STATE PATTERN

SIMPLE CALCULATOR EXAMPLE

Marina Ionel

ALLOW AN OBJECT TO ALTER ITS BEHAVIOR WHEN ITS INTERNAL STATE CHANGES. THE OBJECT WILL APPEAR TO CHANGE ITS CLASS.

”

INTENT OF THE STATE PATTERN

(Gamma et al., 1995)

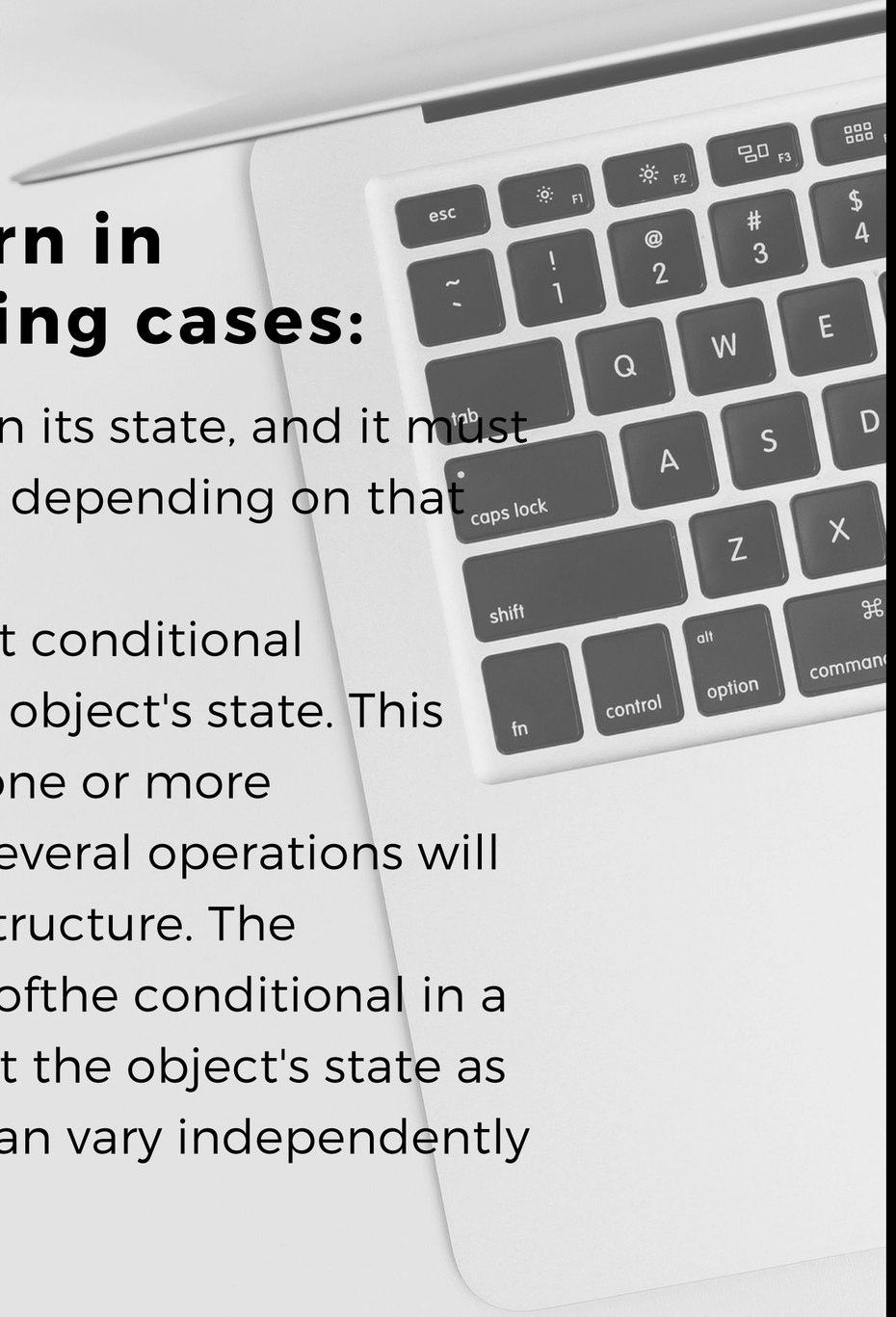
Applicability

Use the State pattern in either of the following cases:

An object's behavior depends on its state, and it must change its behavior at run-time depending on that state.

Operations have large, multipart conditional statements that depend on the object's state. This state is usually represented by one or more enumerated constants. Often, several operations will contain this same conditional structure. The Statepattern puts each branch ofthe conditional in a separate class. This lets you treat the object's state as an object in its own right that can vary independently from other objects.

(Gamma et al., 1995)



Consequences

IT LOCALIZES STATE-SPECIFIC BEHAVIOR AND PARTITIONS BEHAVIOR FOR DIFFERENT STATES

IT MAKES STATE TRANSITIONS EXPLICIT

STATE OBJECTS CAN BE SHARED

Simple Calculator

COMBINING STATE AND SINGLETON DESIGN PATTERNS

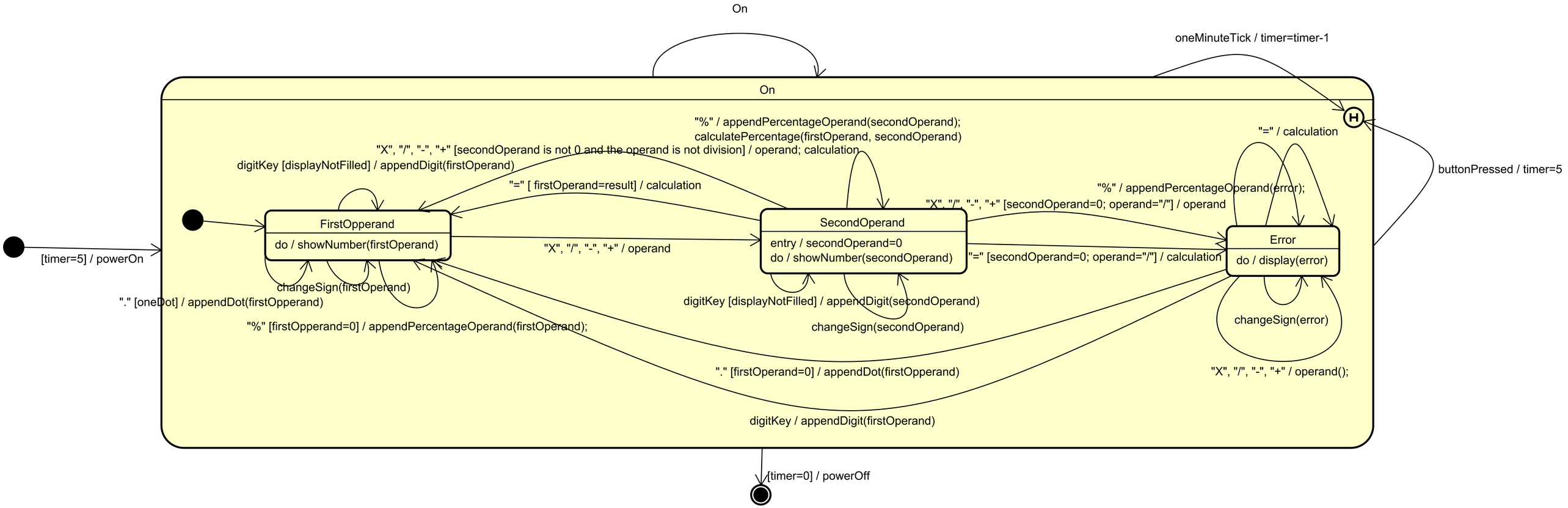
ENSURE A CLASS ONLY HAS ONE INSTANCE, AND PROVIDE A GLOBAL POINT OF ACCESS TO IT.

”

INTENT OF THE SINGLETON PATTERN

(Gamma et al., 1995)

State Machine Diagram



NOTE

Another way of transiting to the Error state (that is not indicated in the diagram) is by getting a result wich has more than 10 numbers: overflow.

Comments

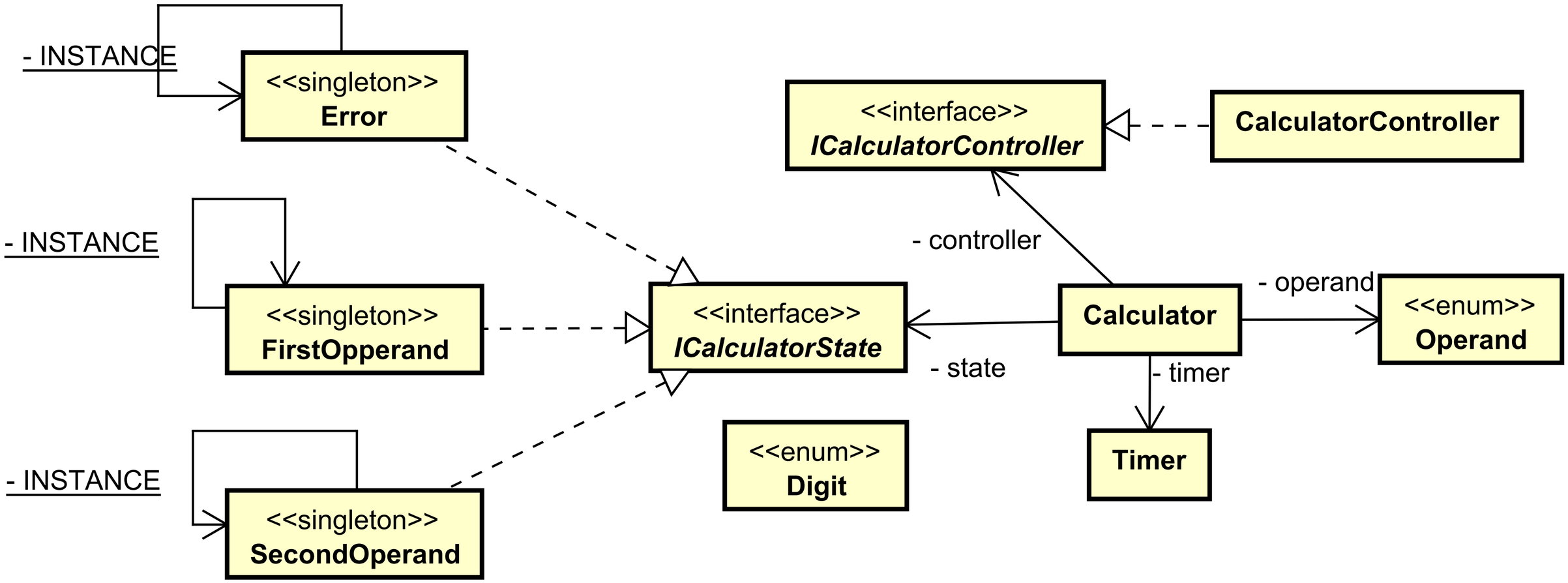
STATES

The Diagram presents one suprastate "On" and the states "First Operand", "Second Operand" and "Error".

HISTORY

The shallow history pseudostate represents the timer.

Class Diagram



COMMENTS

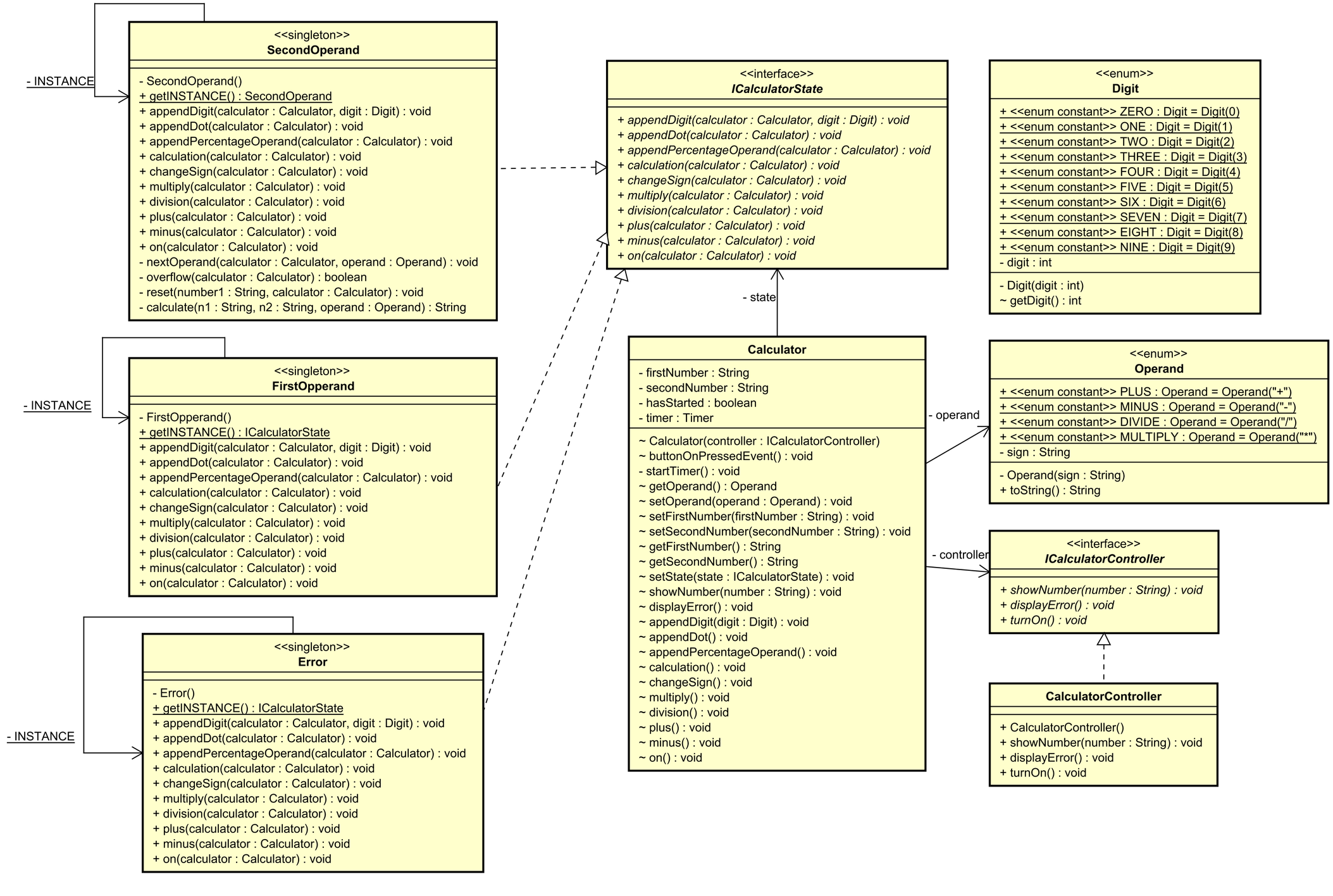
As can be seen in the diagram, the states (First Operand, Second Operand and Error) are singletons and implement the intreface **ICalculatorState.**

Digits and basic operands (+, -, /, *) are respresented as enums.

The calculator has a Timer field (java.util.timer). This is the representation of the timer indicated in the state machine diagram.

The controller interface and class represent the connection to the hardware.

Detailed Class Diagram

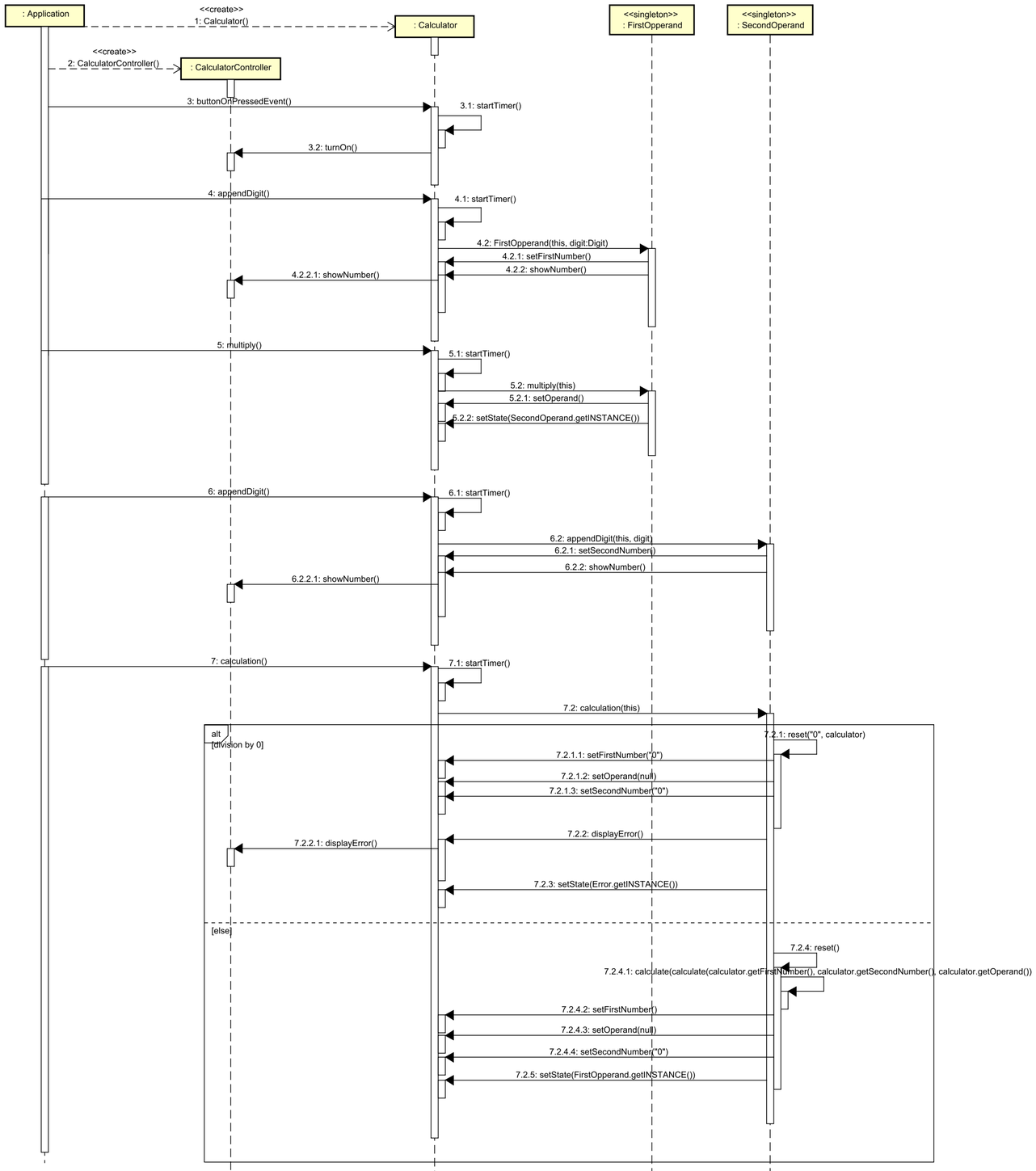


COMMENTS

The Calculator class represents the context in the state pattern and it is mainly package private because in this simple example everything is the same package, so there is no need of making this class public. As can be seen, this class depends on abstractions: ICalculatorState, ICalculatorController (Dependency inversion principle).

The Enum Digit represents the buttons with digits. This solution simulates as close as possible the real calculator, rather than using integers as a parameter for the methods.

Sequence Diagram



SOLID PRINCIPLES

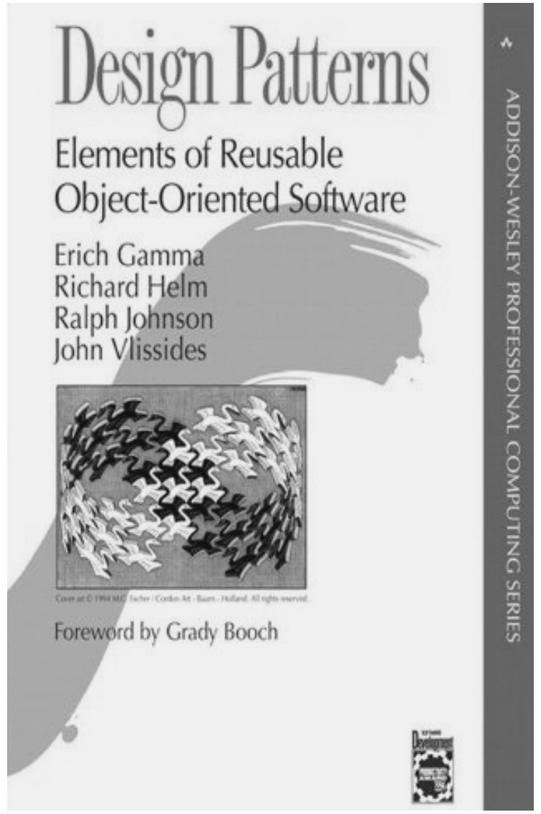
SRP	OCP	LSP	ISP	DIP
Single responsibility principle	Open/Closed Principle	Liskov substitution principle	Interface Segregation Principle	Dependency inversion principle

SOLID principles

Are SOLID principles violated?

"I cannot provide an example of violation of the SOLID principles based on the calculator example, but I can say that methods like appendDigit in FirstOperand and SecondOperand, have the same functionality and just set a different field in the calculator class and this does not correspond to the DRY principle, but this is a result of the State design pattern."

References



DESIGN PATTERNS: ELEMENTS OF REUSABLE OBJECT-ORIENTED SOFTWARE

by Erich Gamma, John Vlissides, Ralph Johnson, and Richard Helm