Modules: Dreams & Reality

Gilad Bracha
SAP Labs

Reality

Hardware Modules

Hardware Modules

Hardware Modules

Plug'n Play

aka: Dynamic Configuration

Hardware Modules

Plug'n Play

In to each other!

Hardware Modules

Plug'n Play

aka: Mutual Recursion

In to each other!

Hardware Modules

Multiple Instances of the same design

Mainstream Module Problems

- No mutual recursion
- Single instance of a design per run
- No distinction between module definition and module instances
- Awkward to define multiple configurations
- Modules are 2nd class citizens: cannot be manipulated or reflected

Hardware Modules

Multiple **Instances** of the same design

Designs are instantiated

Classes are instantiated

Designs are hierarchical, and may be nested

Classes may be nested

Designs are themselves artifacts in real space, and can be manipulated

Classes are artifacts in the cyberspace, and can be manipulated

Use classes as unit of modular design

Newspeak

Newspeak

- Newspeak is a dynamic, class based language with two defining properties:
  - All names are late bound
  - No global namespace

Classes Define Modules

Newspeak modularity is based *exclusively* on classes

- No packages, modules, bundles, templates ...

Newspeak

- Newspeak is a dynamic, class based language with two defining properties:
  - All names are late bound
  - No global namespace

No References to Variables

Representation Independence

- Always use slots via accessors

No References to Classes

- Always use accessors
- Classes are first class objects
  - Designs are artifacts
- Classes are always virtual
- Classes are always mixins
- Class hierarchy inheritance

Newspeak

- Newspeak is a dynamic, class based language with two defining properties:
  - All names are late bound
  - No global namespace

The Insidious Import

module *BraveNewWorldExplorer;*

import *Collections. MutableArrayList;*

The Insidious Import

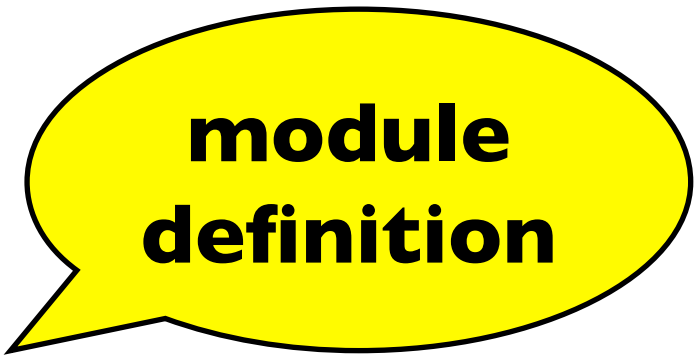
module *BraveNewWorldExplorer;*

import *Collections. MutableArrayList;*

nested within "module"

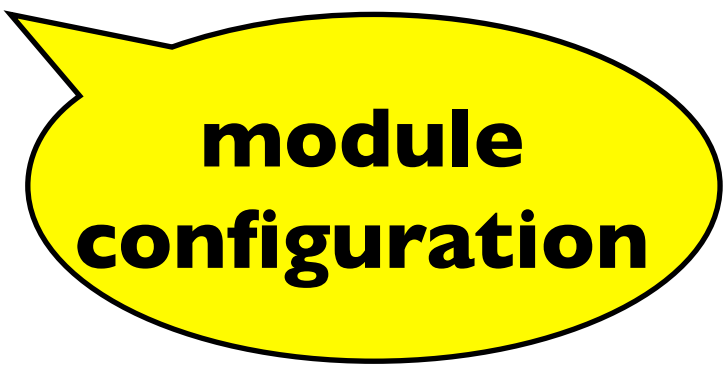
Global name!

The Insidious Import



module *BraveNewWorldExplorer;*

import *Collections. MutableArrayList;*



Module Definition

```
class BraveNewWorldExplorer usingLib: platform = (
  |
  private MutableArrayList =
      platform collections MutableArrayList.
  ...
   |
)( ...)
```

Module Configuration

```
main: platform args: as = (
    platform core HopscotchFramework
        HopscotchWindow
            openSubject:
            ((BraveNewWorldExplorer
            usingPlatform: platform) FileSubject
            onModel: (as at: 1)
            ))
```

Module Configuration

```
main: platform args: as = (
    platform core HopscotchFramework
        HopscotchWindow
            openSubject:
            ((BraveNewWorldExplorer
            usingPlatform: platform) FileSubject
            onModel: (as at: 1)
            ))
```

Module Configuration

```
class BraveNewWorldExplorerApp

    packageUsing: ideNamespace  = (|

        BraveNewWorldExplorer =

            ideNamespace BraveNewWorldExplorer.

    |)(

    main: platform args: as = (...)

    )
```

Module Configuration

Instantiate ***BraveNewWorldExplorerApp*** using tools (e.g., IDE).

Module Deployment

BraveNewWorldExplorerApp instance can be deployed via object serialization.

Module Loading

Serialized instance of ***BraveNewWorldExplorerApp*** can be loaded via object deserialization, followed by invocation of ***main:args:***.

Modules are Sandboxes

Factory method parameters are objects/capabilities that determine per-module sandbox

Side by Side Modules

```
platform:: Platform new.

m1:: NewspeakParsing
        using: platform
       parseLib: (CombinatorialParsing
                              usingLib: platform)

m2:: NewspeakParsing
        using: platform
       parseLib: (PackratParsing usingLib: platform)
```

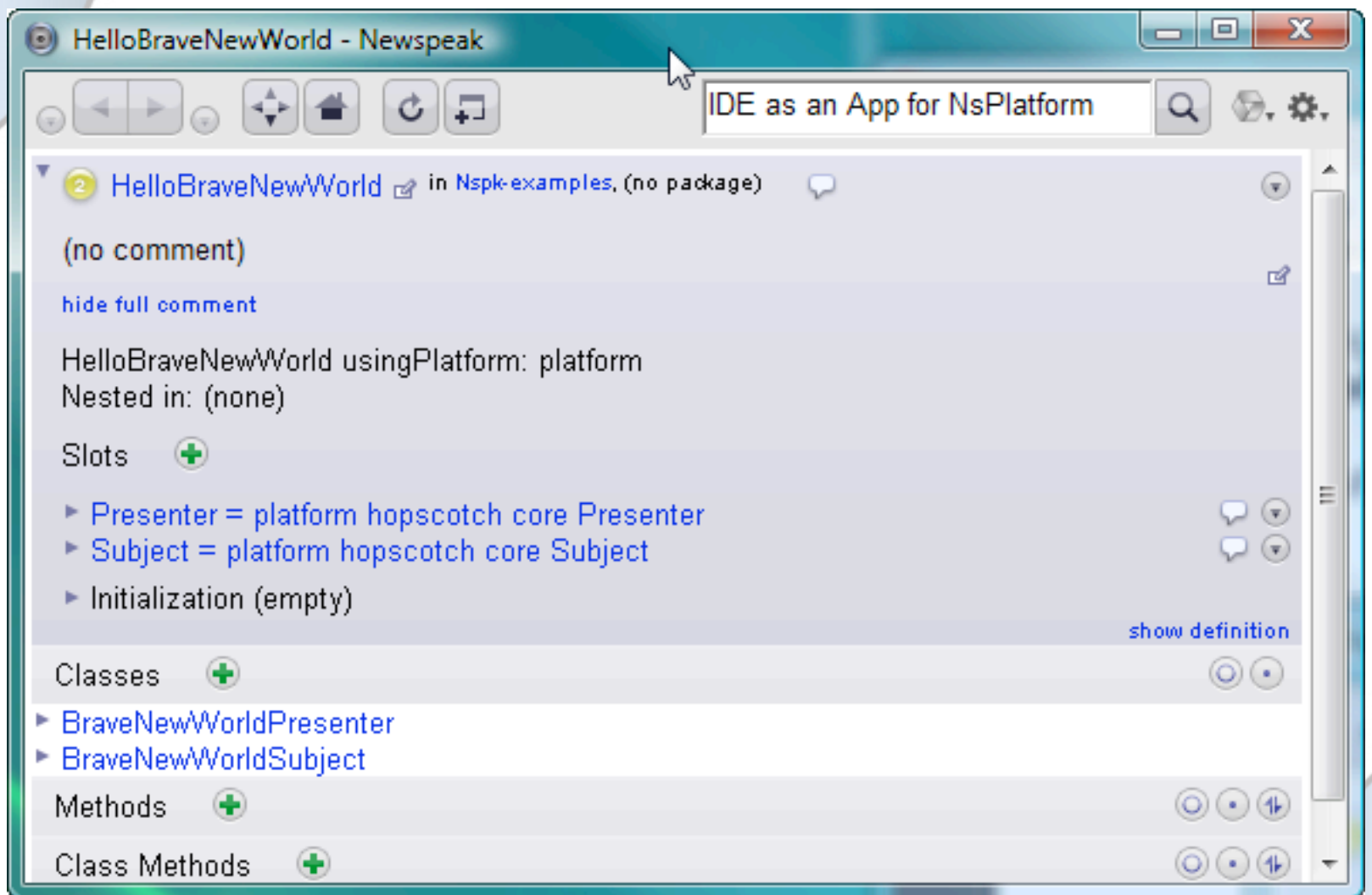
Multiple Implementations

- Modules are objects, accessed via an interface
- Different implementations can co-exist

Newspeak Provides

- Natural and powerful synergy between:
  - Message-based programming
  - Component style modularity
  - Virtual classes, mixins, class hierarchy inheritance
  - Object capability model and security
  - Mirror based reflection
  - Actor style concurrency
  - Pluggable types

Status



Status

- Available at *http://newspeaklanguage.org*
  - open source under Apache 2.0 license
- Work in Progress
  - Expect some tweaks to syntax and semantics
  - Still not complete

Dreams

- Cross-cutting concerns do not map to individual physical components
- Software lets us try and go beyond what hardware can do
- No physics to keep us honest

Mixins

- Can handle many problems, but not all
- However, the cases that go beyond mixins require very intrusive interfaces

Hope?

- Aspects are a poor attempt to capture rules that govern overall system interactions
- Poverty is unattractive, but real rule based systems are not

Related Work

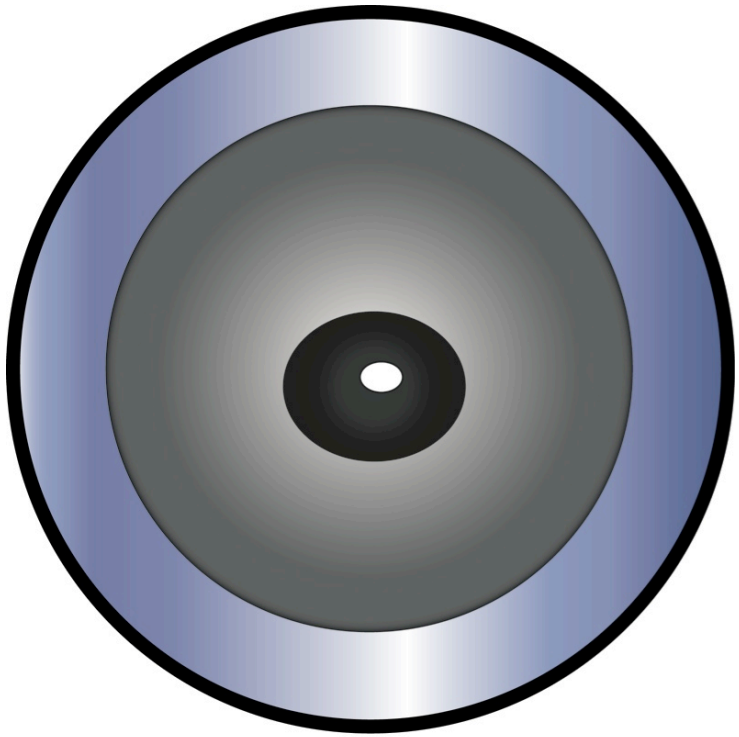
- Self, Smalltalk
- Beta, gBeta, Virtual Classes
- Ossher & Harrison OOPSLA 92
- Miller: E
- Racket/Units
- ML
- Steimann: Paradoxical Success of AOP
- Much more, but margin is too narrow ...

Credits

- Peter Ahe
- Vassili Bykov
- Yaron Kashai
- Bill Maddox
- Eliot Miranda

Volunteers

- Joshua Benuck
- Nikolay Botev
- Luis Diego Fallas
- Felix Geller
- John Hedditch
- Raffaello Giulietti
- Matthias Kleine
- Ryan Macnak
- Yardena Meymann
- Stephen Pair
- David Pennell
- Steve Rees
- Vadim Tsushko



Newspeak

It's double plusgood