This sample demonstrates using composition notifications on components. It's an extension to the previous basic sample, Configuration Variable .

Project name: "G.Notification"

For information on how to get the code, and run the sample, please see About Basic Samples .

Description

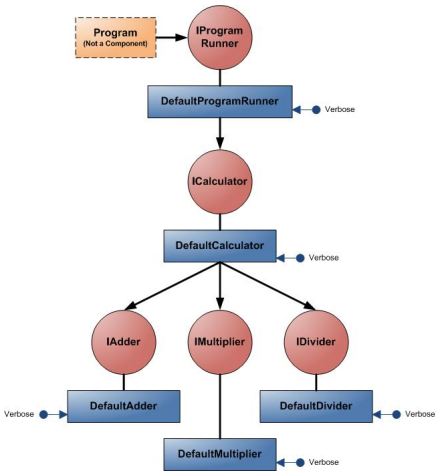
The functionality and composition of this sample is exactly the same as the previous sample . Only composition notification methods are added to the component implementations.

Composition notification methods, are methods in the components marked with the [OnCompositionComplete] attribute. The only thing that happens in these methods in the sample, is to print out a message on the console, so that you can see the order of events during composition of components.

In real software systems, these methods can be used to perform further initialization of the component that is not possible in the constructor, because the component plugs and configuration points are not set.

Note that you can have as many methods as you want which are marked with [OnCompositionComplete], and Composer will call them all. An alternative way is to implement an interface in your component.

Dependency Diagram



Sample output

CONSTRUCTOR - DefaultProgramRunner
CONSTRUCTOR - DefaultCalculator
CONSTRUCTOR - DefaultAdder
SET CONFIG - DefaultAdder.Verbose
NOTIFICATION - DefaultAdder: OnCompositionComplete.
SET PLUG - DefaultCalculator.Adder
CONSTRUCTOR - DefaultMultiplier
SET CONFIG - DefaultMultiplier.Verbose
NOTIFICATION - DefaultMultiplier: OnCompositionComplete.
SET PLUG - DefaultCalculator.Multiplier
CONSTRUCTOR - DefaultDivider
SET CONFIG - DefaultDivider.Verbose
NOTIFICATION - DefaultDivider: OnCompositionComplete.
SET PLUG - DefaultCalculator.Divider
SET CONFIG - DefaultCalculator.Verbose
NOTIFICATION - DefaultCalculator: OnCompositionComplete.
SET PLUG - DefaultProgramRunner.Calculator
SET CONFIG - DefaultProgramRunner.Verbose
NOTIFICATION - DefaultProgramRunner: OnCompositionComplete. METHOD CALL - DefaultProgramRunner.Run() METHOD CALL - DefaultCalculator.Add(67, 12)
METHOD CALL - DefaultAdder.Add(67, 12)
67 + 12 = 79 METHOD CALL - DefaultCalculator.Subtract(67, 12)
METHOD CALL - DefaultAdder.Add(67, -12)
67 - 12 = 55 METHOD CALL - DefaultCalculator.Multiply(67, 12)
METHOD CALL - DefaultMultiplier.Multiply(67, 12)
67 * 12 = 804 METHOD CALL - DefaultCalculator.Divide(67, 12)
METHOD CALL - DefaultDivider.Divide(67, 12)
METHOD CALL - DefaultCalculator.Remainder(67, 12)
METHOD CALL - DefaultDivider.Remainder(67, 12)
67 / 12 = 5 (with remainder = 7)