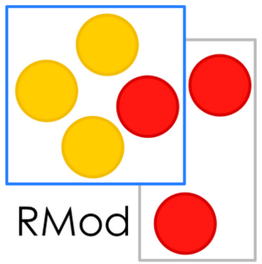
Understanding class names

Nour J. Agouf, Stephane Ducasse, and Anne Etien



For the experiment

For each participant

- Allocate between 30 min and 1h
- Record your screen during your experiment
- Please express your thoughts loudly,
- Take notes of the changes on class names you would like to rename, and the recurrent patterns you may detect

As a group

- Please compare your notes and compile a single todo -
  We can join for this session
- Send us: videos + notes + actions you did

Problematics

- How to know if we can trust that a class name indicates the kind of class it is?

  Examples:

  - ClyTableDecorationStrategy is it a strategy or an annotation?
  - ClyVariables is it a query or a result?
- How to assess if class is regularly and coherently named?

Our proposal: Class names Distribution visualisation

What’s ClassNames Distribution?

It is a package-centered visualisation of the distribution of class names suffixes in a project with inheritance perspective.

Purpose

The visualisation is intended to assist code reviewers in the comprehension of source code, the detection of naming conventions, and their violations (inconsistent naming)

Before explaining the visualisation and its associate tool, let’s explain our experiment protocol.

The experiment protocol

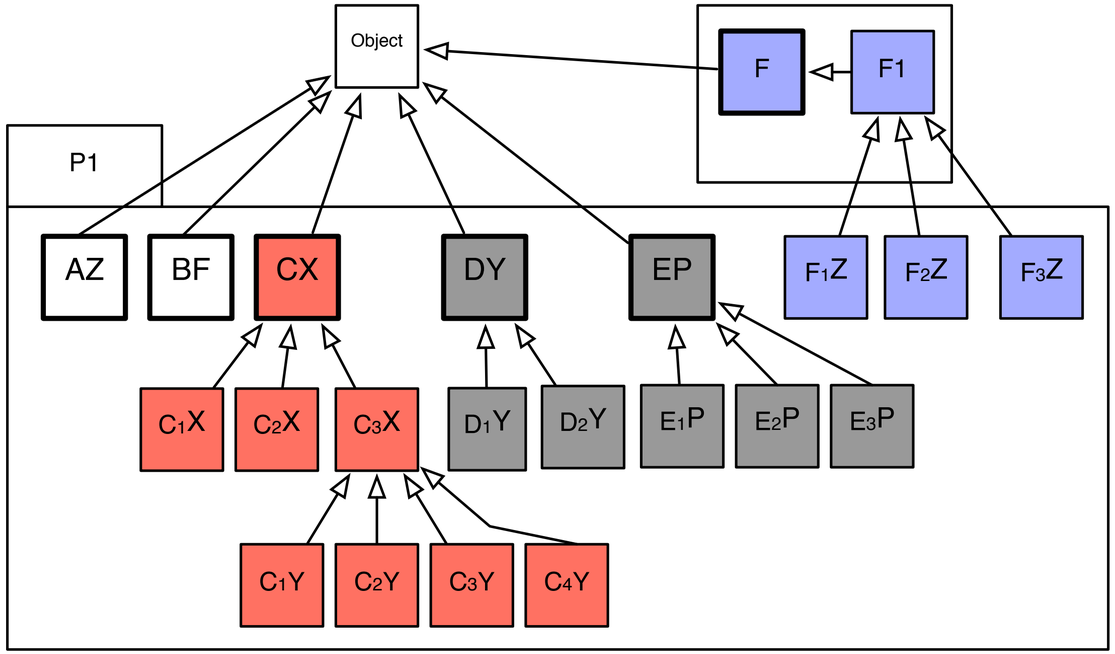
- For each project to study
  - Each member of the team will independently:
    1. Install the tool and configure it for the project
    2. Record his/her screen while using the tool to identify class name inconsistencies. Don’t hesitate to loudly explain what you have in mind if you also record the micro.
    3. Note on a separate document or directly rename in his/her image the classes inconsistently named. We need to keep track on the changes per participant.

The experiment protocol 2

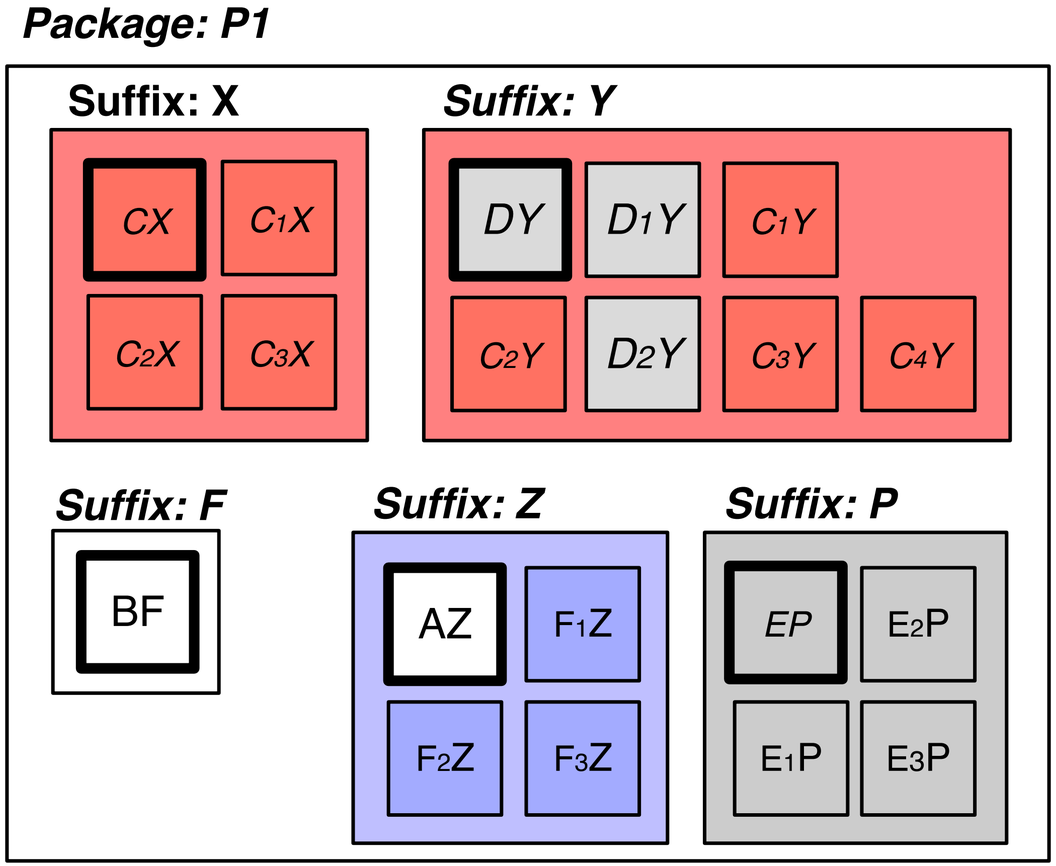
- Make a meeting with all the participant of the experiments to:
  - Discuss about the changes
  - Accept some changes
- Send us feedbacks on:
  - The number of accepted / rejected renaming; in production?
  - If you got naming conventions: what were they? were they followed or violated?
  - The discovery of class name convention violation you were not aware
  - The use of the tool in the process
- Send us all this material.

The tool description

One View per package

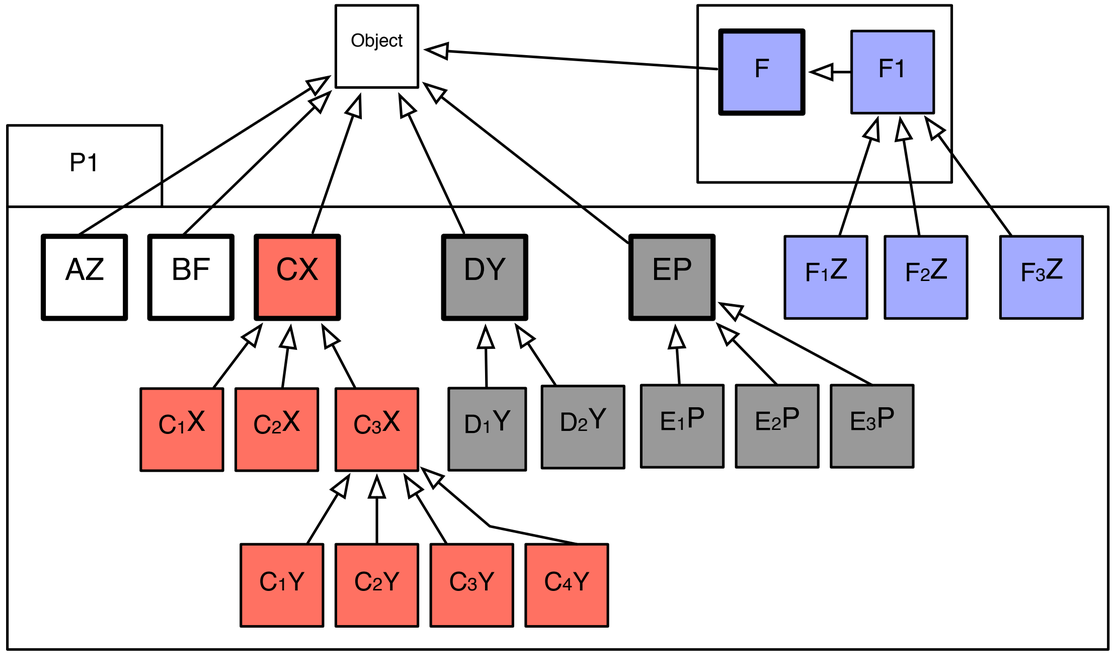


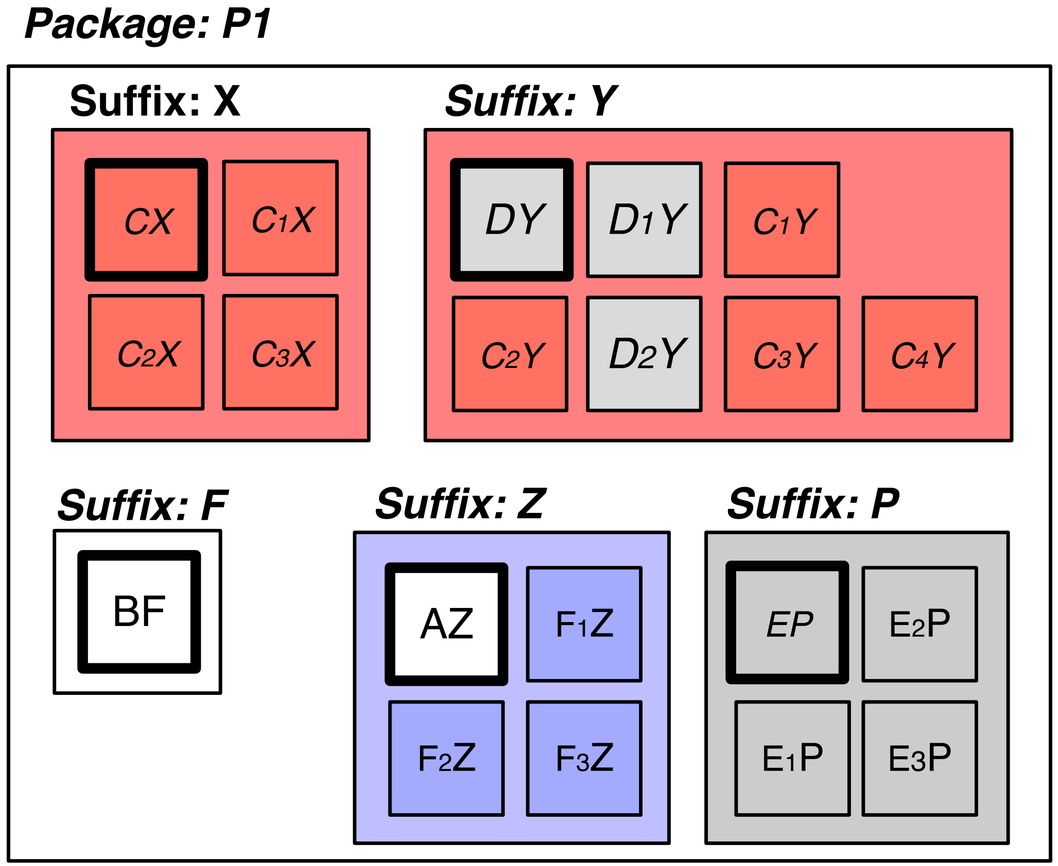
A schematic mini project composed of A, B, C, D, E, F hierarchies.



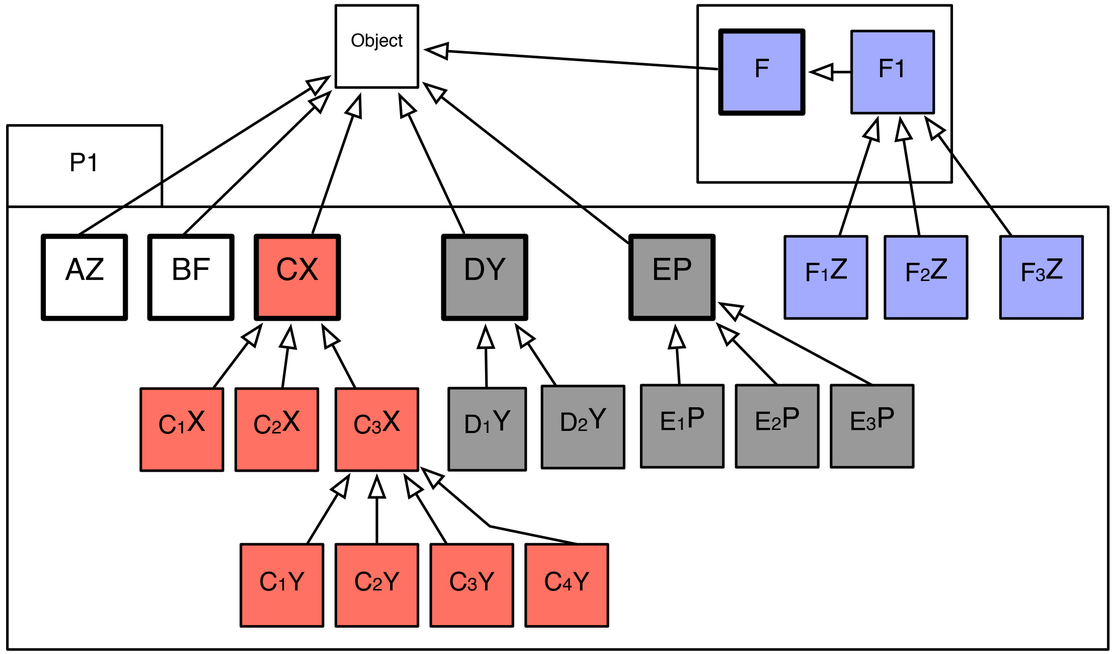
Its corresponding Class Name Distribution visualisation

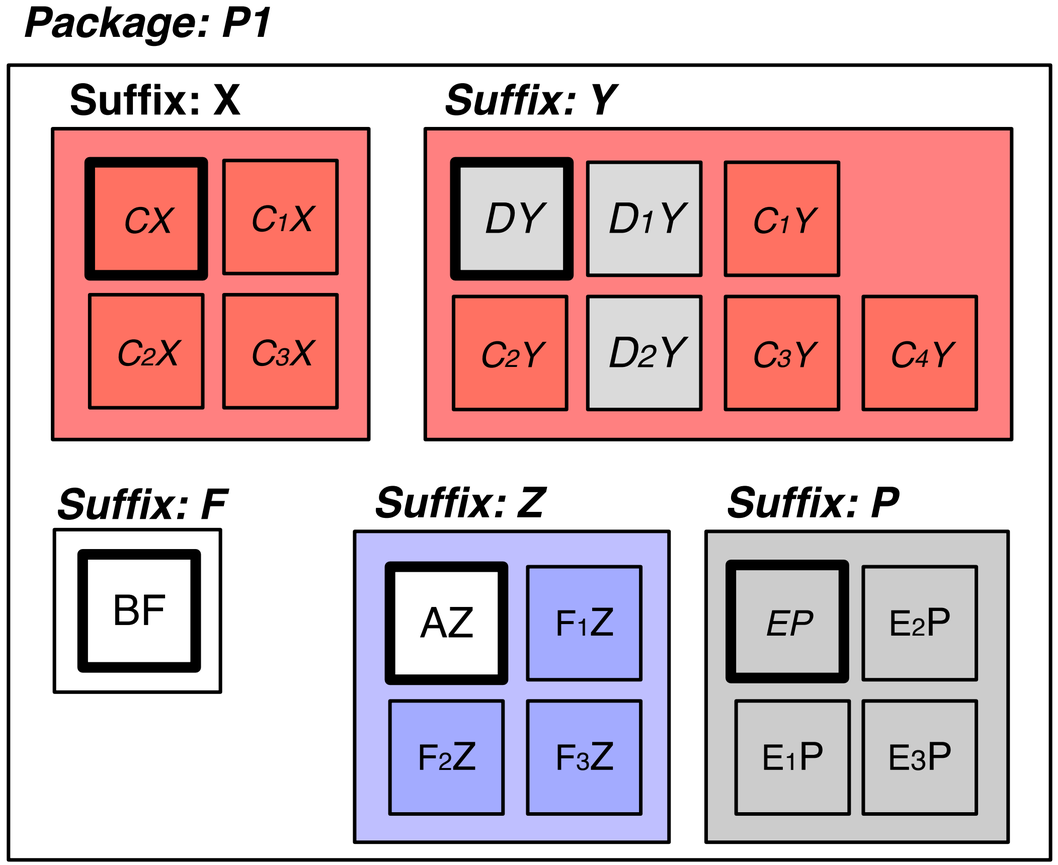
Root classes have bold borders (CX, DY, BF, AZ, EP)





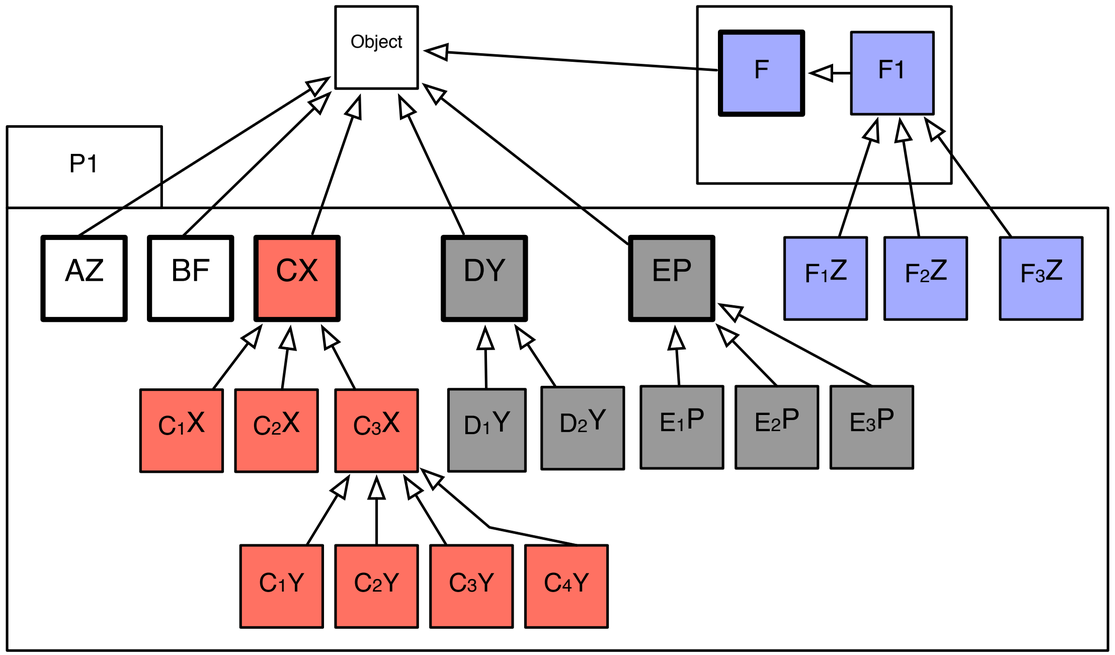
Single classes are White (BF, AZ)

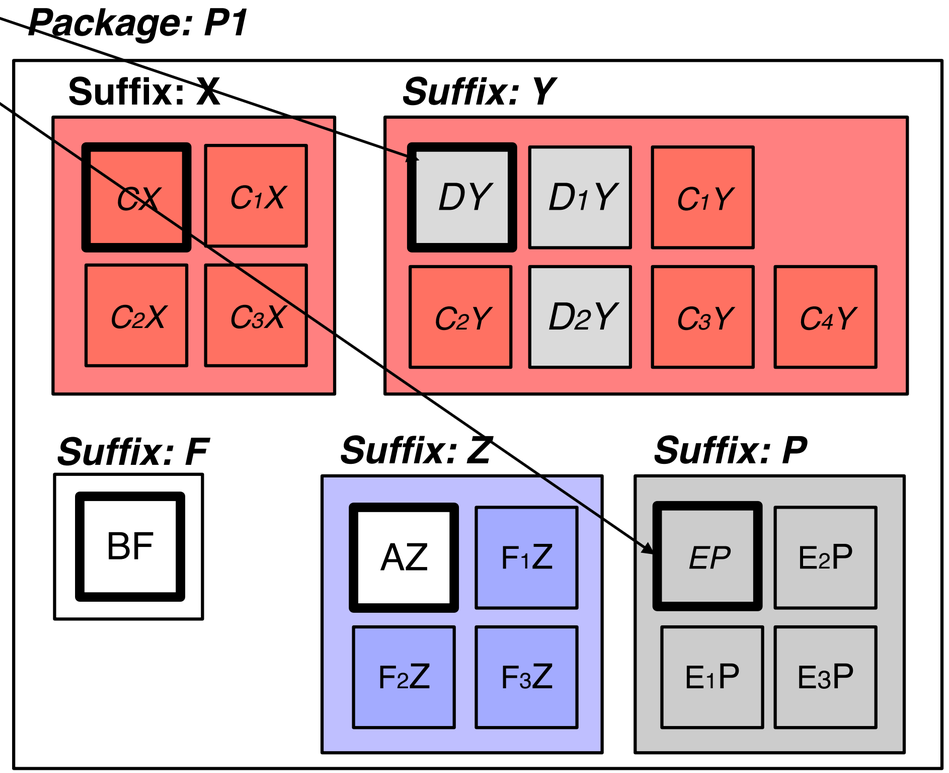




White means that the class is either: a Trait or belongs to no hierarchy and has no subclasses

Coherently named hierarchies in Gray



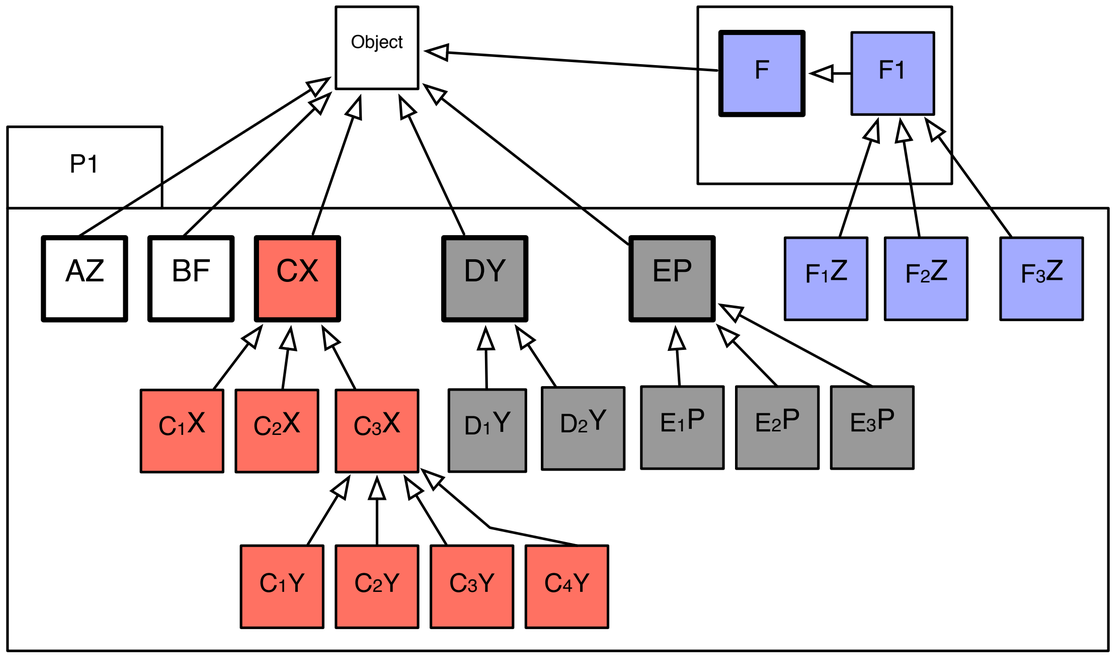


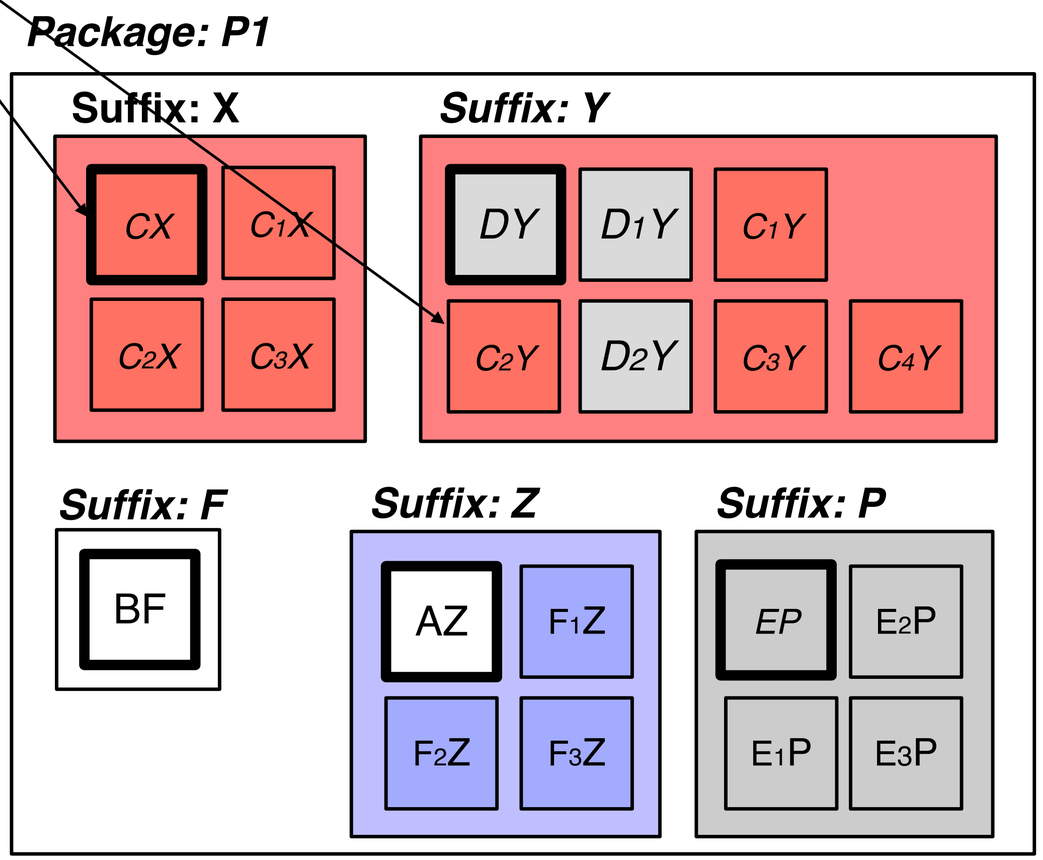
Gray means the hierarchy is consistently named

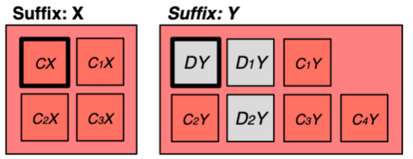
(Since they’re named correctly, do not worry about them)

One Color per ‘strange’ hierarchy.

CX uses more than one suffix, X and Y





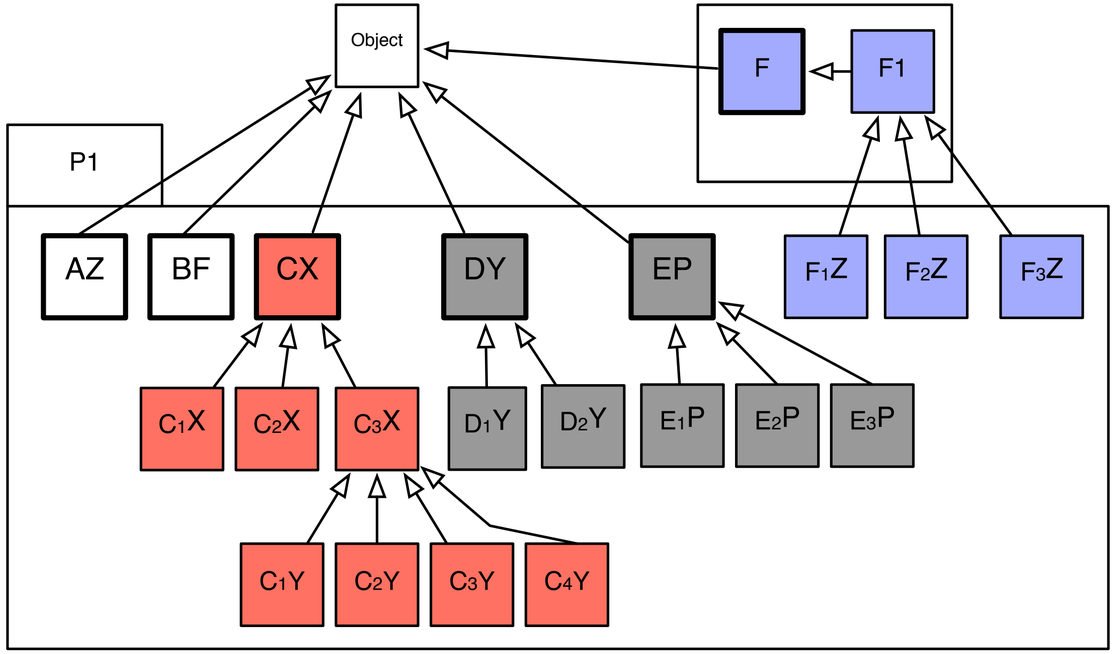
multiple suffixes (

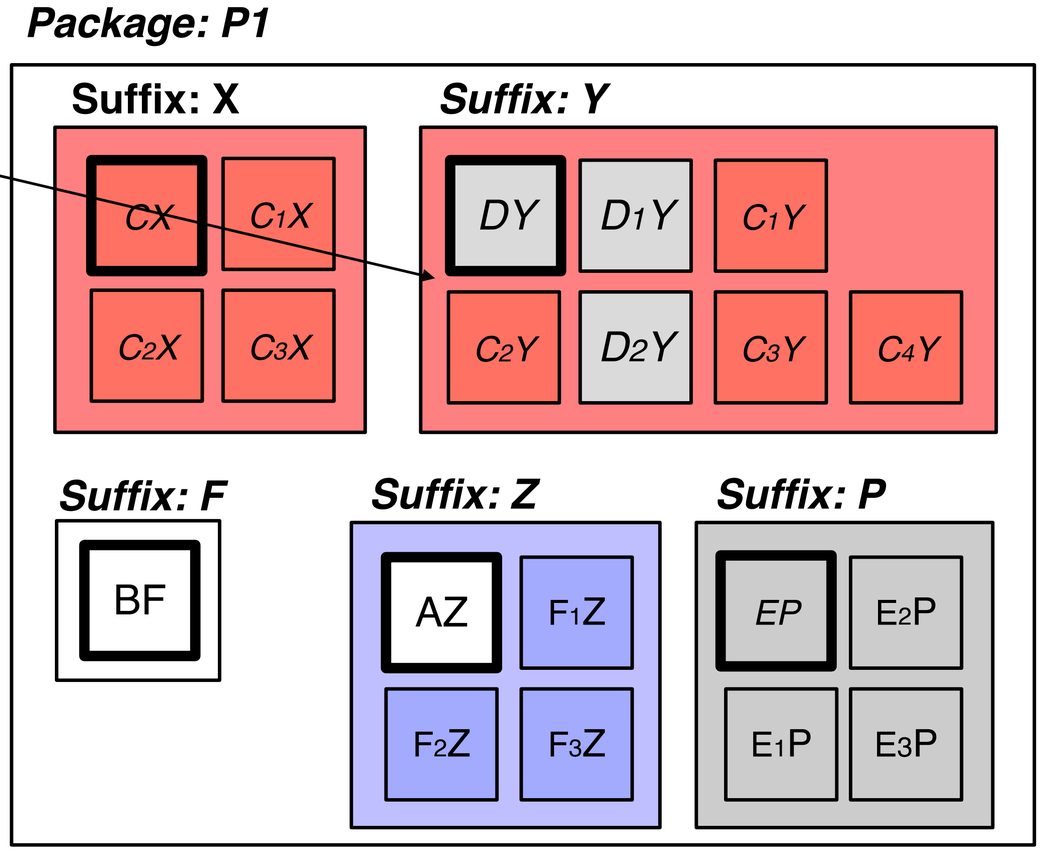
)

=> hierarchy is not consistently named
=> should check why

Here the background is red

because we have 4 classes from CX





Recap:

A solo class with no subclasses → **No hierarchy, single class**

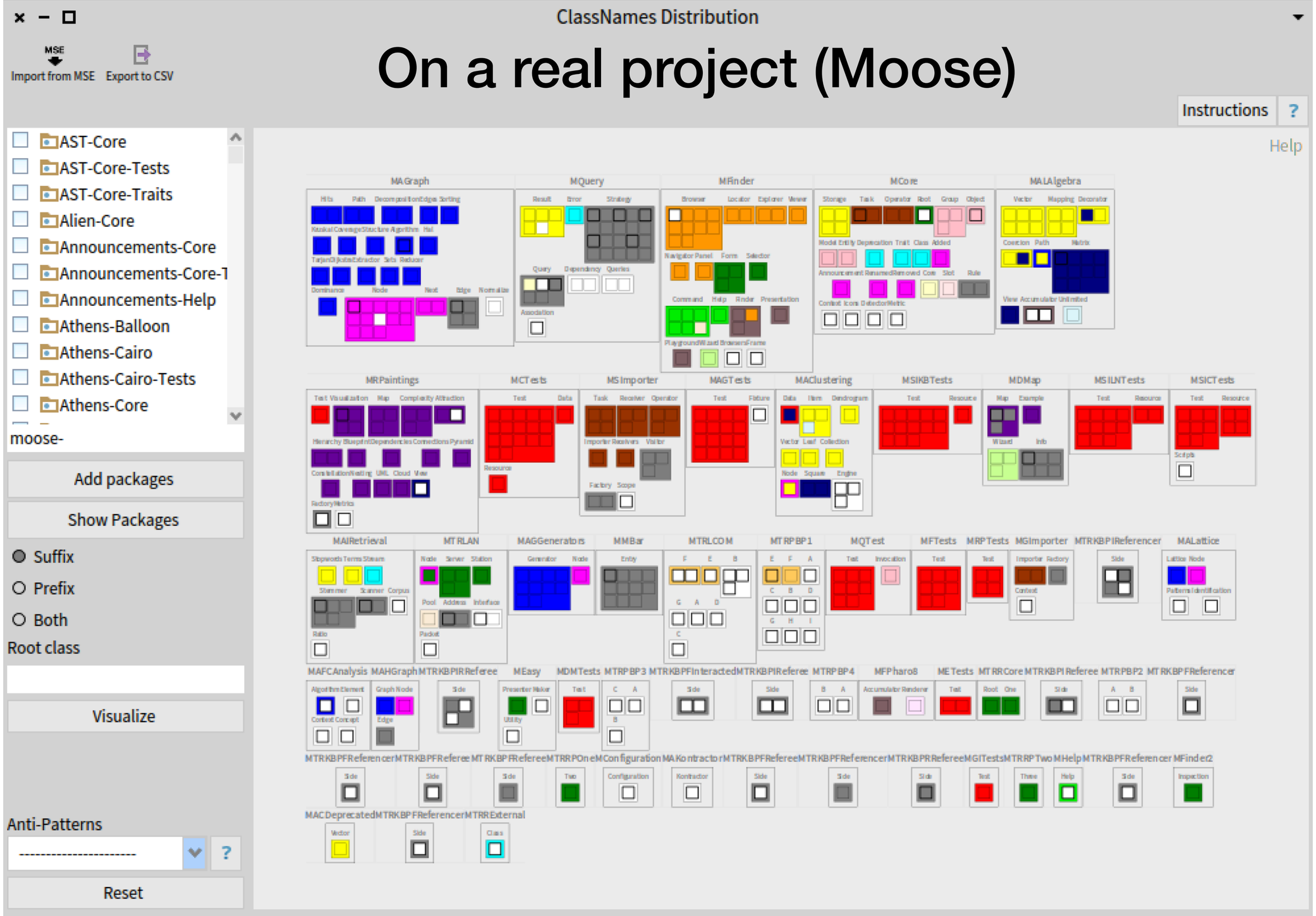
One hierarchy -> One suffix → **Consistent naming**

→ **One hierarchy => one color**

Color palette of the first 24 hierarchies using more than one suffix.
One hierarchy -> several suffixes

Color of classes starting from the 25th hierarchy

On a real project (Moose)



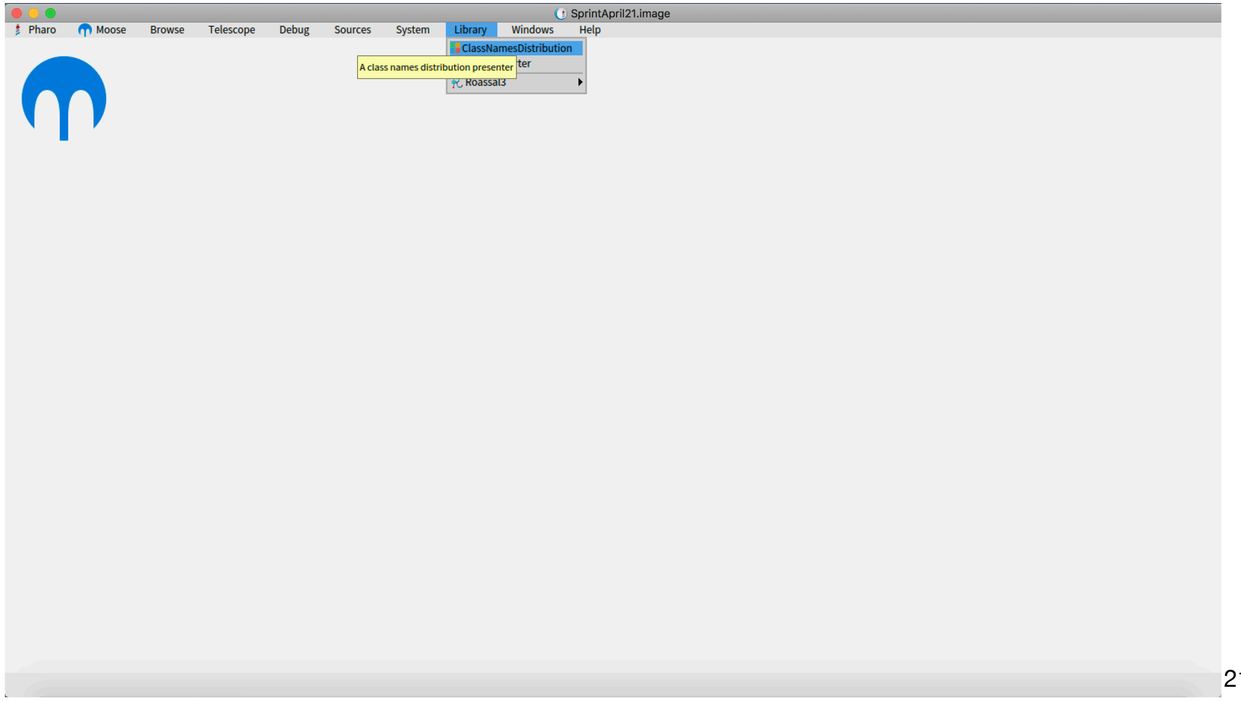
Install the tool on a Moose 9 image

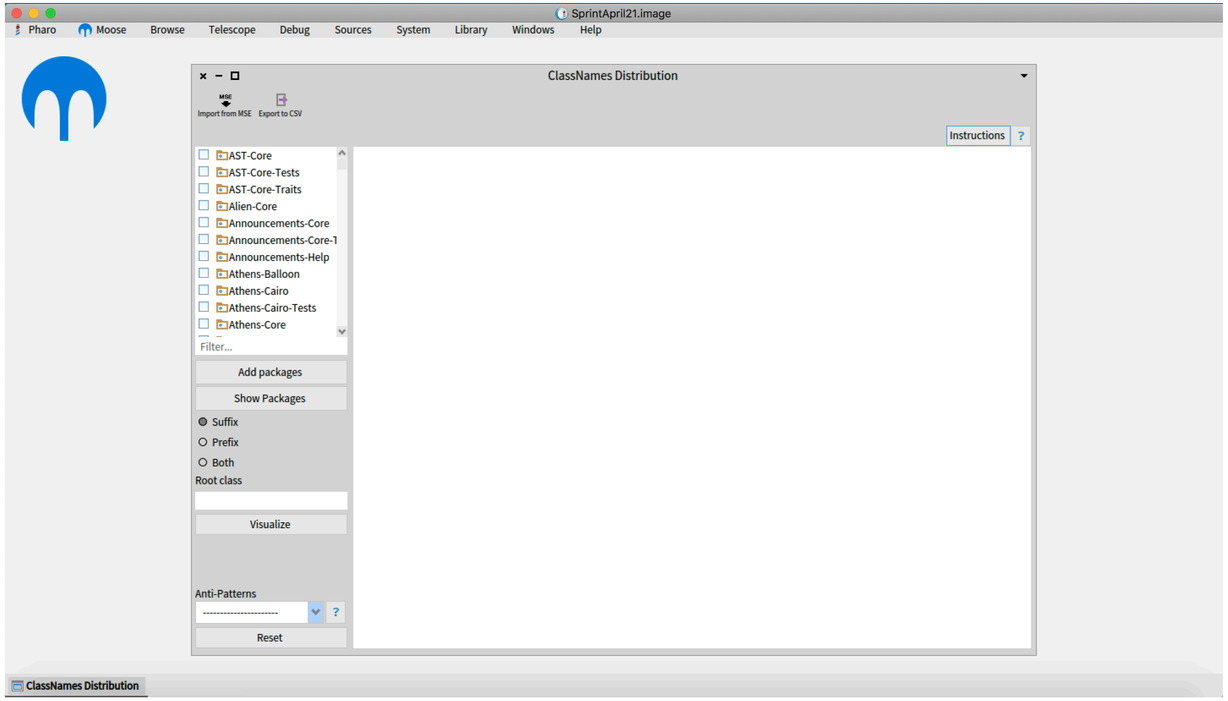
1. Take a fresh Moose 9 image
2. Install your project to evaluate if it is not by default in the image
3. Install Class Name Distribution by executing the following code from a Playground

```
Metacello new
  baseline: 'ClassNameAnalyser';
  repository: 'github://NourDjihan/ClassNameAnalyser/src';
  load
```

Open the tool

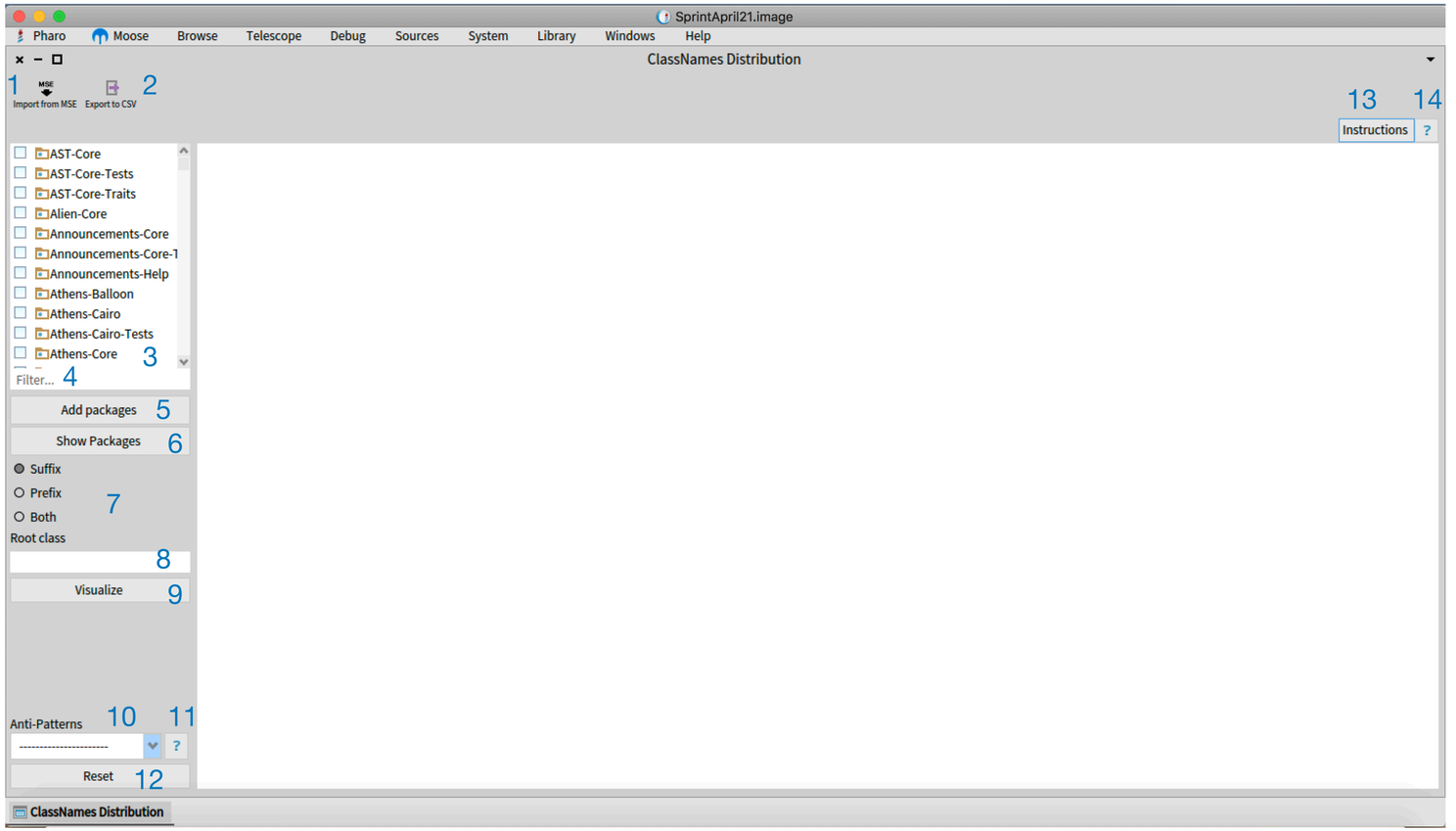
On the top of your image, click on Library and then ClassNamesDistribution as shown in the left picture.
It opens a new window as on the right picture.





Configure the tool for your project

1. In the filter field (4) enter the name of your project to select the corresponding packages (ex: `Roassal3-`)
2. Click on all the packages (3) of your project (the shortcut Ctrl+A works)
3. Click on Add packages (5)
4. Click on Visualize (9)

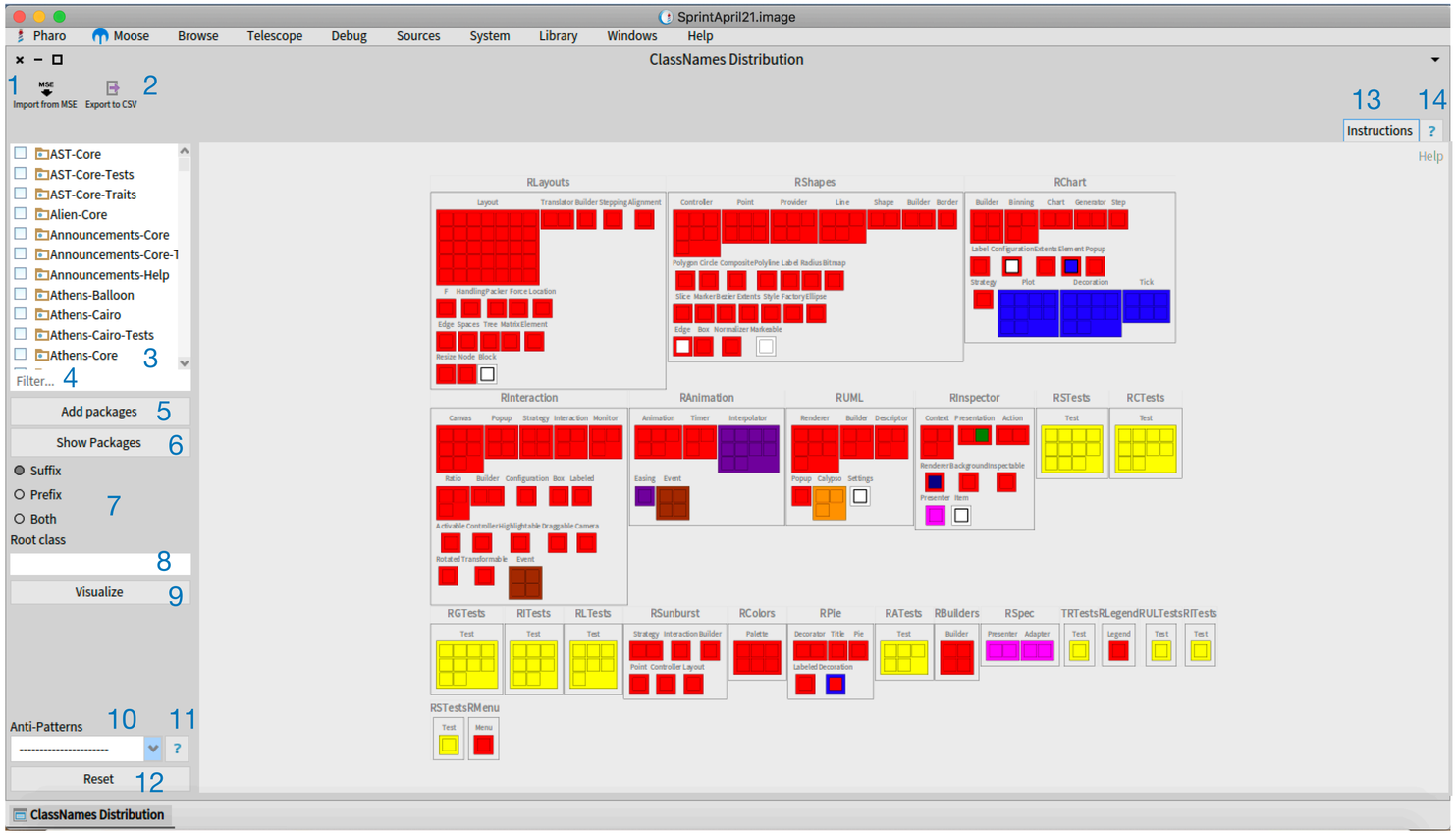


Configure the tool for your project

This part is optional

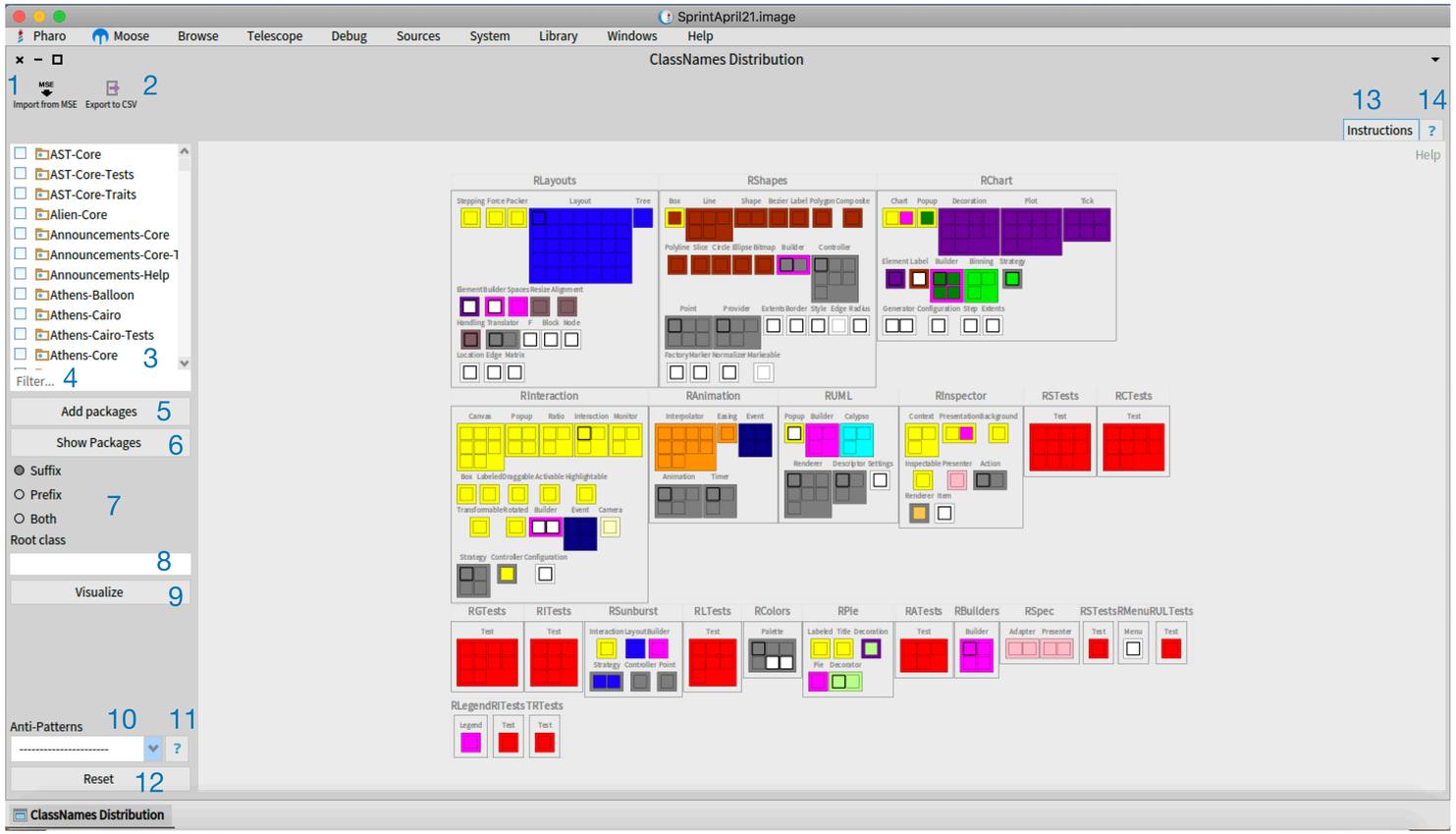
Specify a root class if all your class inherits from the same class

1. Write a root class name in the field text (8) for example: RSObject
2. Click on the Visualize button (9)



Ready to use the tool!!

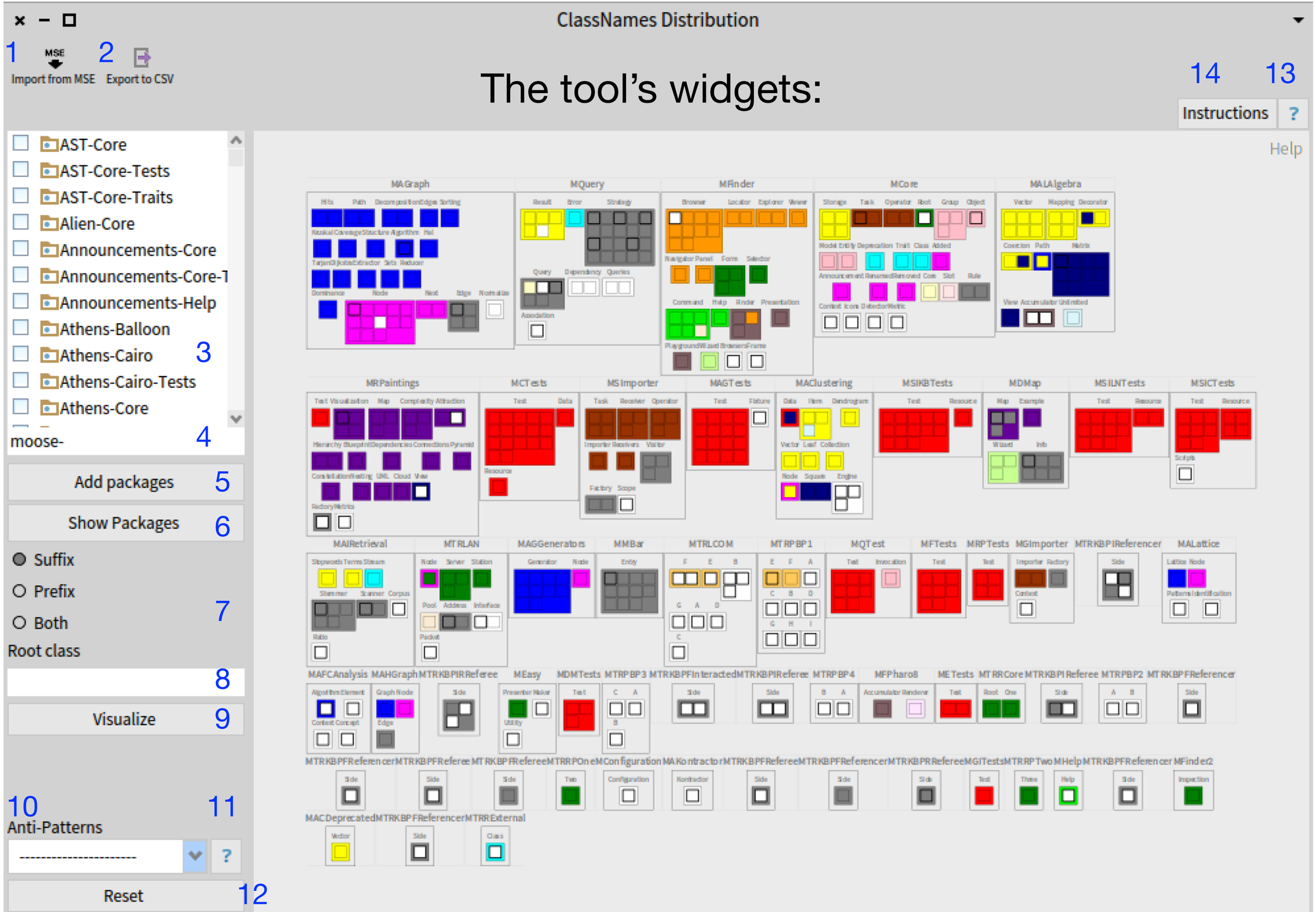
We will try to understand the inconsistencies in the class naming by looking at the colored hierarchies.



Normally, you don't need to use these widgets anymore

(from 1 to 9)

The tool's widgets:



1- Import the MSE file

2- Export the visualisation data into

a CSV file

3- The list of packages

4- Write the name of the

project (first word of packages names

in Pharo)

5- Click to add the selected packages

6- Click to show the selected packages,

delete some or all.

7- Select the token to be extracted from

the class name

8- Define the root class (‘Object’ by default)

9- Click to see the visualisation

10- Select the pattern

11- Patterns explanation

12- Reset the visualisation to its first state

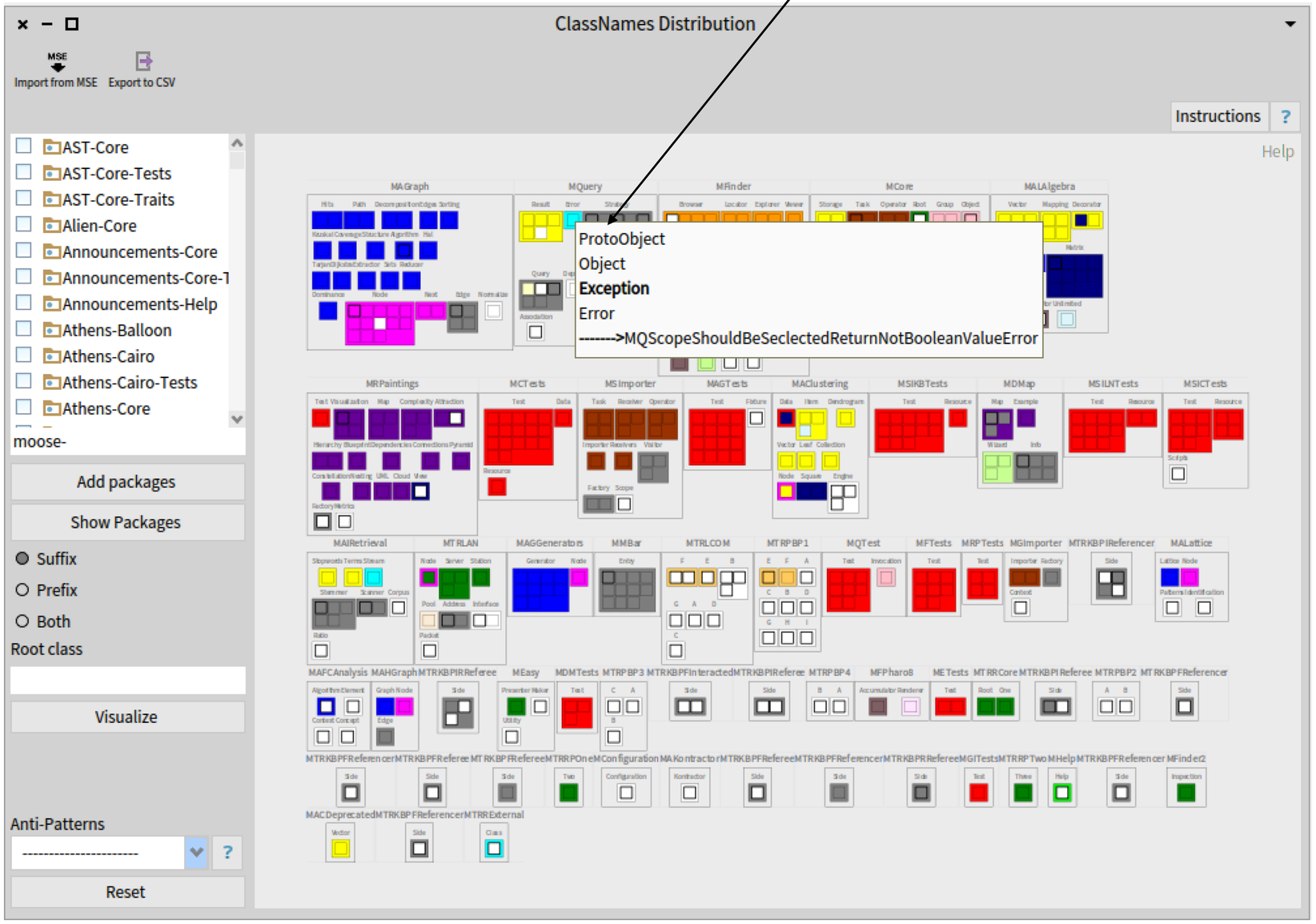
13- Help of the visualisation

(explanation of ClassNames

Distribution principles)

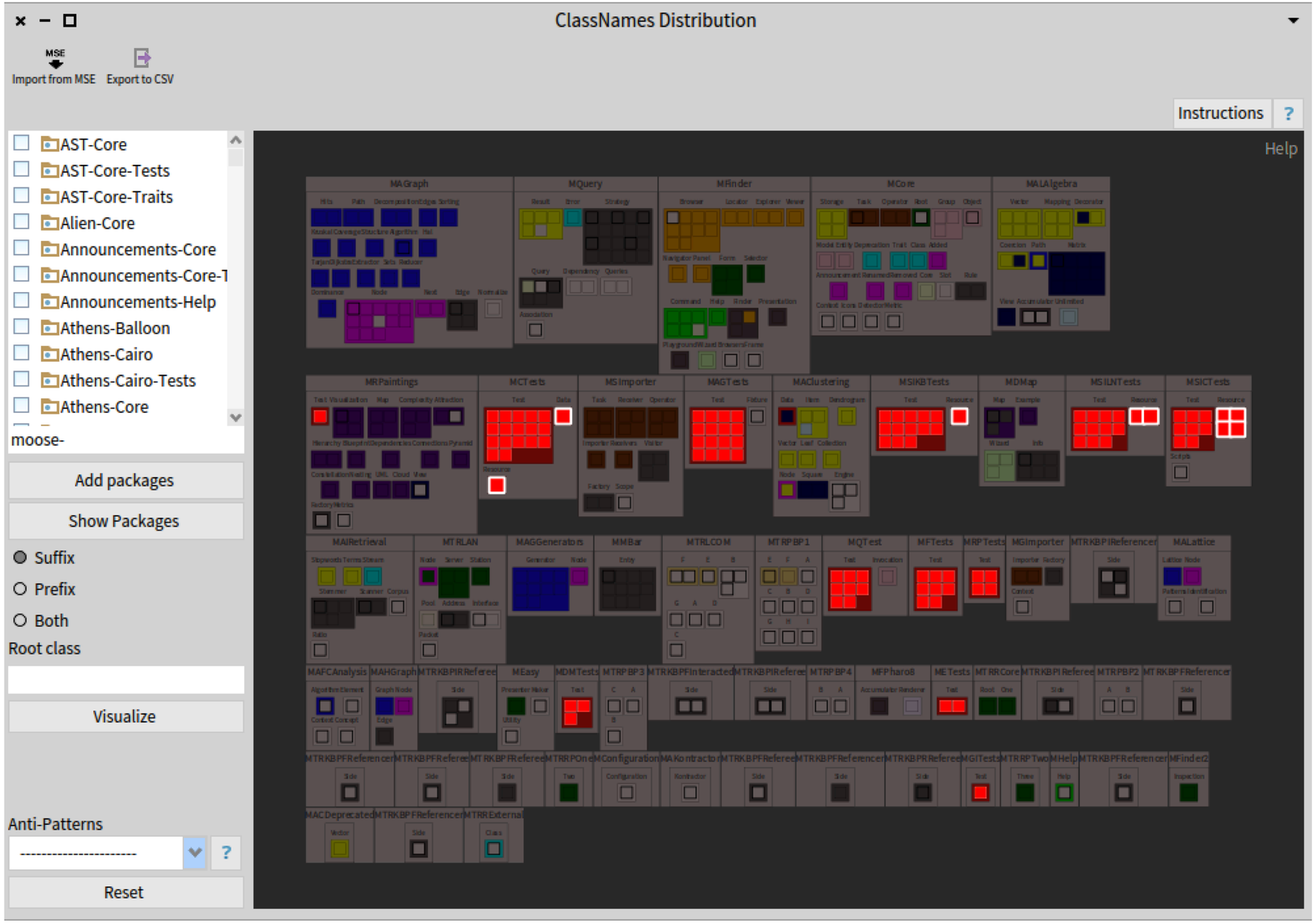
14- The instructions to follow using the tool.

Mouse hover a class shows the hierarchy of the class:
- The root class in bold,
- The arrow is followed by the name of the class itself



Left click on a class highlights the whole hierarchy of the class

The red hierarchy classes are highlighted



Class boxes in white border indicate suspiscious cases

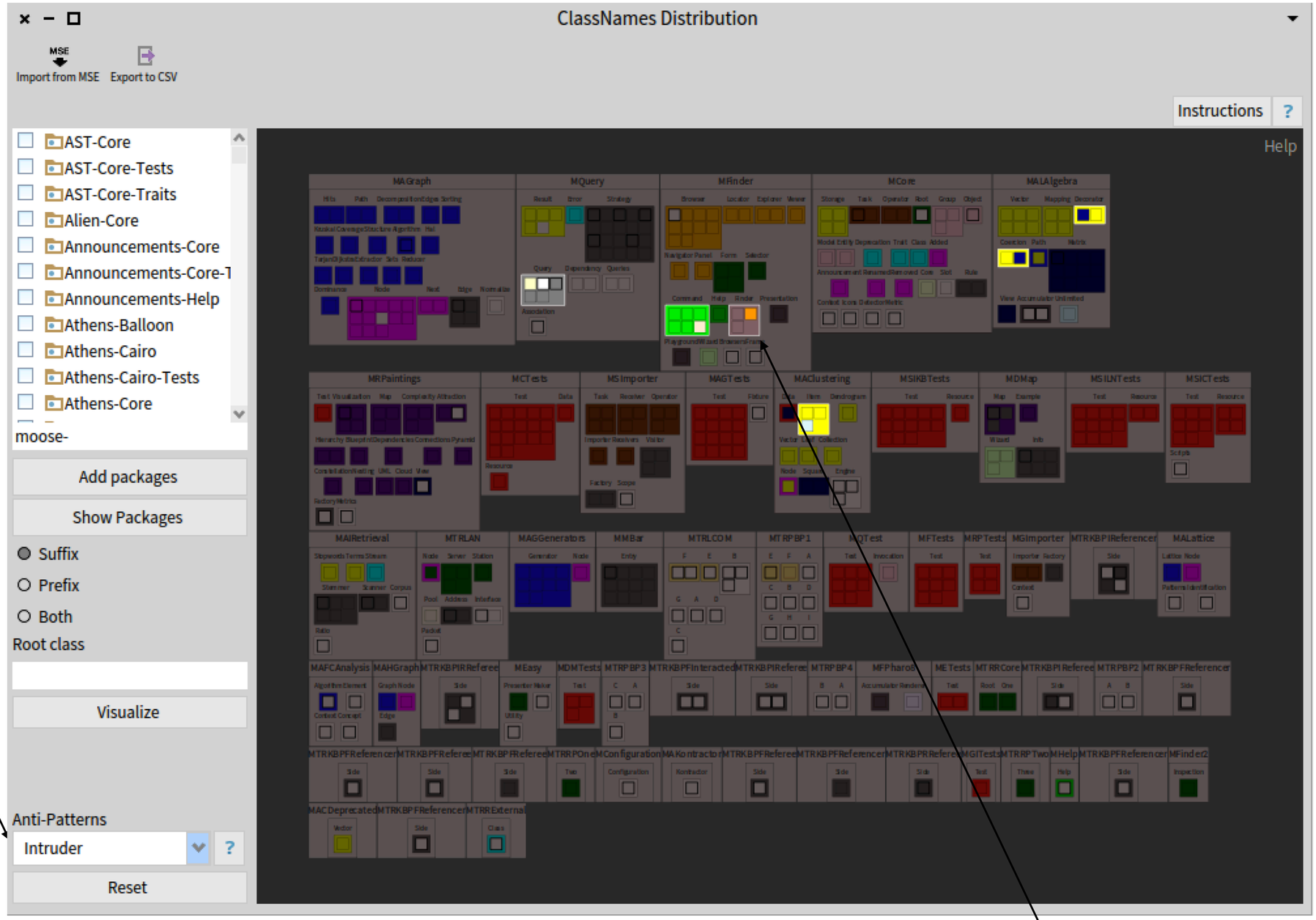
Summary of the mouse interactions

on a class box

- **Mouse hover** shows the hierarchy of the class:
  - The root class in bold,
  - The arrow is followed by the name of the class itself
- **Left click** highlights the whole hierarchy of the class
  (so all the classes of the same color)
- **Right click** opens the class browser
  (if you need more information about the class or want to directly rename it in that case click again on the *Visualize* button (9) to update the visualisation)

Anti-Pattern:

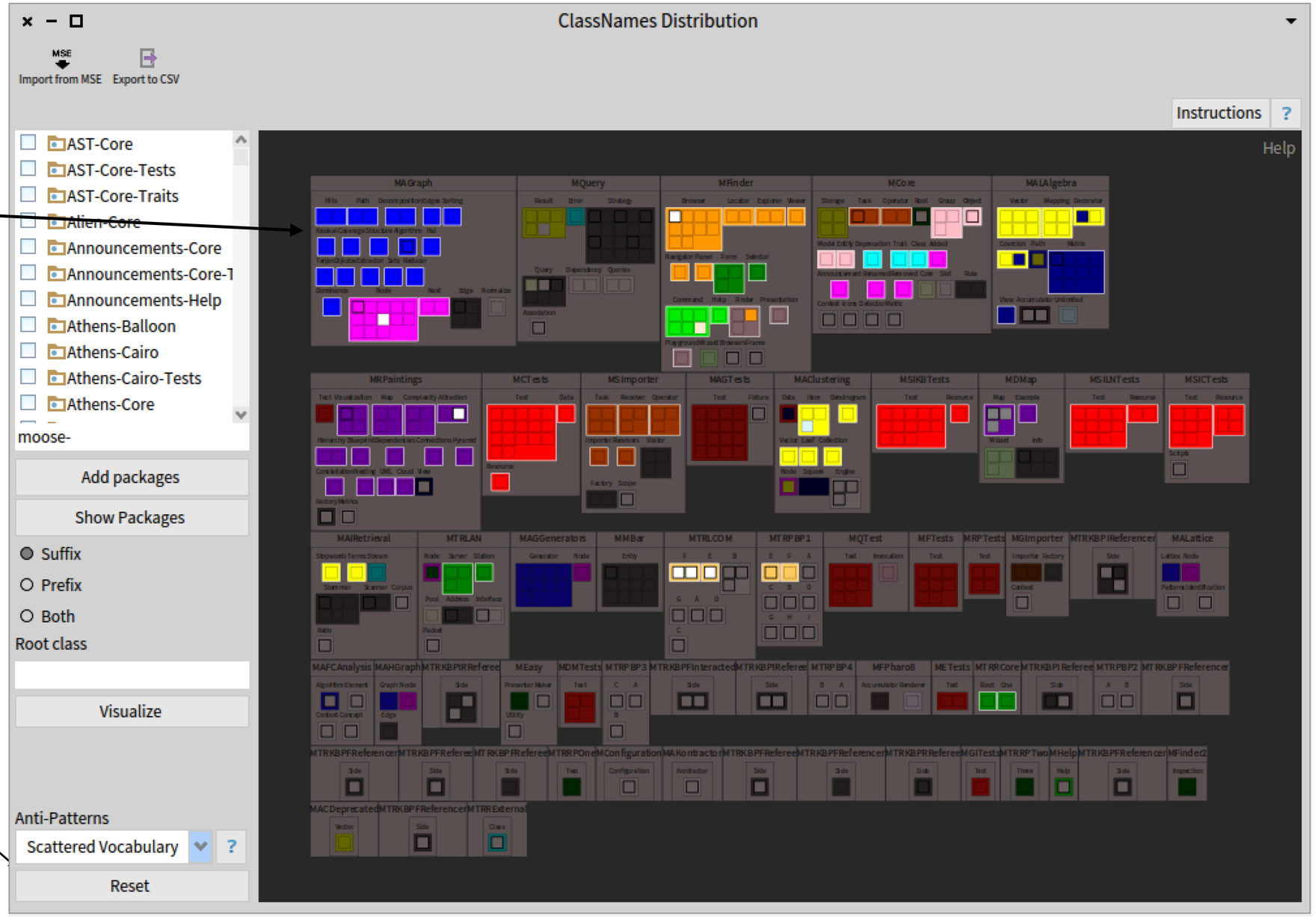
Intruder



Why having twice the same suffix in different hierarchies (light brown and orange)?

Hierarchy in blue has
16 suffixes in
the first package

Anti-Pattern:
Scattered
Vocabulary



Why would a hierarchy use more than one, or 2 suffixes?

Recap:

A solo class with no subclasses → **No hierarchy, single class**

One hierarchy -> One suffix → **Consistent naming**

→ **One hierarchy => one color**

Color palette of the first 24 hierarchies using more than one suffix.
One hierarchy -> several suffixes

Color of classes starting from the 25th hierarchy

For the experiment

For each participant

- Allocate between 30 min and 1h
- Record your screen during your experiment
- Please express your thoughts loudly,
- Take notes of the changes on class names you would like to rename, and the recurrent patterns you may detect

As a group

- Please compare your notes and compile a single todo -
  We can join for this session
- Send us: videos + notes + actions you did