

Enseñando programación para el desarrollo profesional

Nicolás Passerini¹ Fernando Dodino^{1,2,3}
Javier Fernandes¹ Pablo Tesone⁴ Carlos Lombardi¹

¹Universidad Nacional de Quilmes

²Universidad Nacional de San Martín

³Universidad Nacional del Oeste

⁴Institut National de Recherche en Informatique et en Automatique.

Festival Latinoamericano de Instalación de Software Libre
Universidad Nacional de Quilmes

22/4/2017

Agenda

- 1 Introduccion
- 2 Features Avanzados
- 3 Wollok Game
- 4 Experiencia en el Aula
- 5 Desarrollo de Wollok
- 6 Próximos Pasos

Contexto

- Nos interesa la enseñanza de **programación con objetos**
- Principalmente en tecnicaturas e ingenierías
 - es decir: futuros desarrolladores de software
- Fundamentalmente universitarios
 - Pero también estamos trabajando con secundarios

Problema

- Bajos niveles de aprobación
- Suelen propagarse malas prácticas de programación
- Dificultades en la comprensión de los conceptos básicos
- Baja calidad del código producido

¿Por qué pasa eso?

- Capacidad de abstracción insuficiente
- Base matemática débil

Es nuestra responsabilidad cubrir esas falencias.

Problema

- Bajos niveles de aprobación
- Suelen propagarse malas prácticas de programación
- Dificultades en la comprensión de los conceptos básicos
- Baja calidad del código producido

¿Por qué pasa eso?

- Capacidad de abstracción insuficiente
- Base matemática débil

Es nuestra responsabilidad cubrir esas falencias.

Problema

- Bajos niveles de aprobación
- Suelen propagarse malas prácticas de programación
- Dificultades en la comprensión de los conceptos básicos
- Baja calidad del código producido

¿Por qué pasa eso?

- Capacidad de abstracción insuficiente
- Base matemática débil

Es nuestra responsabilidad cubrir esas falencias.

Problema

- Bajos niveles de aprobación
- Suelen propagarse malas prácticas de programación
- Dificultades en la comprensión de los conceptos básicos
- Baja calidad del código producido

¿Por qué pasa eso?

- Capacidad de abstracción insuficiente
- Base matemática débil

Es nuestra responsabilidad cubrir esas falencias.

¿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**

¿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**

Propuesta pedagógica

Objetivos

- **Inclusión**

Diseñar la currícula a partir de las características de los estudiantes.

- **Calidad**

A la vez que incrementar la calidad

- **Aplicabilidad**

Asegurando que los saberes impartidos sean aplicables en el ámbito profesional actual y futuro

- **Perspectiva futura**

Y liderando el desarrollo y la incorporación de nuevas ideas y tecnologías

Propuesta pedagógica

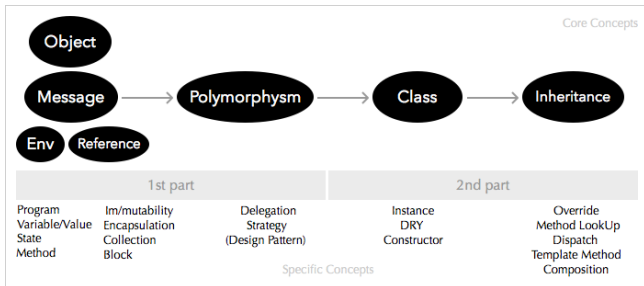
Pilares

- Temario y recorrido
- Seguimiento
- Deducción, intuición, autonomía
- Soporte de herramientas

Introducción

¿Qué es Wolok?

- Lenguaje + muchas herramientas
- Soporte para nuestro enfoque pedagógico
- Cercanos a las herramientas profesionales *mainstream*



Introducción

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

- Sintaxis cuidada
 - selección de keywords (ej: `const`, `method`)
 - `return` obligatorio
 - énfasis en objetos y mensajes ¹
- Combina *object-based* con *class-based programming*
- APIs minimalistas (ej: colecciones)
- *Import system*

¹¡Aunque no todo es objeto-mensaje!

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

Features Avanzados

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

Features Avanzados

Debugger

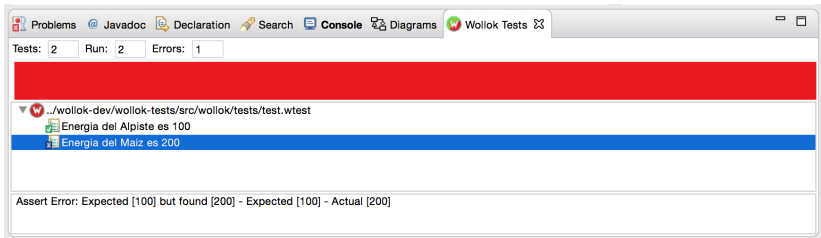
Debugger

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

Features Avanzados

Tests

Tests



Features Avanzados

Soporte para Sublime

Soporte para Sublime

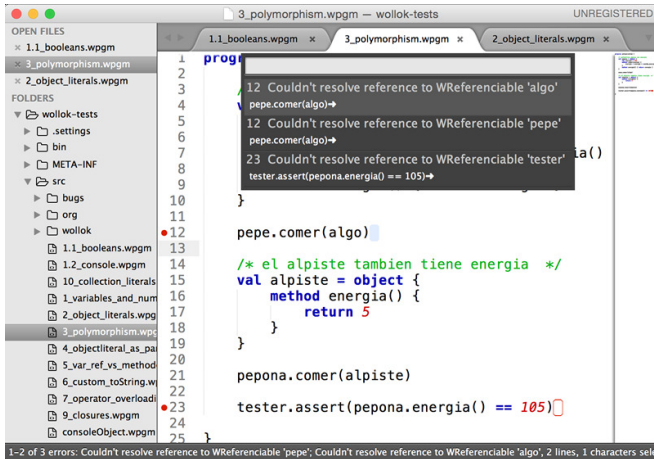
- WDK
 - No IDE
 - ~ 70MB (vs ~ 140)
 - Headless: wchecker, winterpreter, wtest
- Syntax highlight
- Templates
- Linter

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



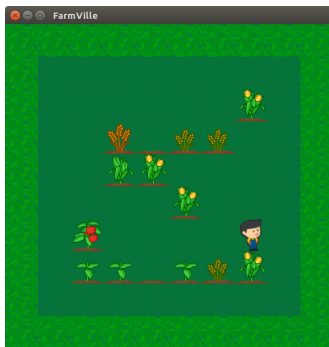
Wollok Game

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

Wollok Game

FarmVille

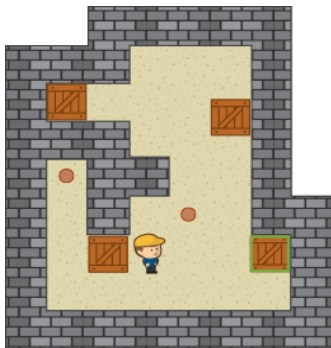
FarmVille - Demo



Wollok Game

Sokoban

Sokoban - Demo



Wollok Game

Futuro

Futuro

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

Experiencia en el Aula

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**

Experiencia en el Aula

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**

Experiencia en el Aula

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**




Experiencia en el Aula

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**

Desarrollo de Wollok

- 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

Desarrollo de Wollok

Continuous Integration & Deployment

- **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

Desarrollo de Wollok

Testing & TDD

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

Testing & TDD

Runtime

Testeo del Intérprete

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


Testing & TDD

Cheques Estáticos

Testeo de Cheques 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  
  }  
}
```

Próximos pasos

Próximos Pasos

- Nuevos tests
- Separación de void y nothing
- Nuevos constructores
- ... y muchas otras discusiones sobre la mejor sintaxis
- *Contract-based* programming

Y muchas actividades para sumar más gente al proyecto.

Muchas gracias

¡Muchas Gracias!

