

# Wollok: en el aula y más allá

Javier Fernandes<sup>1,2</sup>   Nicolás Passerini<sup>1,2,4</sup>

Pablo Tesone<sup>3,1,2,4</sup>   Débora Fortini<sup>1,4</sup>

Nahuel Palumbo<sup>4</sup>   Juan Contardo<sup>4</sup>   Carlos Raffellini<sup>4</sup>

<sup>1</sup>Universidad Nacional de Quilmes

<sup>2</sup>Universidad Nacional de San Martín

<sup>3</sup>Universidad Nacional del Oeste

<sup>4</sup>Universidad Tecnológica Nacional - F.R. Buenos Aires.

Workshop de Ingeniería en Sistemas y Tecnologías de la Información

19/09/2015

# Agenda

# Introducción

¿Por qué es difícil aprender OOP?

- Enfoque en un lenguaje particular
- Demasiados conceptos

```
package examples;  
  
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```

- **Entornos de desarrollo** limitados o inadecuados
- Aprender a programar exige **aprender a abstraer**

# Introducción

¿Por qué es difícil aprender OOP?

- Enfoque en un lenguaje particular
- Demasiados conceptos

```
package examples;  
  
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```

- **Entornos de desarrollo** limitados o inadecuados
- Aprender a programar exige **aprender a abstraer**

### 🦋Ozono

<http://ozono.uqbar-project.org/>

- Basado en Smalltalk
- **Recorrido incremental**
  - Metamodelo simplificado: sin clases ni herencia
  - Foco en objeto - mensaje - referencia - polimorfismo
- Herramientas de visualización de código
  - Diagramas de objetos / clases

### 🔲Gobstones

<http://www.gobstones.org/>

- Cuidadosa selección de los elementos sintácticos
- Elimina la necesidad de entrada-salida
- Separación entre elementos con efecto y elementos puros

### 🦋Ozono

<http://ozono.uqbar-project.org/>

- Basado en Smalltalk
- **Recorrido incremental**
  - Metamodelo simplificado: sin clases ni herencia
  - Foco en objeto - mensaje - referencia - polimorfismo
- Herramientas de visualización de código
  - Diagramas de objetos / clases

### 🔲Gobstones

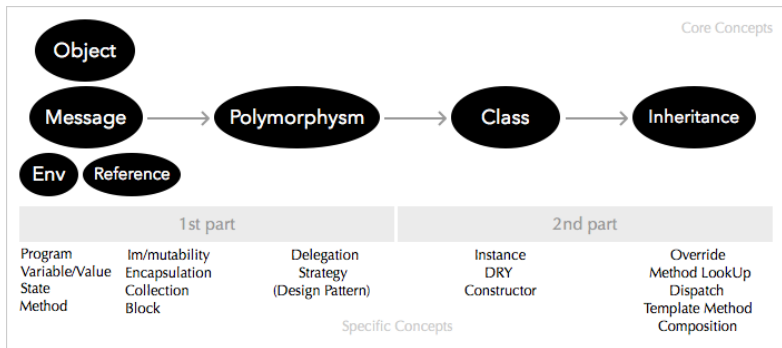
<http://www.gobstones.org/>

- Cuidadosa selección de los elementos sintácticos
- Elimina la necesidad de entrada-salida
- Separación entre elementos con efecto y elementos puros

# Introducción

¿Qué es WolloK?

- Lenguaje + muchas herramientas
- Optimizados para la enseñanza
- Cercanos a las herramientas profesionales *mainstream*



# Introducción

¿Qué es Wollok? - Un entorno optimizado para la enseñanza




- Sintaxis cuidada
  - selección de keywords
  - return obligatorio
  - énfasis en objetos y mensajes (aunque no todo es objeto mensaje)
- Combina object-based con class-based programming
- APIs minimalistas (ej: colecciones)
- Import system



# Introducción

Cuidado: No perderle pisada la evolución de las herramientas industriales

- Ambiente de objetos basado en archivos
- Framework de testing integrado
- Sintaxis concisa y "moderna"  
Ej: lambdas, literales para colecciones, excepciones, constructores
- Sistema de tipos *pluggeable* (en proceso)
- Mixins (planificado)

- OpenSource: LGPLv3
- Stack:  Eclipse XText + Xtend Lang
- SCM:
  - **Código:**  GitHub ([uqbar-project/wollok](https://github.com/uqbar-project/wollok))
  - **Build:** Maven + Tycho
  - **Continuous Integration:**  Travis
  - **Continuous Deployment**
  - **Coverage:** coveralls + jacoco
- Testing & TDD

- **GitFlow**
  - Feature Branches
  - Pull-Requests
  - *dev* → *master* ← *hotfixes*
- **Integration:**
  - Travis
  - compile, test, coverage, deploy
- **Deployment:**
  - **Productos** (IDE): multiples plataformas
  - **Update Sites**
  - **WDK**
  - 2 Ambientes: Stable & Dev

- 87% Cobertura
- **Runtime**
  - Testean ejecución
  - Interprete
  - **JUnit + iDSL**
- **Estáticos**
  - **Chequeos:** XPect
  - **Type System:** JUnit + iDSL
  - **Autocomplete:** XPect
  - **Formateo:** JUnit + iDSL
- **Pendientes**
  - Quick-Fixes
  - Refactors

## Testeo del Intérprete

```
class PostFixOperationTestCase extends AbstractWollokInterpreterTestCase {  
  
    @Test  
    def void testPlusPlus() {'''  
        program p {  
            var n = 1  
            n++  
            assert.that(n == 2)  
        }'''  
        .interpretPropagatingErrors  
    }  
}
```

### Testeo de Chequeos Estáticos

```
/* XPECT_SETUP org.uqbar.project.wollok.tests.xpect.WollokXPectTest END_SETUP */  
  
class Golondrina {  
    var energia = 100  
  
    method energia() {  
        // XPECT errors --> "Cannot assign a variable to itself. It does not have any effect" at "energia"  
        energia = energia  
    }  
}
```

- Debugger
- Diagramas: Clases y Objetos (en desarrollo)
- Tests
- ContentAssist y QuickFixes
- Sublime Plugins
- I18N

### Debugger

- UI integrada a Eclipse Debug
- Breakpoints: agregar, remover, deshabilitar, etc
- Step, into, out
- Inspeccionar variables
- Diagrama de Objetos



## Tests

The screenshot shows the 'Wollok Tests' window in an IDE. At the top, there are tabs for 'Problems', 'Javadoc', 'Declaration', 'Search', 'Console', 'Diagrams', and 'Wollok Tests'. Below the tabs, a summary bar shows 'Tests: 2', 'Run: 2', and 'Errors: 1'. The main area displays a list of test results for the file '.../wollok-dev/wollok-tests/src/wollok/tests/test.wtest'. The first test, 'Energia del Alpiste es 100', is marked with a green checkmark. The second test, 'Energia del Maiz es 200', is marked with a red 'X' and is highlighted in blue. Below the list, an 'Assert Error' message is displayed: 'Assert Error: Expected [100] but found [200] - Expected [100] - Actual [200]'.

Problems Javadoc Declaration Search Console Diagrams Wollok Tests

Tests: 2 Run: 2 Errors: 1

.../wollok-dev/wollok-tests/src/wollok/tests/test.wtest

- Energia del Alpiste es 100
- Energia del Maiz es 200

Assert Error: Expected [100] but found [200] - Expected [100] - Actual [200]

# Features Avanzados

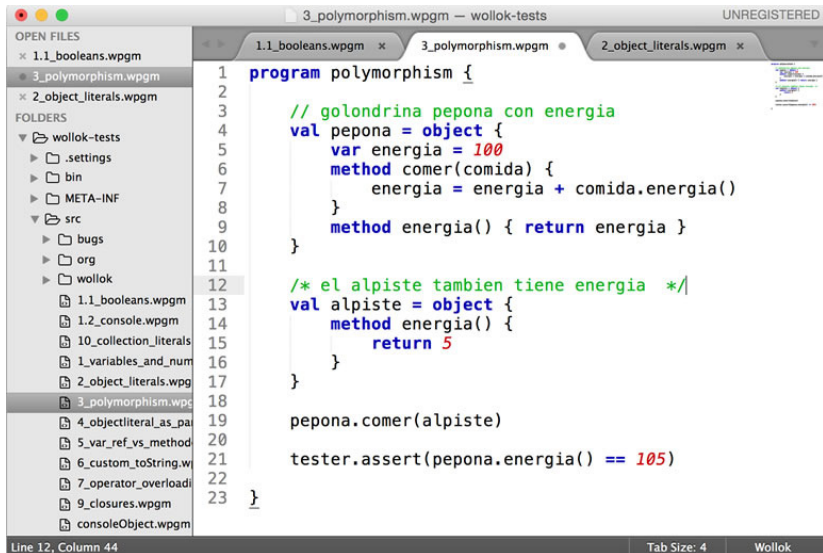
## Soporte para Sublime

### Soporte para Sublime

- WDK
  - No IDE
  - ~ 70MB (vs ~ 140)
  - Headless: wchecker, winterpreter, wtest
- Syntax highlight
- Templates
- Linter

# Sublime Support

## Syntax Highlight



```
1 program polymorphism {
2
3   // golondrina pepona con energia
4   val pepona = object {
5     var energia = 100
6     method comer(comida) {
7       energia = energia + comida.energia()
8     }
9     method energia() { return energia }
10  }
11
12  /* el alpiste tambien tiene energia */
13  val alpiste = object {
14    method energia() {
15      return 5
16    }
17  }
18
19  pepona.comer(alpiste)
20
21  tester.assert(pepona.energia() == 105)
22
23 }
```

# Sublime Support

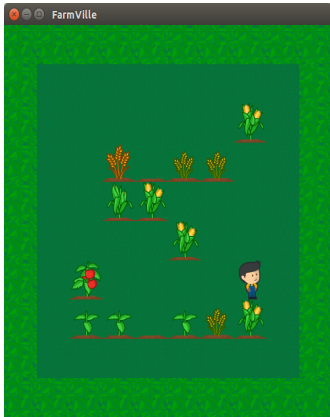
## Linter

The screenshot shows the Sublime Text editor interface with a project named 'wollok-tests'. The 'FOLDERS' sidebar on the left lists the project structure, including 'src' and 'wollok' subfolders. The 'OPEN FILES' sidebar shows three files: '1.1\_booleans.wpgm', '3\_polymorphism.wpgm', and '2\_object\_literals.wpgm'. The main editor window displays the code in '3\_polymorphism.wpgm'. The code includes a 'program' block with a 'comer' method on a 'pepe' object and an 'energia' method on an 'alpiste' object. A linting error tooltip is visible, showing three errors: 'Couldn't resolve reference to WReferenciable 'algo'', 'Couldn't resolve reference to WReferenciable 'pepe'', and 'Couldn't resolve reference to WReferenciable 'tester''. The status bar at the bottom indicates '1-2 of 3 errors: Couldn't resolve reference to WReferenciable 'pepe'; Couldn't resolve reference to WReferenciable 'algo', 2 lines, 1 characters selected'.

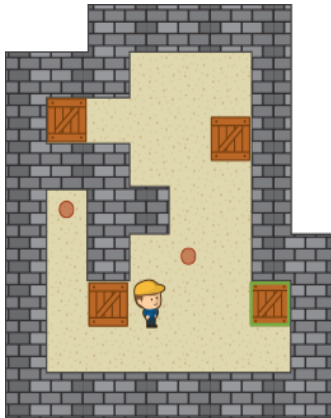
```
1 program {
2
3   12 Couldn't resolve reference to WReferenciable 'algo'
4   pepe.comer(algo)→
5
6   12 Couldn't resolve reference to WReferenciable 'pepe'
7   pepe.comer(algo)→
8
9   23 Couldn't resolve reference to WReferenciable 'tester'
10  tester.assert(pepona.energia() == 105)→
11
12 }
13
14 pepe.comer(algo)
15
16 /* el alpiste tambien tiene energia */
17 val alpiste = object {
18   method energia() {
19     return 5
20   }
21 }
22
23 pepona.comer(alpiste)
24
25 tester.assert(pepona.energia() == 105)
26 }
```

- Herramienta complementaria al testeo unitario y consola interactiva.
- Mejorar la comprensión de conceptos.
- Visualización de comportamiento
- Motivación en el aprendizaje fomentando la participación.

## FarmVille - Demo



## Sokoban - Demo



### Futuro

- + Tipos de **Juegos**
  - Survival
  - Por turnos
- + Tipos de **Interacciones**
- Features Gráficos
  - Animaciones
  - Fondos infinitos
  - Distintos vistas (lateral, isométrica, etc)



Los alumnos se apropian intuitivamente de las herramientas

- Integración class-based / object-based
- El REPL resulta más intuitivo que los workspaces de Smalltalk
- Mayor control sobre los tests unitarios
- Editores

Un **recorrido incremental** apoyado en **herramientas** adecuadas,  
permite aprovechar la **intuición** del estudiante  
fomentando su **autonomía, creatividad y motivación**

Los alumnos se apropian intuitivamente de las herramientas

- Integración class-based / object-based
- El REPL resulta más intuitivo que los workspaces de Smalltalk
- Mayor control sobre los tests unitarios
- Editores

Un **recorrido incremental** apoyado en **herramientas** adecuadas,  
permite aprovechar la **intuición** del estudiante  
fomentando su **autonomía, creatividad y motivación**

Los alumnos se apropian intuitivamente de las herramientas

- Integración class-based / object-based
- El REPL resulta más intuitivo que los workspaces de Smalltalk
- Mayor control sobre los tests unitarios
- Editores

Un **recorrido incremental** apoyado en **herramientas** adecuadas,  
permite aprovechar la **intuición** del estudiante  
fomentando su **autonomía, creatividad y motivación**

Los alumnos se apropian intuitivamente de las herramientas

- Integración class-based / object-based
- El REPL resulta más intuitivo que los workspaces de Smalltalk
- Mayor control sobre los tests unitarios
- Editores

Un **recorrido incremental** apoyado en **herramientas** adecuadas,  
permite aprovechar la **intuición** del estudiante  
fomentando su **autonomía, creatividad y motivación**

## Próximos Pasos

- Varias discusiones sobre la mejor sintaxis
- Herencia basada en mixins
- Implementar wollock-game en el aula
- Plataforma p/interacción Alumno  $\leftrightarrow$  Docente

Y muchas actividades para sumar más gente al proyecto.

¡Muchas Gracias!

