Visual Paradigm

Singleton Pattern Tutorial

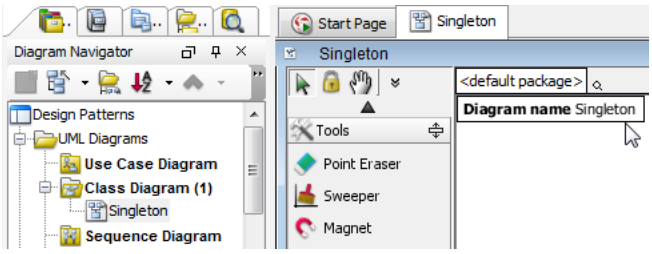
Written Date : September 30, 2009

This tutorial is aimed to guide the definition and application of Gang of Four (GoF) singleton design pattern. By reading this tutorial, you will know how to develop a model for the signletion pattern, and how to apply it in practice.

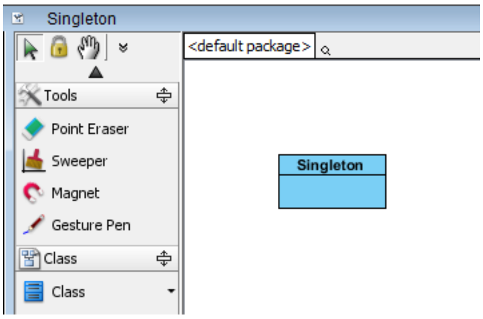
Modeling Design Pattern with Class Diagram

1. Create a new project *Design Patterns*.

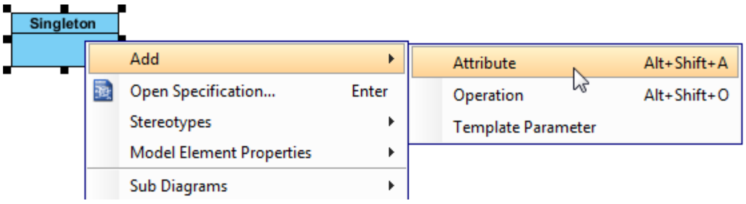
2. Create a class diagram *Singleton*.



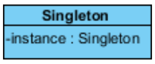
3. Select **Class** from diagram toolbar. Click on the diagram to create a class. Name it as *Singleton*.



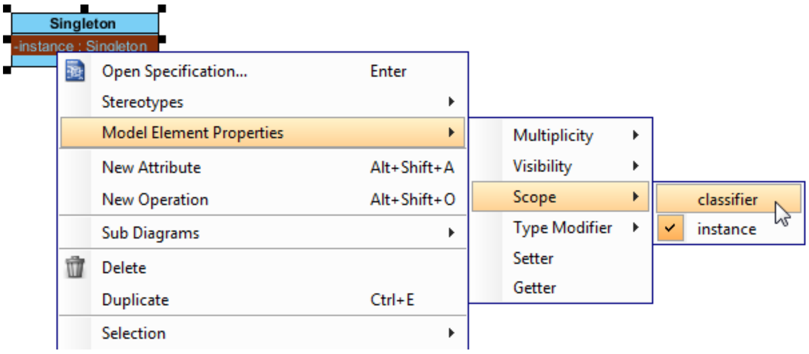
4. Right-click on the *Singleton* class and select **Add** > **Attribute** from the popup menu.



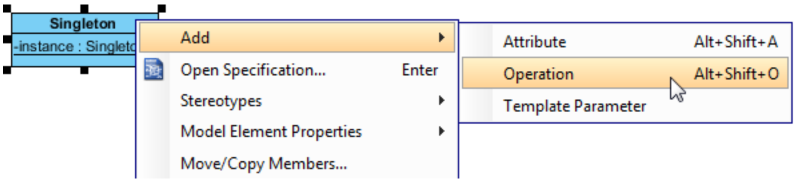
5. Name the attribute *instance*. Set its type as *Singleton*.



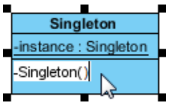
6. The attribute instance need to be static. Right-click on the attribut and select **Model Element Properties** > **Scope** > **Classifier** from the popup menu.



7. Create constructor for the *Singleton* class. Right-click on *Singleton* and select **Add** > **Operation** from the popup menu.

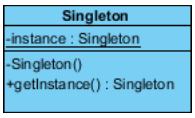


8. Name the operation *Singleton*, which follows the *Singleton* class's name. Change + to - in front of the operation name to indicate that this is a private constructor.

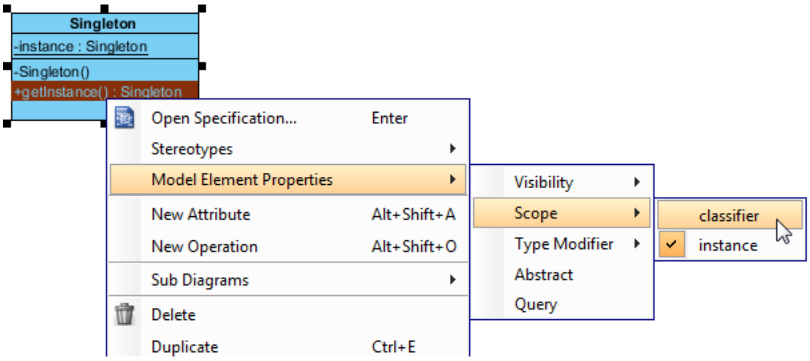


9. Right-click on *Singleton* and select **Add** > **Operation** from the popup menu.

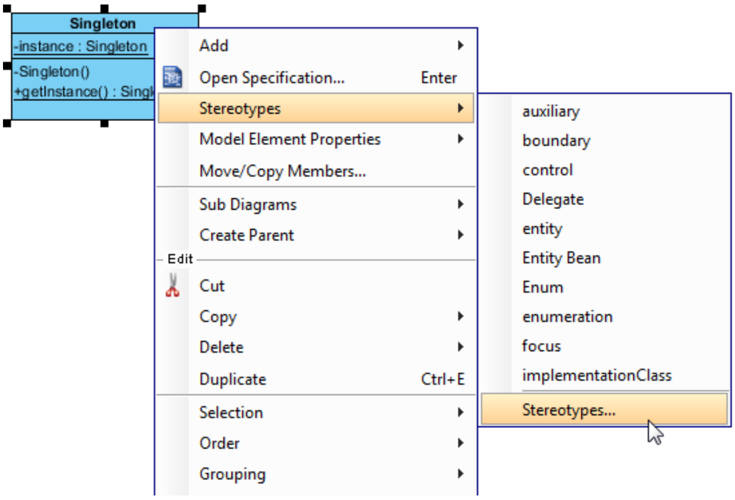
10. Name the operation *getInstance*, and make it return *Singleton*.



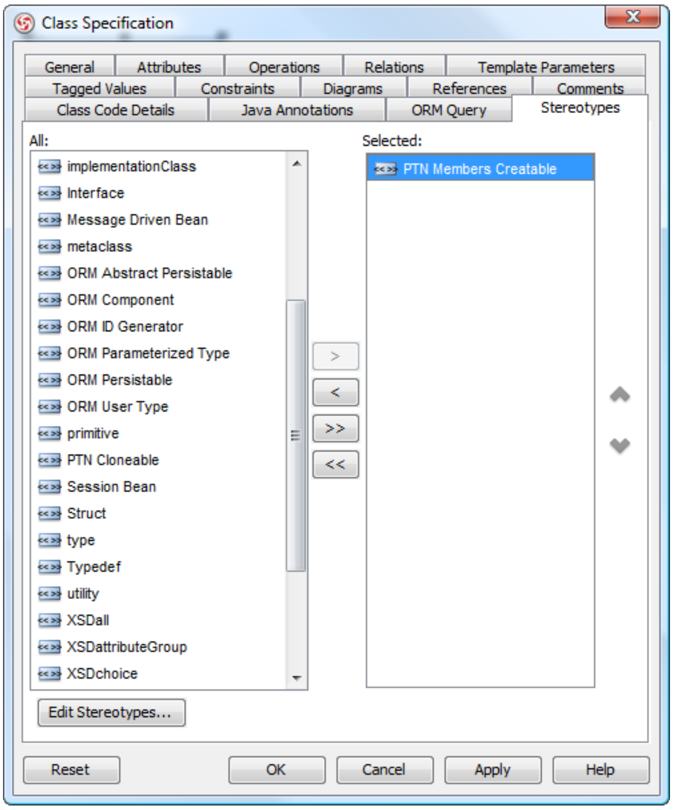
11. The operation *getInstance* need to be static. Right-click on the operation and select **Model Element Properties** > **Scope** > **Classifier** from the popup menu.



12. In practice, there may be operations for accessing data in the Singleton class. To represent this, stereotype the *Singleton* class as PTN Members Creatable. Right-click on the *Singleton* class and select **Stereotypes** > **Stereotypes...** from the popup menu.



13. In the class specification dialog box, select **PTN Members Creatable** and click > to assign it. Click **OK** to confirm.



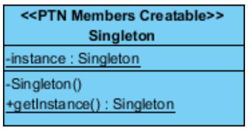
All:

- implementationClass
- Interface
- Message Driven Bean
- metaclass
- ORM Abstract Persistable
- ORM Component
- ORM ID Generator
- ORM Parameterized Type
- ORM Persistable
- ORM User Type
- primitive
- PTN Cloneable
- Session Bean
- Struct
- type
- Typedef
- utility
- XSDall
- XSDattributeGroup
- XSDchoice

Selected:

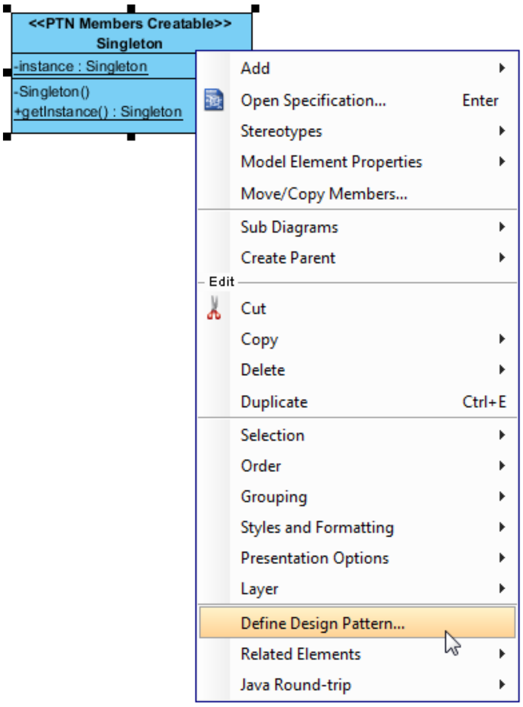
- PTN Members Creatable

Up to now, the diagram should look like this:

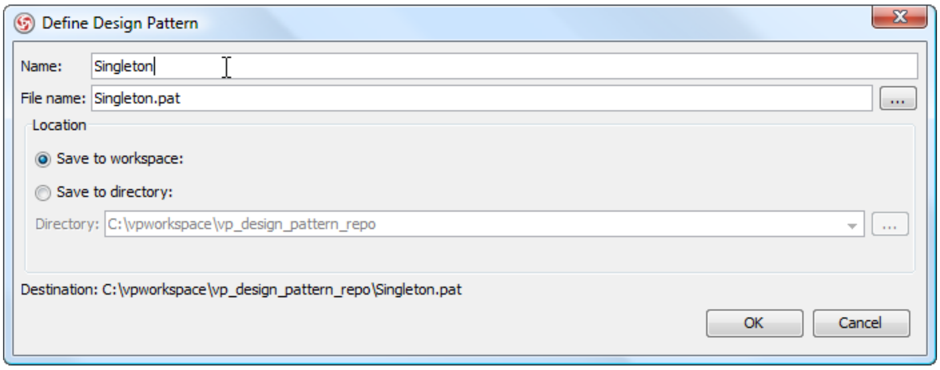


Defining Pattern

1. Right-click on the Singleton class and select **Define Design Pattern...** from the popup menu.



2. In the **Define Design Pattern** dialog box, specify the pattern name *Singleton*. Keep the file name as is. Click **OK** to proceed.


Define Design Pattern

Name: Singleton

File name: Singleton.pat

Location

● Save to workspace:

○ Save to directory:

Directory: C:\vpworkspace\vp_design_pattern_repo

Destination: C:\vpworkspace\vp_design_pattern_repo\Singleton.pat

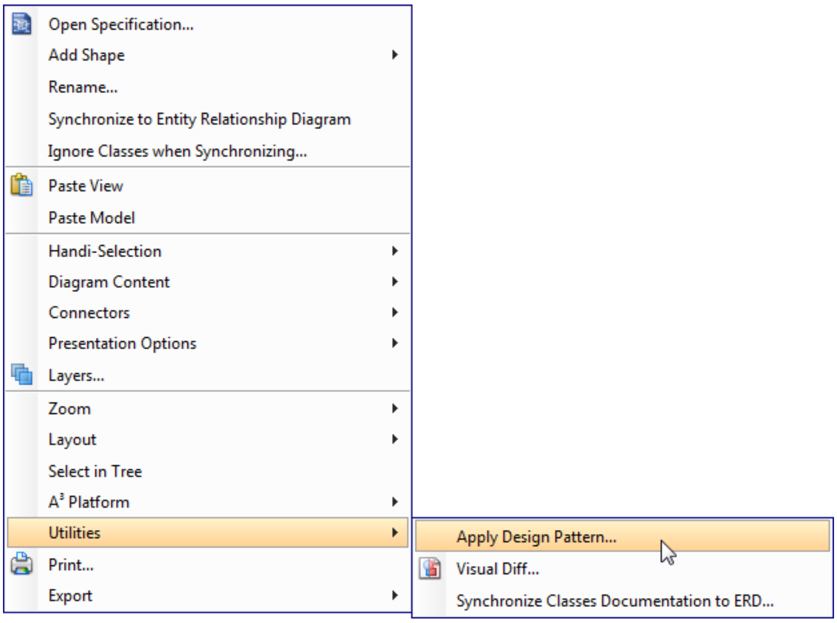
Applying Design Pattern on Class Diagram

In this section, we are going to apply the singleton pattern in modeling a class registry.

1. Create a new project *Class Registry*.

2. Create a class diagram *The Registry*.

3. Right-click on the class diagram and select **Utilities** > **Apply Design Pattern...** from the popup menu.



4. In the **Design Pattern** dialog box, select *Singleton* from the list of patterns.

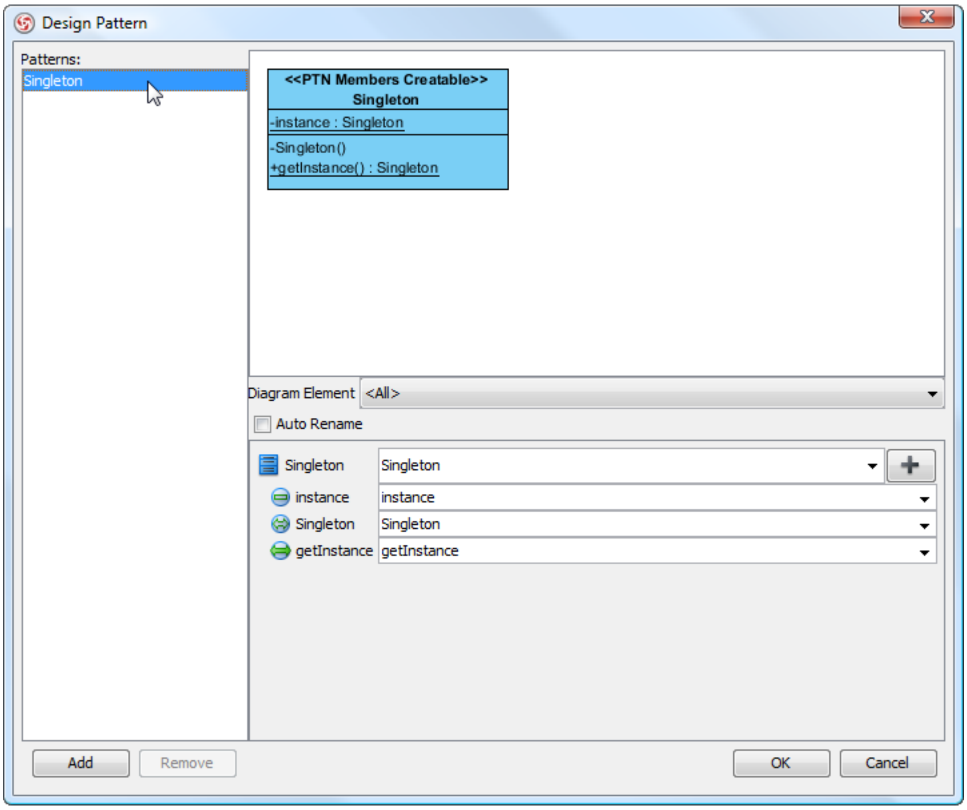
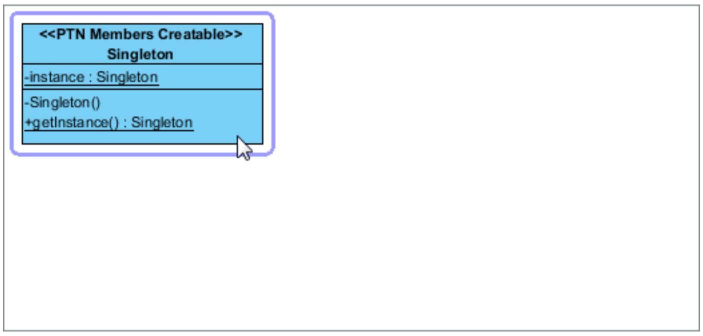


Diagram Element <All>

□ Auto Rename

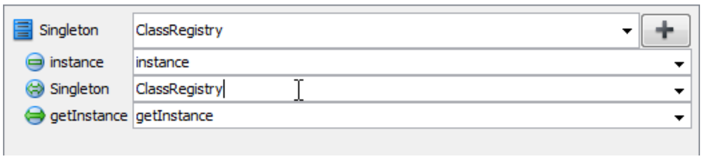
Singleton	Singleton
instance	instance
Singleton	Singleton
getInstance	getInstance

5. Click on *Singleton* in the ovewview.



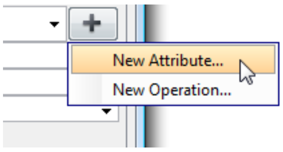
<<PTN Members Creatable>>
Singleton
-instance : Singleton
-Singleton()
+getInstance() : Singleton

6. Rename the class *Singleton*, as well as the constructor to *ClassRegistry* at the bottom pane.

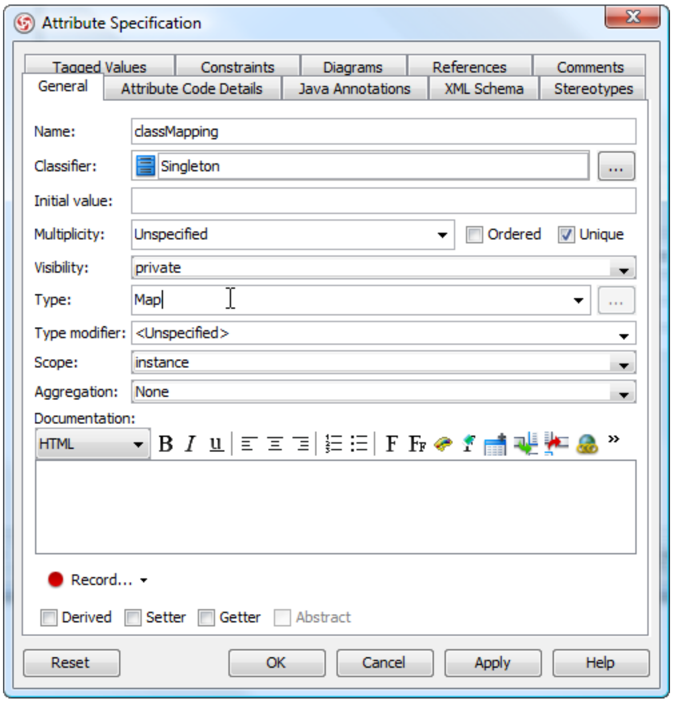


Singleton	ClassRegistry
instance	instance
Singleton	ClassRegistry
getInstance	getInstance

7. We need to add an attribute for holding the classes user register. Click on the + button and select **New Attribute...** from the popup menu.



8. In the **Attribute** Specification, enter *classMapping* as attribute name. Enter *Map* as type.



Name: classMapping

Classifier: Singleton

Initial value:

Multiplicity: Unspecified ☐ Ordered ☑ Unique

Visibility: private

Type: Map

Type modifier: <Unspecified>

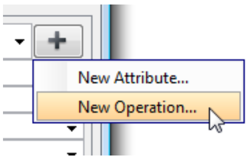
Scope: instance

Aggregation: None

Documentation: HTML

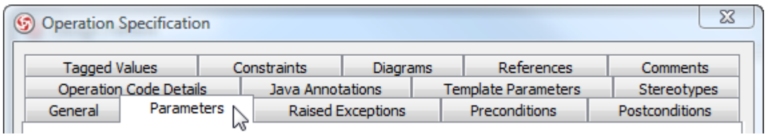
☐ Derived ☐ Setter ☐ Getter ☐ Abstract

9. We need to add operations for registering class and retrieving class by type. Click on the + button and select **New Operation...** from the popup menu.



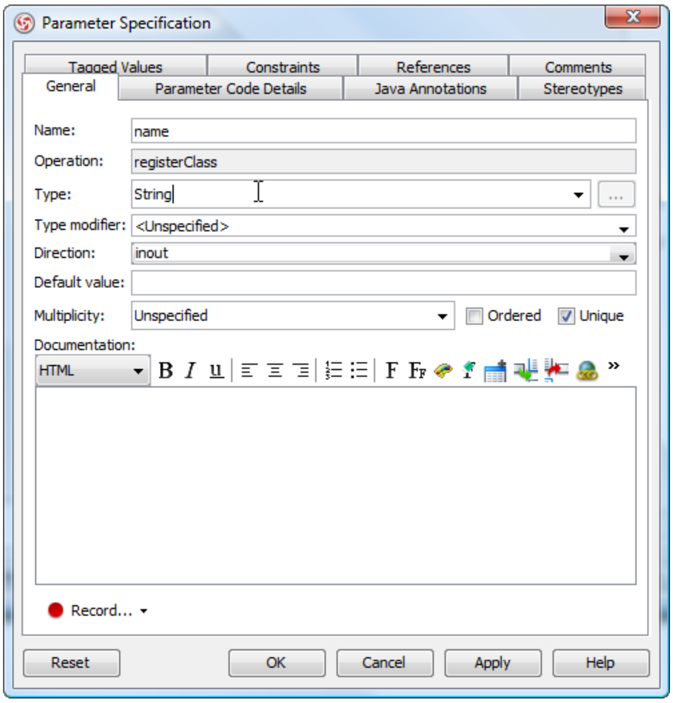
10. In the **Operation** Specification dialog box, enter *registerClass* as operation name.

11. Open the **Parameters** tab.



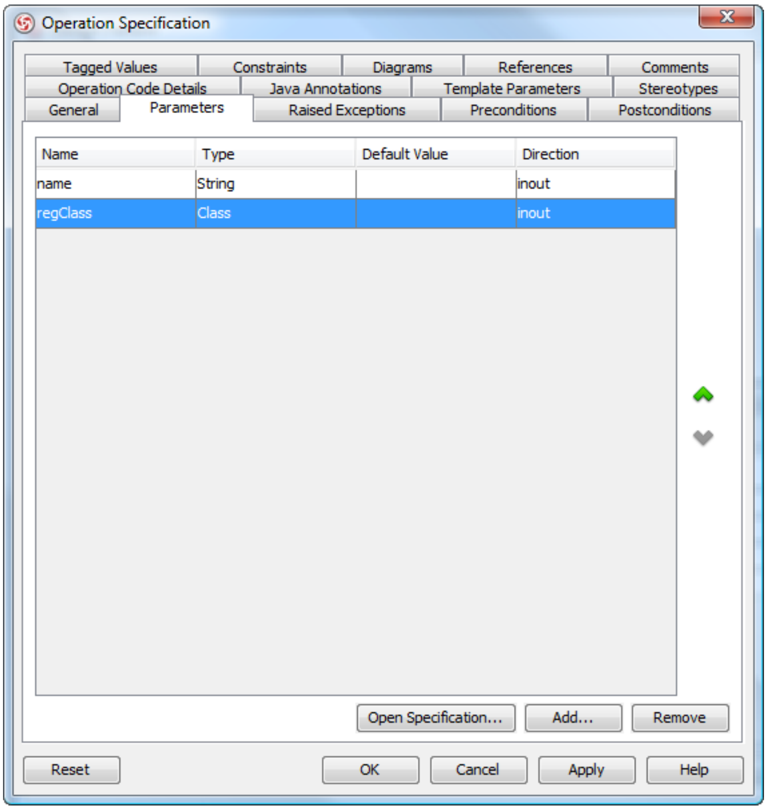
12. Click **Add...** at the bottom of specification dialog box.

13. In the **Parameter** Specification dialog box, enter *name* as parameter name and set *String* as type.



Name: name
Operation: registerClass
Type: String
Type modifier: <Unspecified>
Direction: inout
Default value:
Multiplicity: Unspecified ☐ Ordered ☑ Unique

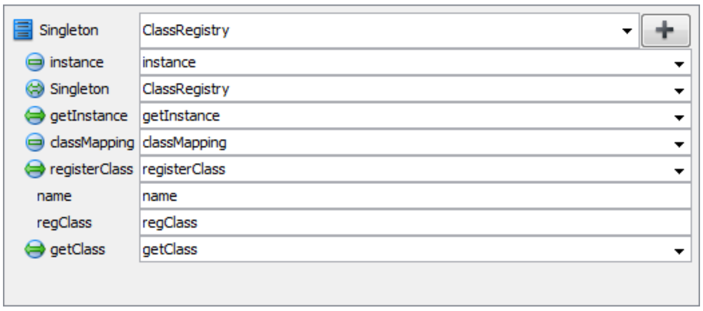
14. Repeat steps 12 and 13 to add parameter *regClass*, and set *Class* as type. Click **OK** to confirm.



Name	Type	Default Value	Direction
name	String		inout
regClass	Class		inout

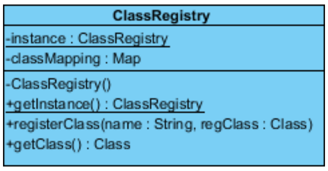
15. Click on the + button and select **New Operation...** from the popup menu.

16. In the **Operation Specification** dialog box, enter *getClass* as name, and set *Class* as return type.



Singleton	ClassRegistry
instance	instance
Singleton	ClassRegistry
getInstance	getInstance
classMapping	classMapping
registerClass	registerClass
name	name
regClass	regClass
getClass	getClass

17. Click **OK** to apply the pattern to diagram. This is the result:


ClassRegistry
-instance : ClassRegistry
-classMapping : Map
-ClassRegistry()
+getInstance() : ClassRegistry
+registerClass(name : String, regClass : Class)
+getClass() : Class

Resources

1. Design Patterns.vpp

2. Singleton.pat

Related Links

- Full set of UML tools and UML diagrams

Visual Paradigm home page
(https://www.visual-paradigm.com/)

Visual Paradigm tutorials
(https://www.visual-paradigm.com/tutorials/)