Customization degree of hooks

Class vs. instance hooks

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

http://www.pharo.org

Goals

- Thinking about hooks
- Granularity of hooks: class vs instance creation
- Do we customize instance creation or not?
- Customisation of instance creation

Example: Introducing a hook

```
CHInterpreter >> visit...: arg

  scope := CHClassScope new xxx
  ...
  scope
```

Extract a class

```
CHInterpreter >> visit...: arg

  scope := self classScopeClass new xxx
  ...
  scope
```

```
CHInterpreter >> classScopeClass
  ^ CHClassScope
```

A subclass can push a different class

```
CHInterpreter2 >> classScopeClass
  ^ MyClassScope
```

- Ok, it works!

Black box parametrization

```
Object << CHInterpreter
  slots: {#classScopeClass};
```

With an instance variable and a setter we can also get black box pametrization

```
CHInterpreter new
  classScopeClass: myClassScope;
  yourself
```

Analysis

- What if we have a more complex instantiation?
- And we want to encapsulate it and let the extender change it?
- `self classScopeClass new xxx` is not at a good granularity

```
CHInterpreter >> visit...

  scope := self classScopeClass new xxx
  ...
  scope
```

Instance creation

```
CHInterpreter >> visit...: arg

  scope := self newClassScope.
  ...
  scope
```

```
CHInterpreter >> newClassScope

  ^ self classScopeClass new xxx
```

Imagine

If you can only pass a class

- you cannot customize and access extra protocols
- you will have to do circumvoluted tricks (wrapping)

Passing object as context

```
CHInterpreter >> visit...: arg

  scope := self newClassScopeNewFor: arg
  ...
  scope
```

```
CHInterpreter >> newClassScopeNewFor: arg

  ^ self newClassScope doSomethingWith: arg
```

- When we pass arg, we give the class the possibility to customise instantiation.
- Subclasses can customize/extend doSomethingWith:

Conclusion

- Class hooks are nice
- But think also about others hooks

Advanced Object-Oriented Design and Development with Pharo

A course by

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

2023