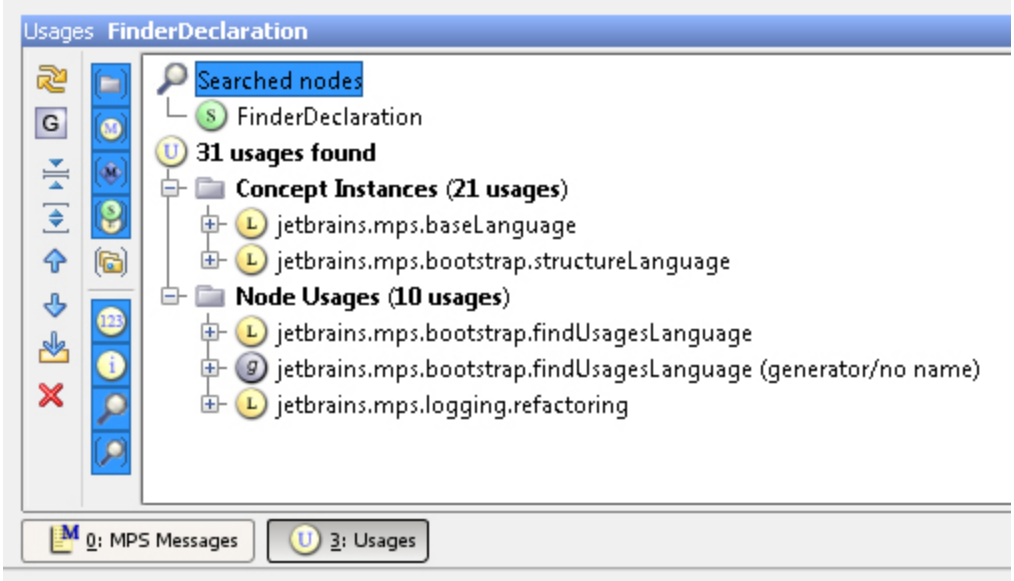
# Find usages

In MPS, any model consists of nodes. Nodes can have many types of relations. These relations may be expressed in a node structure (e.g. "class descendants" relation on classes) or not (e.g. "overriding method" relation on methods). Find Usages is a tool to display some specifically related nodes for a given node.

In MPS, the Find Usages system is fully customizable - you can write your own entities, so-called finders, which represent algorithms for finding related nodes. For every type of relation there is a corresponding finder.

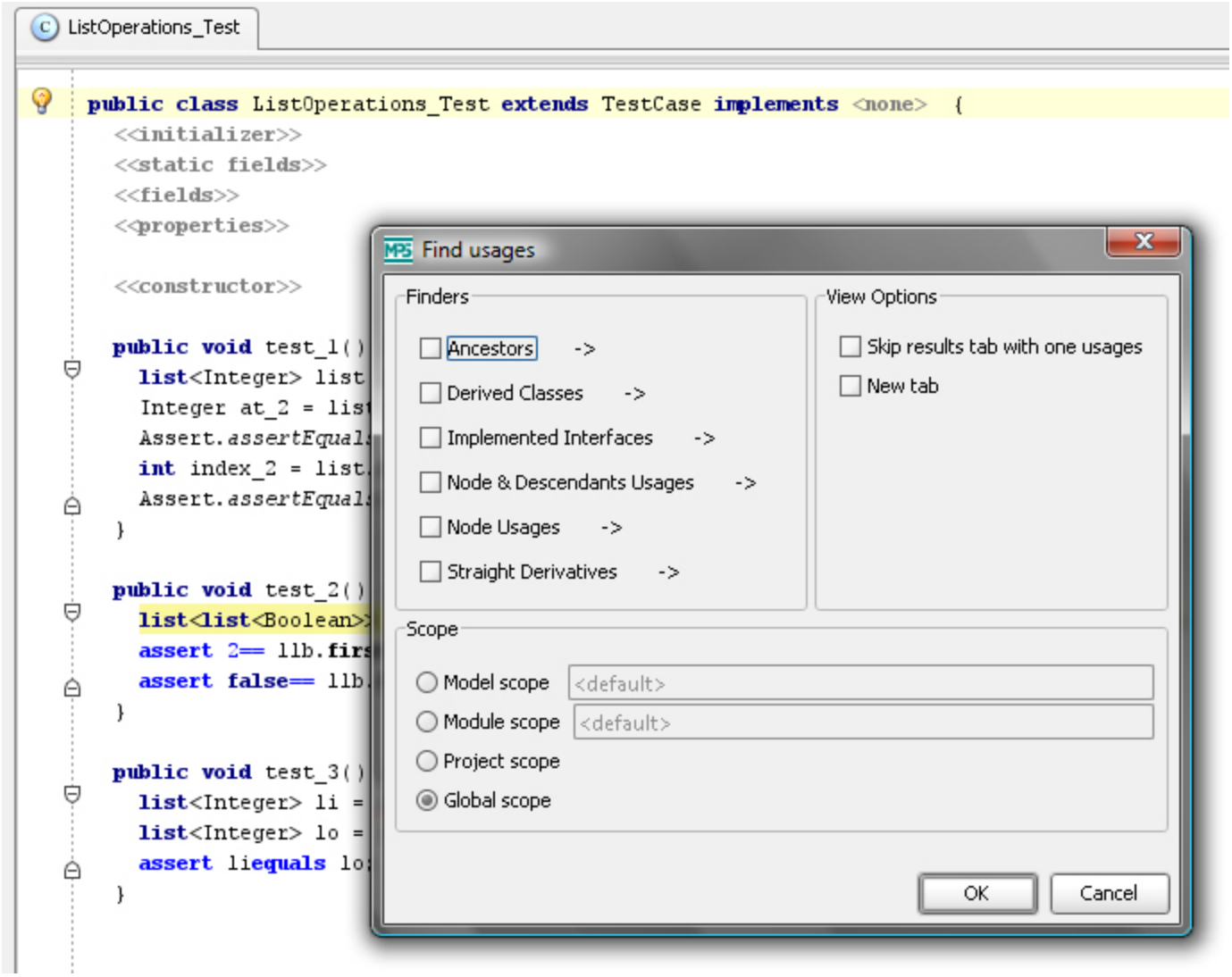
This is how "find usages" result looks like:




## Using Find Usages Subsystem

You can press Alt+F7 on a node (no matter where - in the editor or in the project tree) to see what kind of usages MPS can search for.

You can also right-click a node and select "Find Usages" to open the "Find usages" window.




Finders - select the categories of usages you want to search for

Scope - this lets you select where you want to search for usages - in concrete model, module, current project or everywhere.

View Options - additional view options

After adjusting your search, click OK to run it. Results will be shown in the Find Usages Tool as shown above.

## Finders

To implement your own mechanism for finding related nodes, you should become familiar with Finders. For every relation there is a specific Finder that provides all the information about the search process.

### Where to store my finders?

Finders can be created in any model by importing findUsages language. However, MPS collects finders only from findUsages lan guage aspects. So, if you want your finder to be used by the MPS Find Usages subsystem, it must be stored in the findUsages a spect of your language.

### Finder structure

| | |
|---|---|
| name | The name of a finder. You can choose any name you want, the only obvious constraint being that the names must be unique in the scope of the model. |

| | |
|---|---|
| for concept | Finder will be tested for applicability only to those nodes that are instances of this concept and its subconcepts. |
| description | This string represents the finder in the list of finders. Should be rather short. |
| long description | If it's not clear from the description string what exactly the finder does, you can add a long description, which will be shown as a tooltip for the finder in the list of finders. |
| is visible | Determines whether the finder is visible for the current node. For example, a finder that finds ancestor classes of some class should not be visible when this class has no parent. |
| is applicable | Finders that have passed for concept are tested for applicability to the current node. If this method returns true, the finder is shown in the list of available finders; otherwise it is not shown. The node argument of this method is guaranteed to be an instance of the concept specified in "for concept" or its subconcepts.<br>Please note the difference between is visible and is applicable. The first one is responsible only for viewing. The second one represents a "valid call" contract between the finder and its caller. This is important because we have an execute statement in findUsagesLanguage, which will be described later. See execute section below for details. |
| find | This method should find given node usages in a given scope. For each found usage, use the add result statem ent to register it. |
| searched nodes | This method returns nodes for which the finder searched. These nodes are shown in searched nodes subtree in the tool.<br>For each node to display, use the add node statement to register it. |
| get category | There are a number of variants to group found nodes in the tool. One of them is grouping by category, that is given for every found node by the finder that has found it. This method gives a category to each node found by this finder. |

### What does the MPS Find Usages subsystem do automatically?

- Stores search options between multiple invocations and between MPS runs
- Stores search results between MPS runs
- Automatically handles deleted nodes
- All the visualization and operations with found nodes is done by the subsystem, not by finders

## Specific Statements

### execute

Finders can be reused thanks to the execute statement. The execution of this statement consists of 2 steps: validating the search query (checking for concept and isApplicable), and executing the find method. That's where you can see the difference between isApplicable and isShown. If you use isApplicable for cases when the finder should be applicable, but not shown, you can get an error when using this finder in the execute statement.

## Examples

You can see some finder examples in jetbrains.mps.baseLanguage.findUsages

You can also find all finders by going to the FinderDeclaration concept (Ctrl+N, type "FinderDeclaration", then press ENTER) and finding all instances of this concept (Alt+F7, check instances, then check Global Scope).

Previous Next