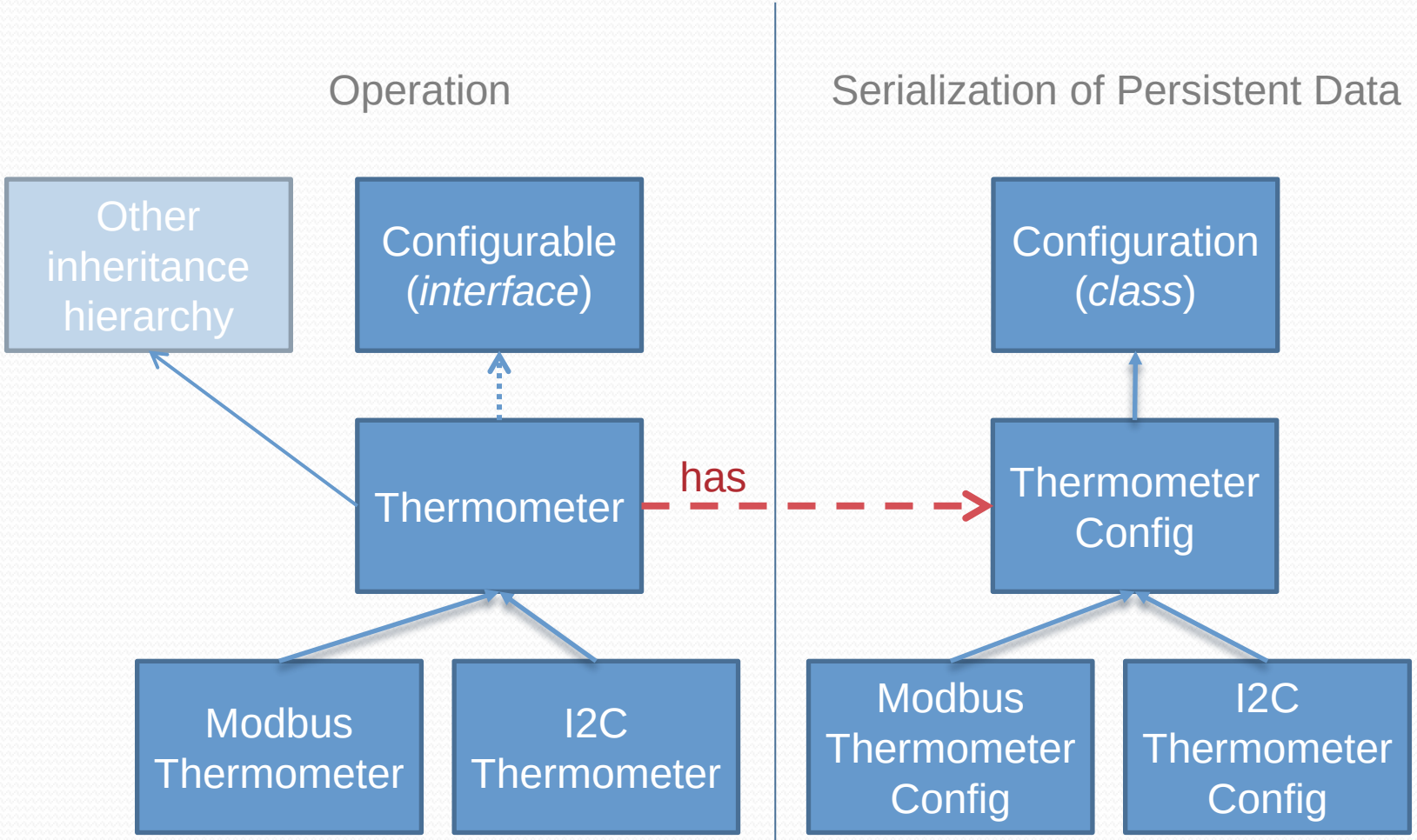
You can still use parts of this framework in a non-actor architecture

Starting Point



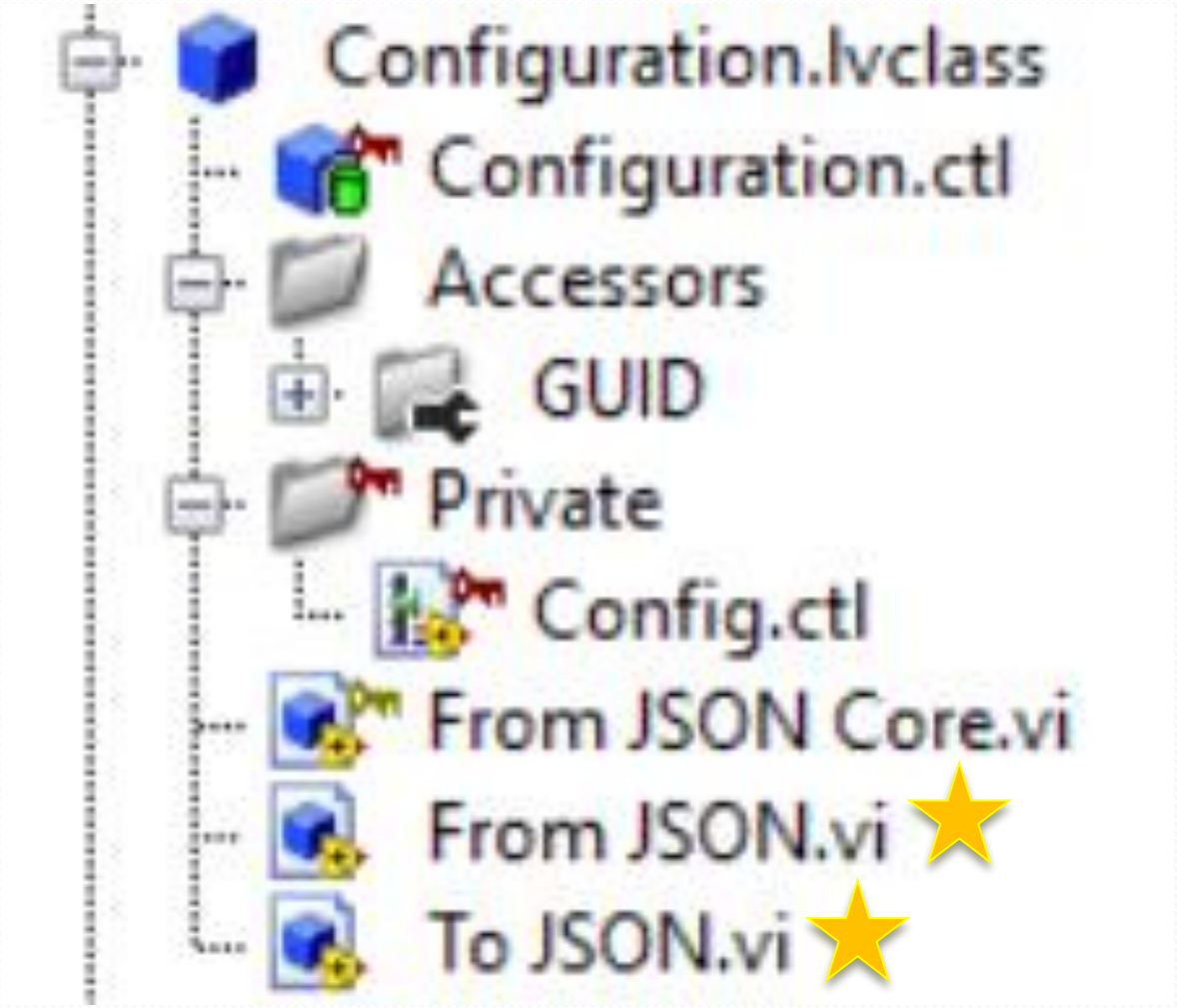
Where do we get the Thermometer objects from?

→ We need to let the user configure them and save/load to/from disk


Operation
Serialization of Persistent Data
Other inheritance hierarchy
Configurable (*interface*)
Thermometer
has
Modbus Thermometer
I2C Thermometer
Configuration (*class*)
Thermometer Config
Modbus Thermometer Config
I2C Thermometer Config

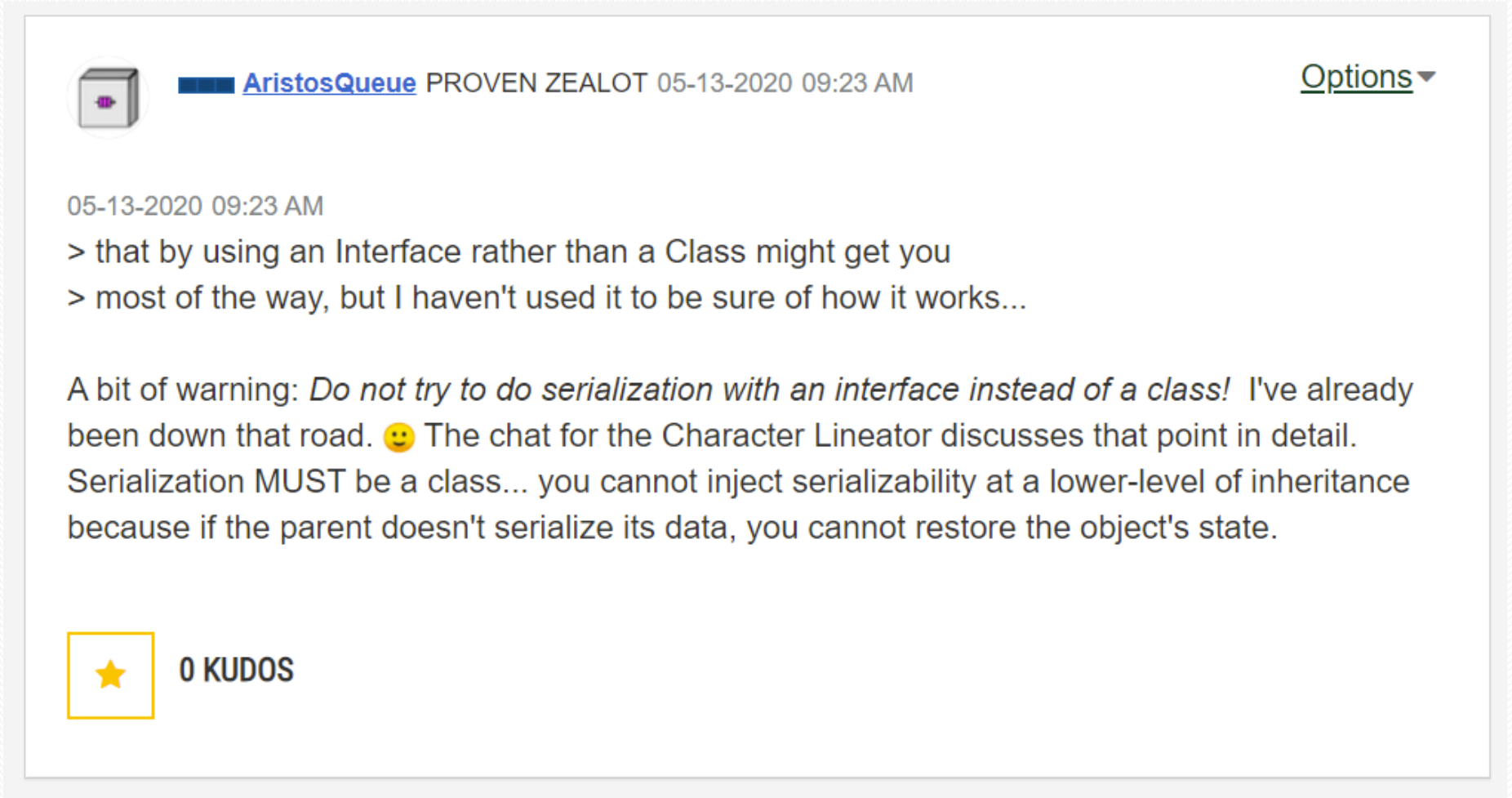
Configurable

Configuration



Why is Configuration a separate class?

And why isn't it an interface?



AristosQueue PROVEN ZEALOT 05-13-2020 09:23 AM

Options

05-13-2020 09:23 AM

> that by using an Interface rather than a Class might get you
> most of the way, but I haven't used it to be sure of how it works...

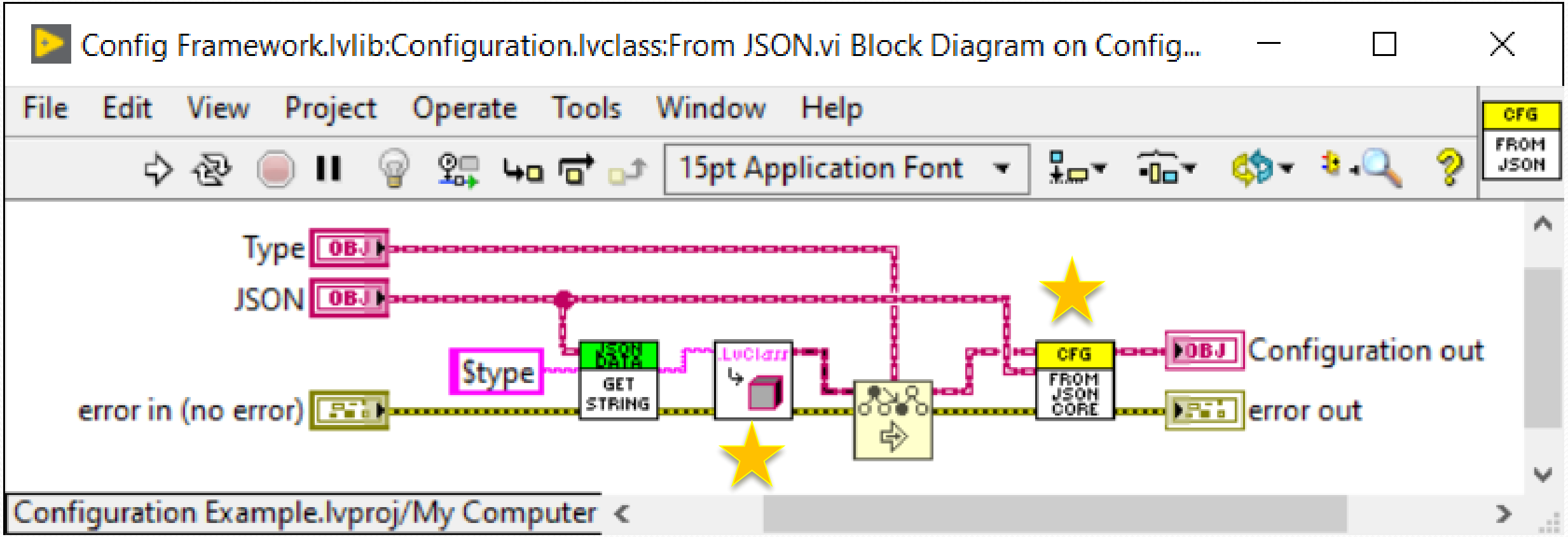
A bit of warning: *Do not try to do serialization with an interface instead of a class!* I've already been down that road. The chat for the Character Lineator discusses that point in detail. Serialization MUST be a class... you cannot inject serializability at a lower-level of inheritance because if the parent doesn't serialize its data, you cannot restore the object's state.

0 KUDOS

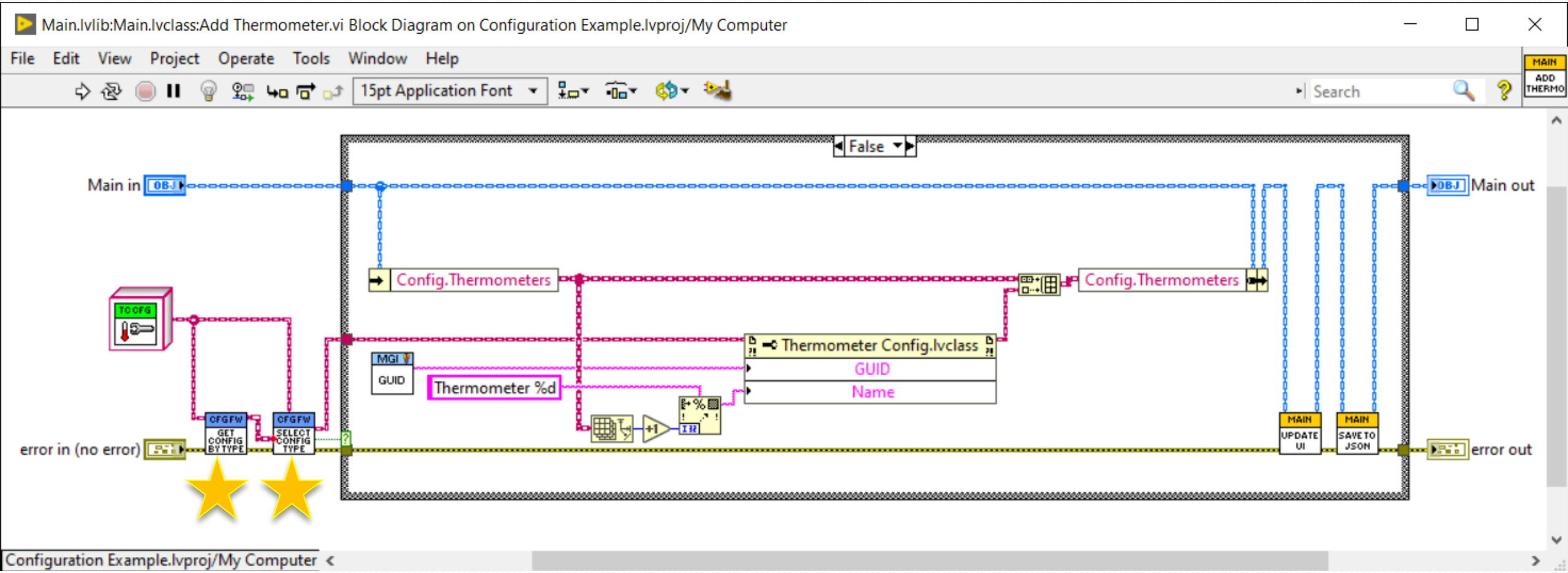
Example of JSON serialization

```
{
  "$type": “Modbus Thermometer Config.lvclass",
  "Configuration": {
    "$version": "1.0",
    "GUID": "40EC99C1-7A22-4234-A181-A5978E61B0FA"
  },
  “Modbus Thermometer": {
    "$version": "1.0",
    "IP Address": “127.0.0.1"
  },
  "Thermometer": {
    "$version": "1.1",
    "Color": 16711680,
    "Name": "Thermometer 1"
}
```

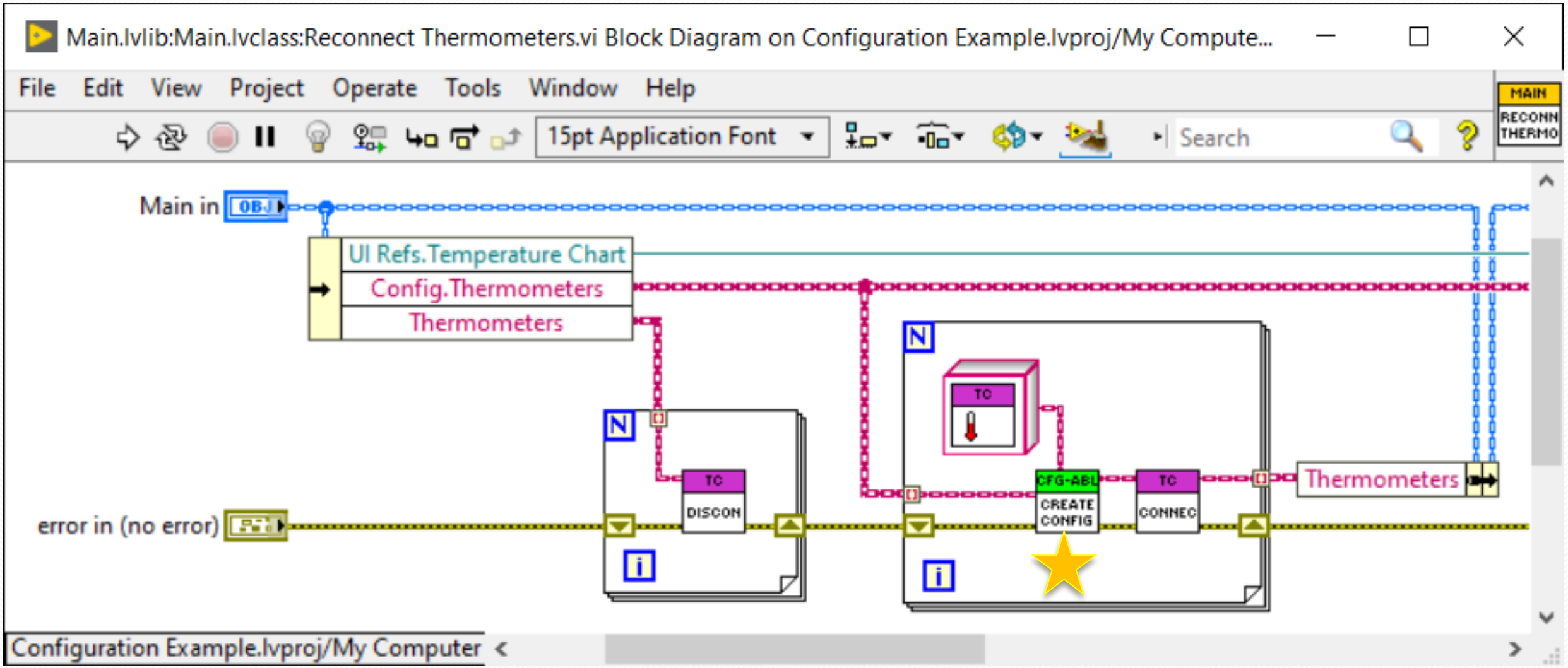
Configuration



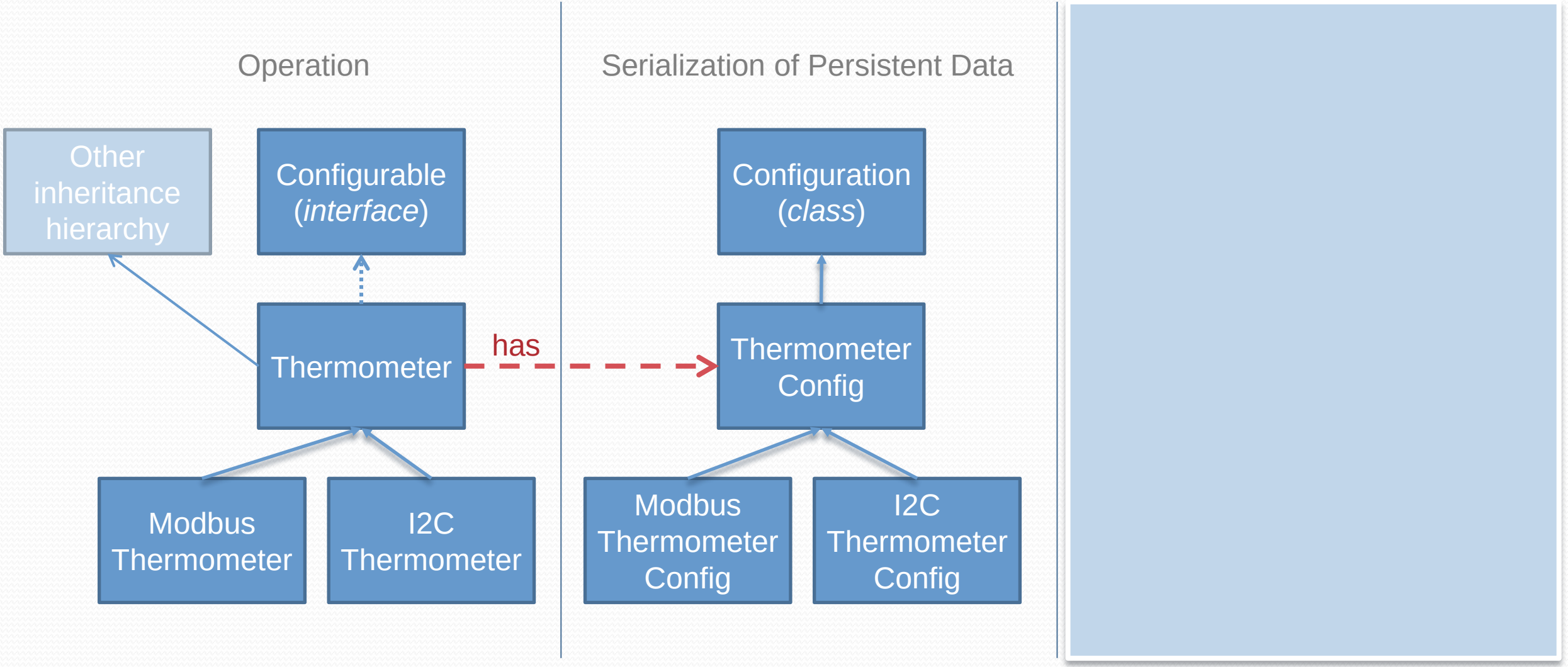
How to create a configuration



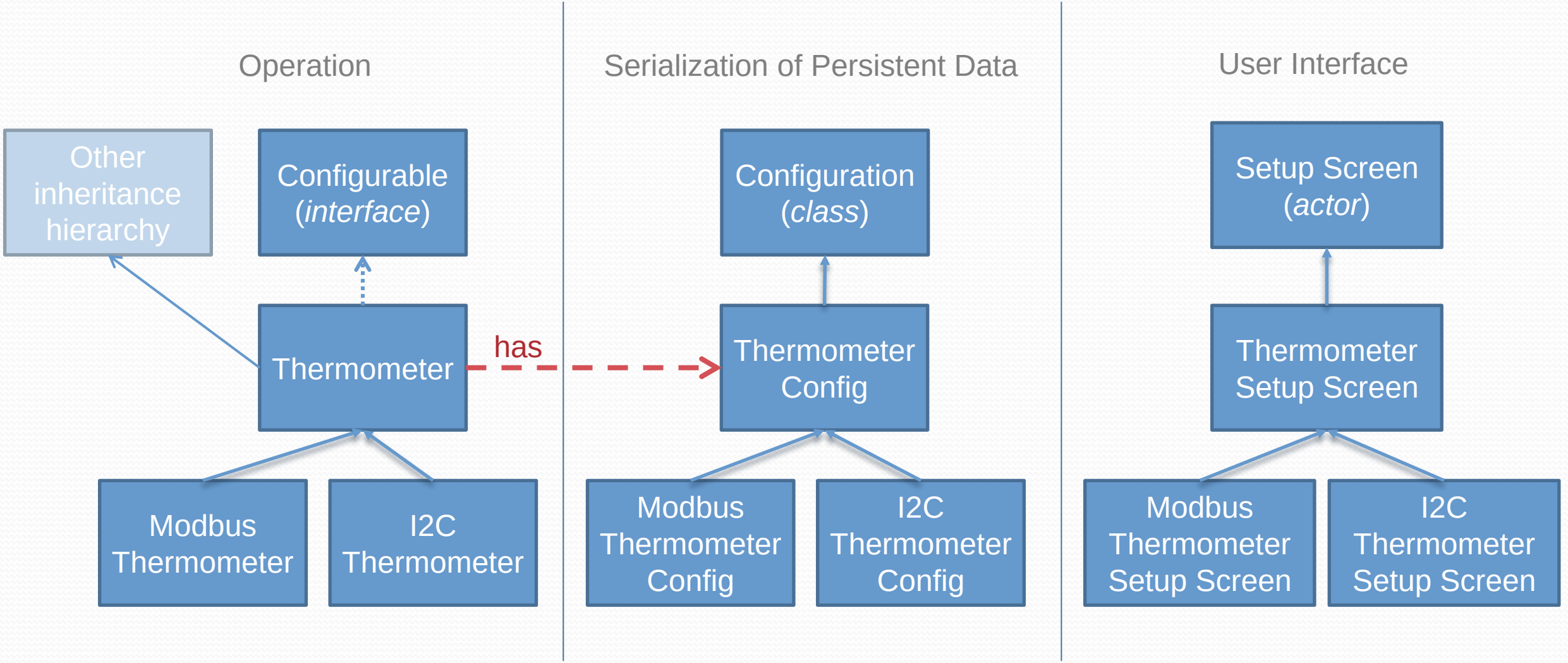
How to create a configurable



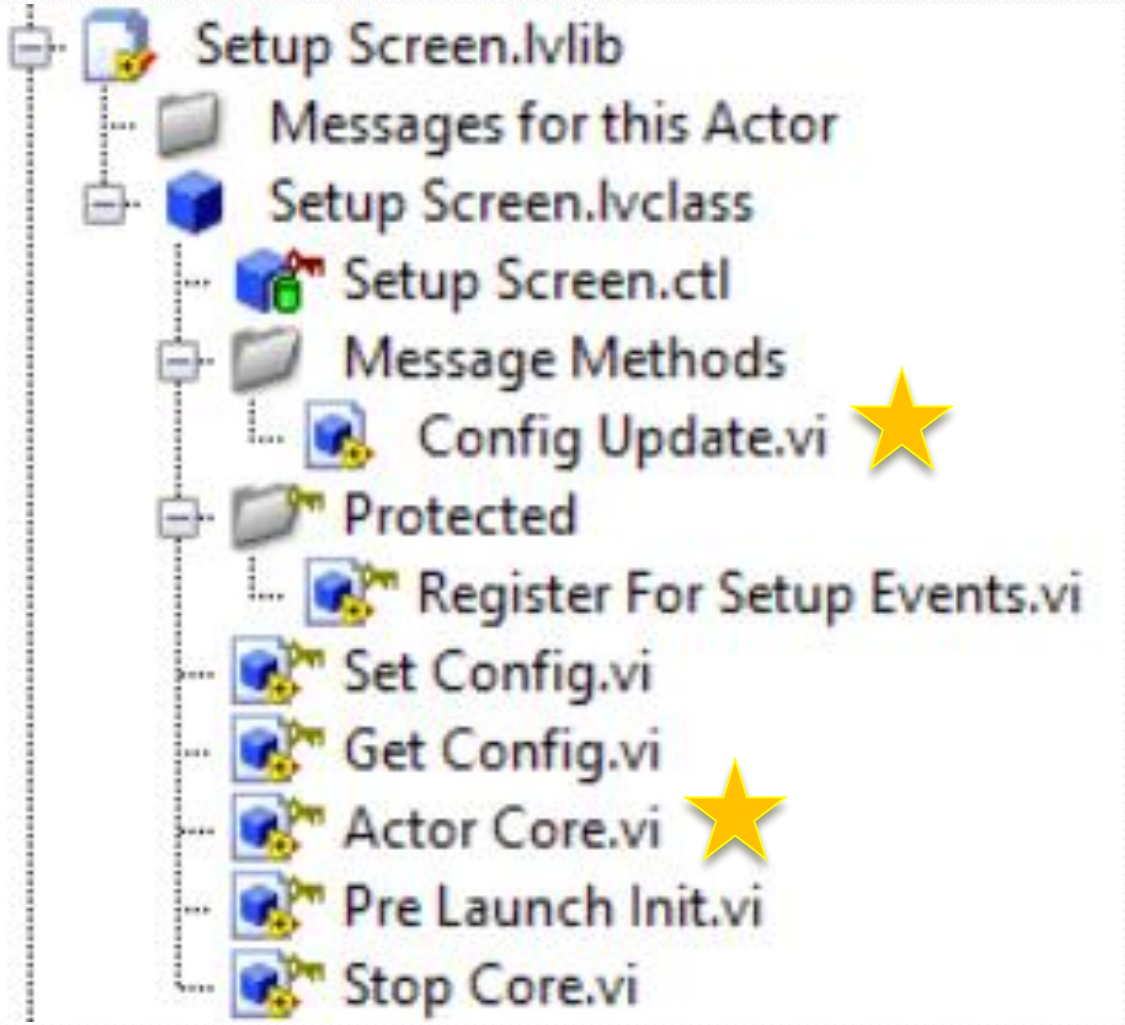
Questions so far?



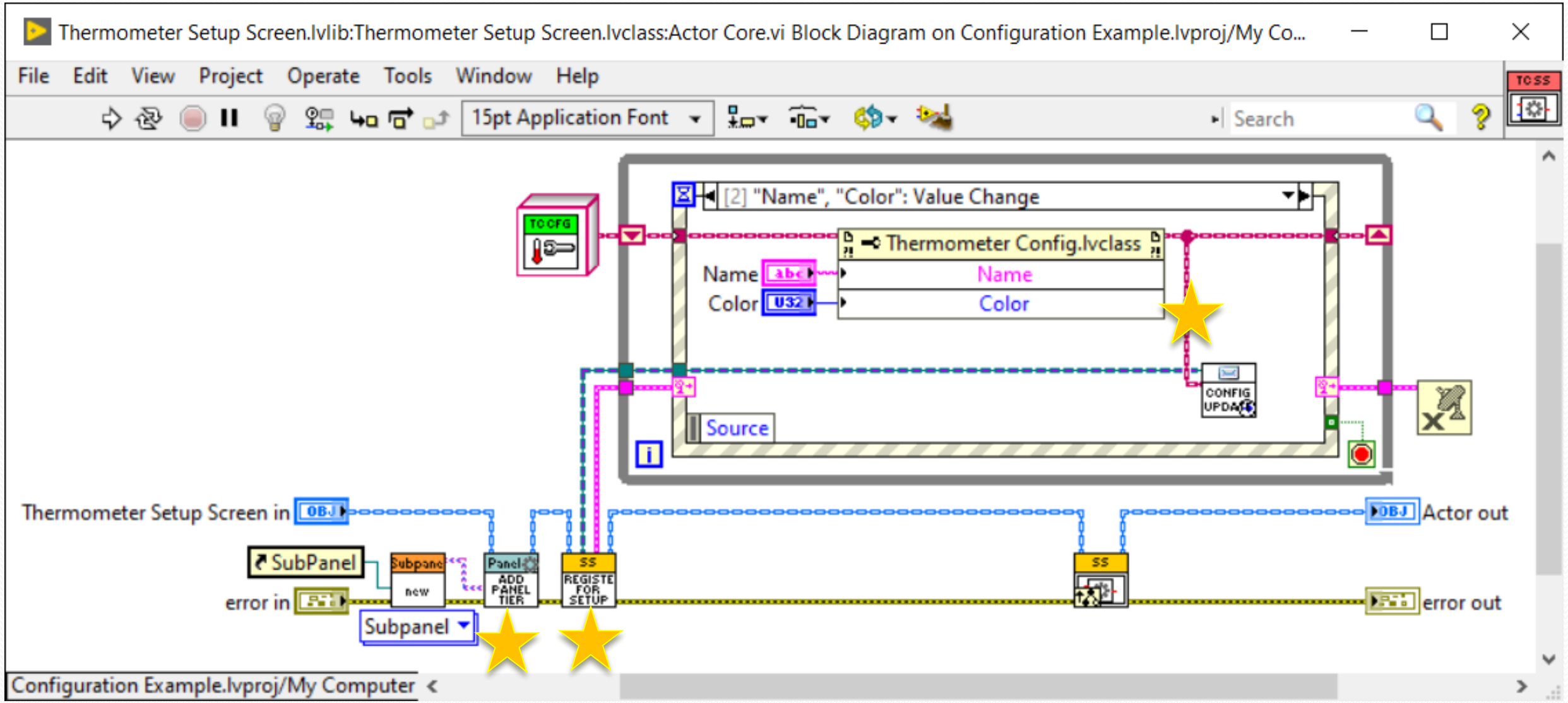
Questions so far?



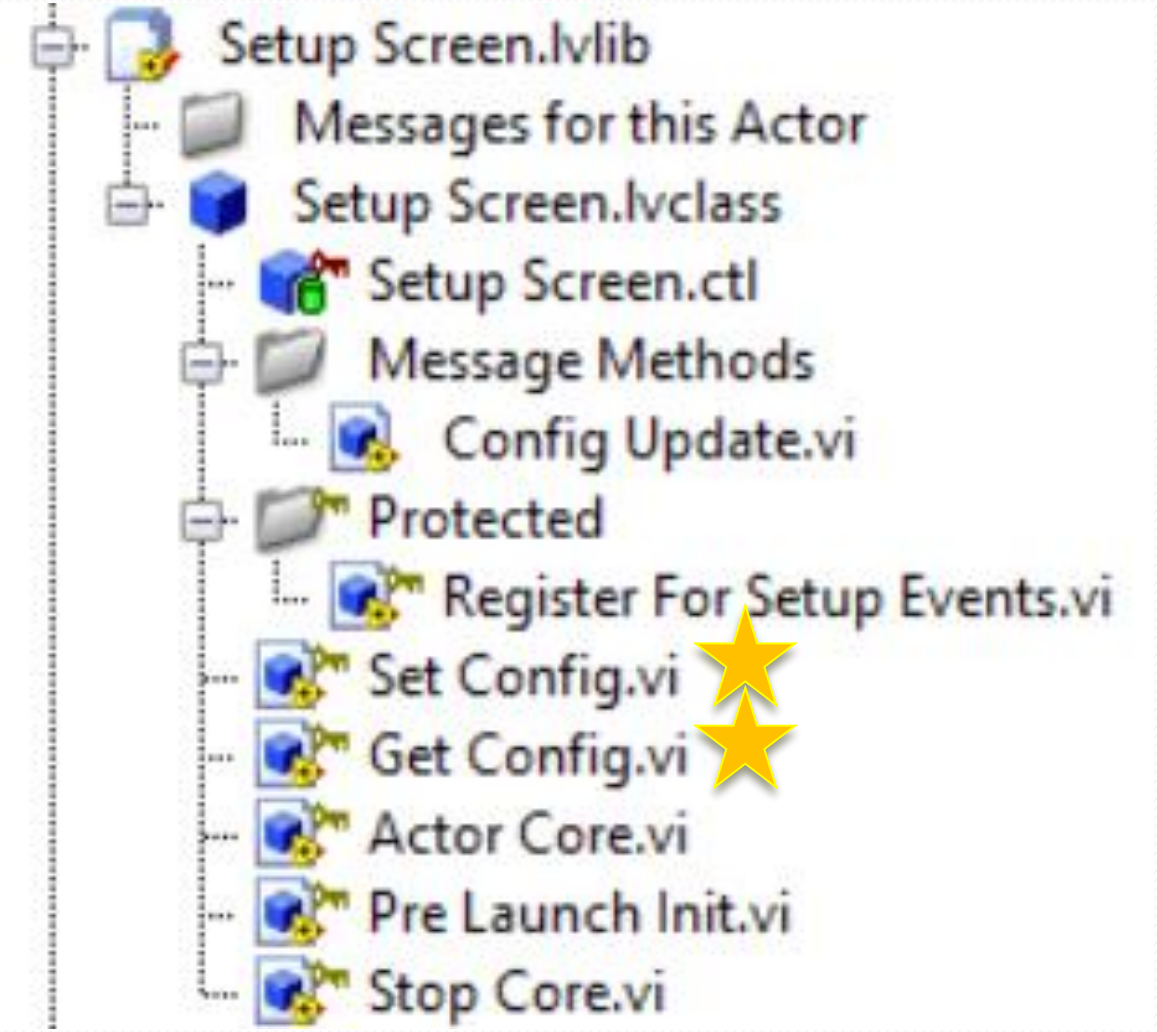
Setup Screen



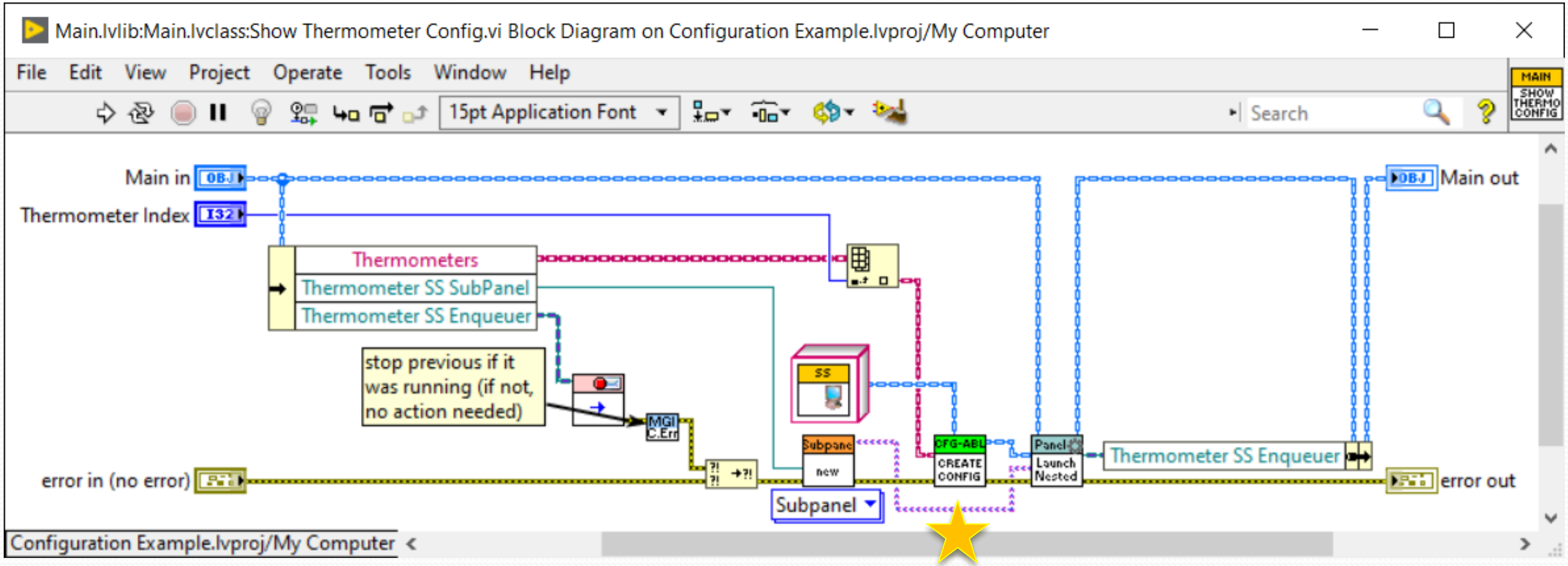
Setup Screen



Setup Screen is actually a Configurable!



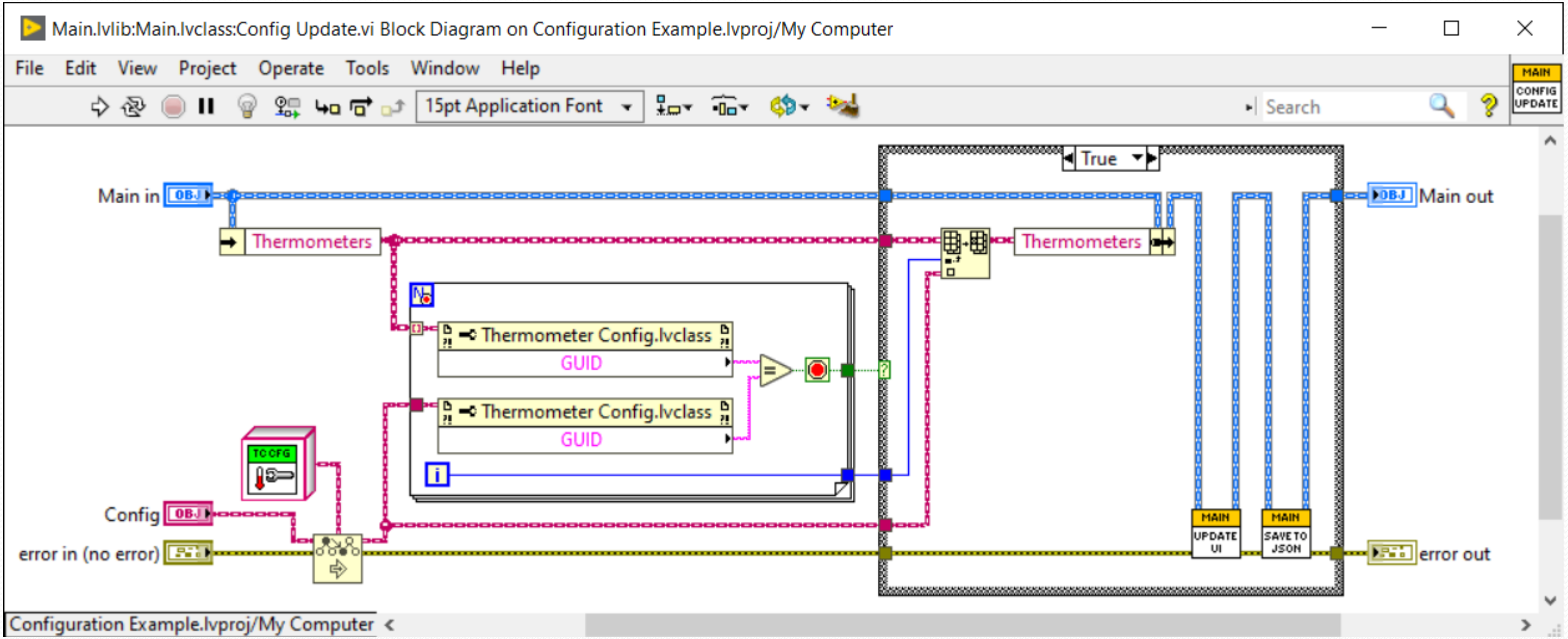
How to edit configuration



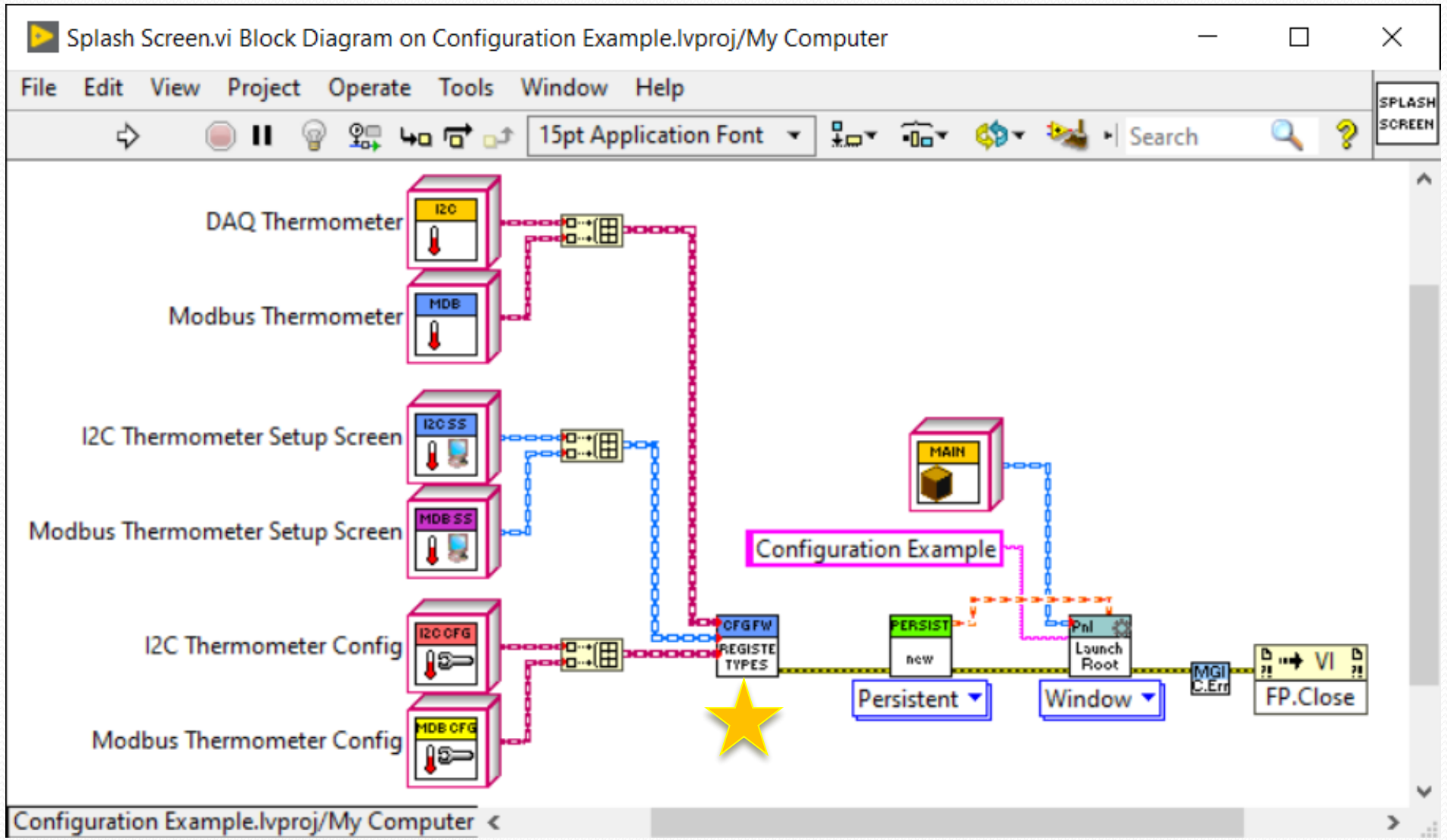
How to edit configuration

Inherit from the Config Update Reciever and override Config Update.vi in the nested caller and you will automatically get updates when the configuration changes

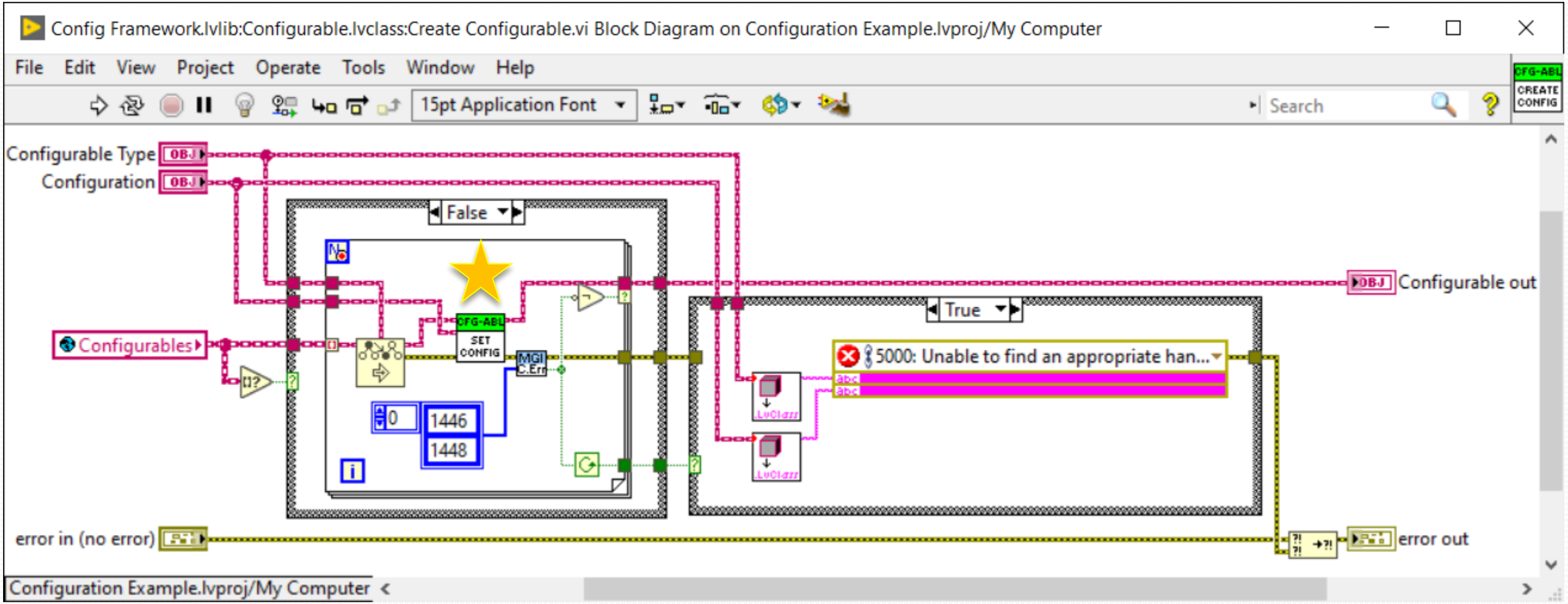
How to edit configuration



How it works - splash screen set up



How it works – Create Configurable



Multiple splash screens

You can also create multiple splash screens where you register more or less configurables. This way you can easily build multiple versions of your product, for example a cheaper one with less features and a more expensive one with more features.