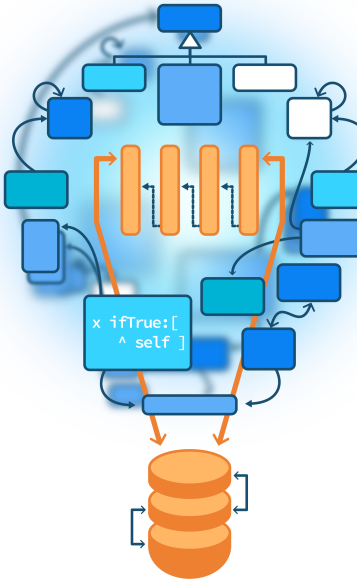
Singleton

a highly misunderstood pattern

S.Ducasse, L. Fabresse, G. Polito, and P. Tesone



Outline

- Singleton
- Singleton discussions
- Singleton misunderstanding

Singleton intent

- **From the book**: Ensure that a class has only one instance, and provide a global point of access to it
- **Better**: Ensure that a class has only one instance available **at the any time**

Problem/Solution

- **Problem:** Need
  - a way to keep some persistent objects around
  - or a class with a unique instance
- **Solution:** Store the first time an instance is created and return it each time a new instance is requested

Most of the time think twice because you probably do not need it!

Example

```
db := DBConnect uniqueInstance.
db2 := DBConnect uniqueInstance.

db2 == db
> true
```

Yes we get only one instance of the database connection

Possible implementation

```
Object << #BDConnect
  sharedVariables: { UniqueInstance }
```

```
BDConnect class >> uniqueInstance
  UniqueInstance isNil
      ifTrue: [ UniqueInstance := self new ].
  ^ UniqueInstance
```

Should we override new?

```
DBConnect class >> new
  ^ self uniqueInstance
```

The intent (uniqueness) is not clear anymore!

- new is normally used to return **newly** created instances
- new means to get a new object and initialize that object
- uniqueInstance doesn't convey the same

Method name variation (I)

uniqueInstance

- Pure singleton ensuring a single global instance
- new should better be blocked

```
Author class >> uniqueInstance
  ^ uniqueInstance ifNil: [ uniqueInstance := self basicNew initialize ]

Author class >> new
  self error: 'Author is a singleton -- send uniqueInstance instead'
```

Method name variation (II)

default

- Some meaningful default instance, but there is no reason to stop the user from **creating more instances**

current

- Keep the same instance system-wide, but we also want to **change it** under some circumstances

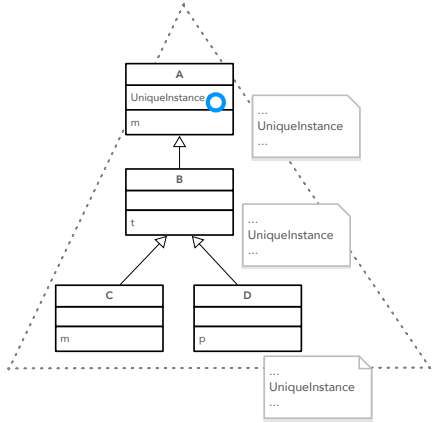
Discussion

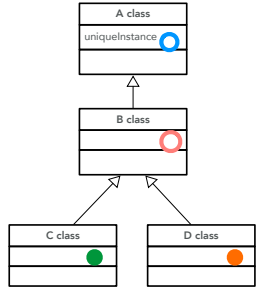
- Even if the language supports global variables, avoid to store a Singleton in a global
- A class is already acting as a global and it can manage the Singleton (one single entry point)

Shared variable vs class instance variable

In Pharo we have:

- **Shared variables**: shared between all the class of a hierarchy
- **Class instance variables**: specific to a single class

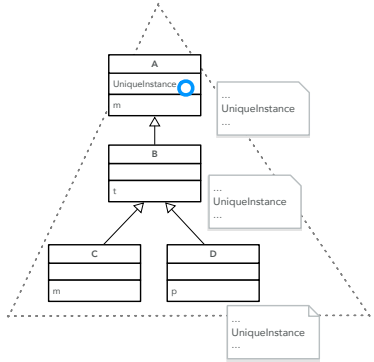


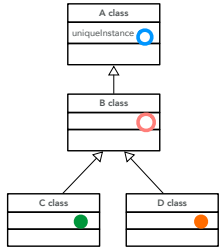


One per hierarchy or one per class

Holding a singleton with

- **a shared variable**: One singleton for a complete hierarchy
- **a class instance variable**:
  - One singleton per class
  - Each subclass has its own singleton





Singleton misunderstanding

- Singleton is **about time**: only one instance at the any time is possible
- Singleton is **not** about access: don't use a singleton because it is easier to access one instance!

Singleton acid test

- If you can add one instance variable to your object and suddenly you do not need a singleton then it was not a singleton but an ugly disguised global variable!
- Sometimes you cannot add an instance variable so the Singleton is ok

Testing singletons

- Singletons are global variables so this makes them more difficult to test
- When running tests, you want to avoid to change the current singleton
- Be careful about not breaking the current singleton
- RPackageOrganizer is a singleton: should not be destroyed when tests are run

Example: RPackageOrganizer

RPackageOrganizer uses withOrganizer: aNewOrganizer do: aBlock for testing behavior

```
withOrganizer: aNewOrganizer do: aBlock
  "Perform an action locally to aNewOrganizer. Does not impact any other organizers."

  | old |
  [ old := self organizer.
  old unregister.
  self organizer: aNewOrganizer.
  aNewOrganizer register.
  aBlock cull: aNewOrganizer ] ensure: [
    self organizer: old.
    old register.
    aNewOrganizer unregister]
```

Conclusion

- Having only one instance **at a time**
- Avoid Singleton as a global
- In general avoid Singleton because it **acts as a global**
- Difficult to test

Produced as part of the course on http://www.fun-mooc.fr

Advanced Object-Oriented Design and Development with Pharo

A course by
S.Ducasse, L. Fabresse, G. Polito, and P. Tesone

Inria
LearningLab

2023