

# Wollok: language + IDE for a gentle and industry-aware introduction to OOP

Nicolás Passerini<sup>1</sup>   Fernando Dodino<sup>1,2,3</sup>  
Javier Fernandes<sup>1</sup>   Pablo Tesone<sup>4</sup>   Carlos Lombardi<sup>1</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>Institut National de Recherche en Informatique et en Automatique.

Twelfth Latin American Conference on Learning Technologies  
Universidad Nacional De La Plata – Facultad de Informática  
11/10/2017

# Agenda

- 1 Introduccion
- 2 Wollok Language + IDE
- 3 Experiencia en el Aula

# 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:** modernización de los conceptos enseñados
  - Más alto nivel
  - Visión integral del desarrollo de software
- **Recorrido**
  - Postergar los conceptos más abstractos y/o secundarios
  - Priorizar la capacidad de diseñar
- **Seguimiento personalizado**
  - Ritmo de estudio guiado por el docente
  - La **autonomía** como parte de los objetivos de la materia
- Favorecer la **deducción** e **intuición**
- Necesitamos **herramientas de soporte** para estudiantes y profesores

# Propuesta pedagógica

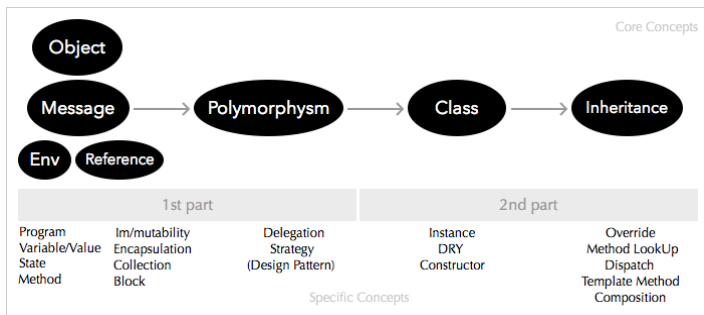
## Pilares

- **Temario:** modernización de los conceptos enseñados
  - Más alto nivel
  - Visión integral del desarrollo de software
- **Recorrido**
  - Postergar los conceptos más abstractos y/o secundarios
  - Priorizar la capacidad de diseñar
- **Seguimiento personalizado**
  - Ritmo de estudio guiado por el docente
  - La **autonomía** como parte de los objetivos de la materia
- Favorecer la **deducción** e **intuición**
- Necesitamos **herramientas de soporte** para estudiantes y profesores

# ¿Qué es Wollok?

## Introducción

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



# ¿Qué es Wollok?

Un entorno optimizado para la enseñanza

- **Sintaxis** diseñada para la enseñanza
  - selección de keywords (ej: `const`, `method`)
  - `return` obligatorio
  - énfasis en objetos y mensajes <sup>1</sup>
- **Reducir características poco adecuadas** a un principiante
  - Constructores simplificados (proximamente)
  - No tiene *reflection*
  - No usamos entrada / salida
- Combina *object-based* con *class-based programming*
- *Import system*
- Testing integrado

---

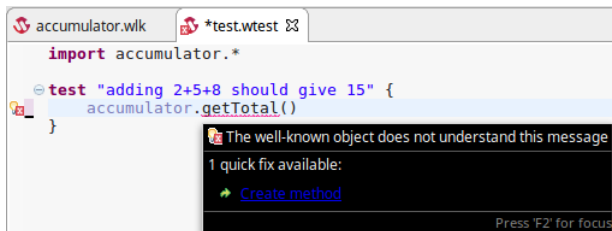
<sup>1</sup>¡Aunque no todo es objeto-mensaje!

# ¿Qué es Wollok?

Cuidado: No perderle pisada la evolución de las herramientas industriales

- Ambiente de objetos **basado en archivos**
- **Sintaxis** concisa y "moderna"  
Ej: lambdas, literales para colecciones, excepciones, constructores
- **Sistema de tipos** *pluggeable* (en proceso)
- Mixins
- **Testeo** automático integrado.
- **Versionado y trabajo en grupo** (básico, git).
- **Debugger** (en proceso).

## Wollok IDE Core Features – Error reporting



- Resalta los errores en el código y a medida que se escribe.
- Errores comprensibles para el estudiante.
- Internacionalizado.
- Favorece la autonomía en el aprendizaje.

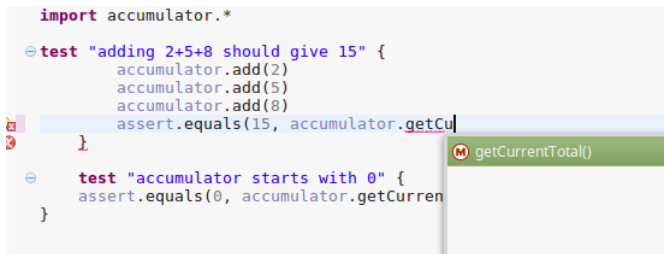


## Wollok IDE *Core Features – Content assist*

```
import accumulator.*

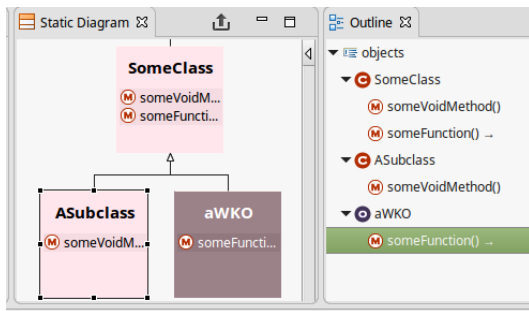
test "adding 2+5+8 should give 15" {
    accumulator.add(2)
    accumulator.add(5)
    accumulator.add(8)
    assert.equals(15, accumulator.getCu
}

test "accumulator starts with 0" {
    assert.equals(0, accumulator.getCurren
}
```

A screenshot of the Wollok IDE interface. It shows a code editor with two test cases. The first test case, "adding 2+5+8 should give 15", is partially completed. The cursor is at the end of the line "assert.equals(15, accumulator.getCu". A content assist popup is visible, showing a green button with a magnifying glass icon and the text "getCurrentTotal()". The second test case, "accumulator starts with 0", is also partially completed with "assert.equals(0, accumulator.getCurren".

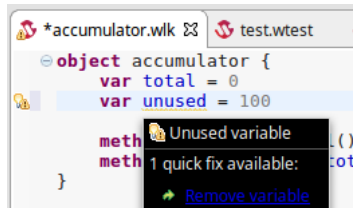
- Guía al estudiante, mostrando opciones para continuar.
- Reduce la necesidad de memorizar sintaxis y *APIs*.
- Evita errores de tipeo.
- **Permite concentrarse en cuestiones de más alto nivel.**

## Wollok IDE *Core Features* – Herramientas de visualización



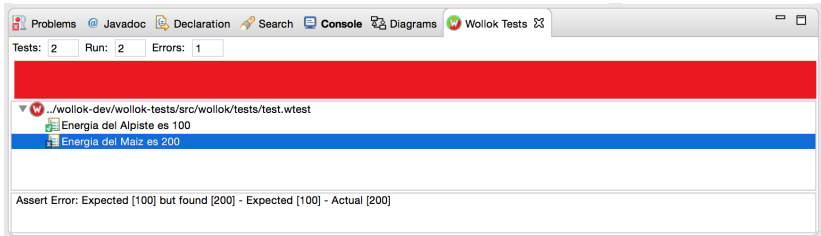
- Diagramas estáticos automáticos y *customizables*.
- *Outline*.
- (Próximamente) diagramas dinámicos.
- **Proveen una vista de más alto nivel del programa.**

## Wollok IDE Core Features – Quick Fixes



- Soluciones automatizadas para errores comunes.
- Aceleran la resolución de problemas simples y/o secundarios.
- Favorecen el *Test Driven Development*.
- Evita perder el foco de la tarea en curso.

# Wollok IDE *Core Features* – Tests integrados



- Se incorpora el testing como una parte esencial de la práctica de la programación.

# Wollok IDE *Core Features*

- **Refactorings**

Simplifican modificaciones sencillas al código, favoreciendo las metodologías de trabajo iterativas.

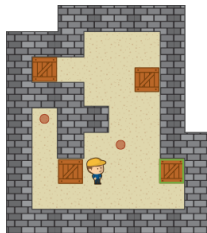
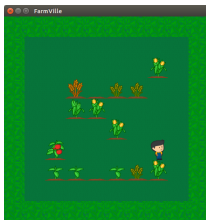
- **Formateo automático**

Ayuda al estudiante a incorporar buenas prácticas de organización espacial de código.

- Soporte para editores livianos (Sublime) y web (Mumuki, otros en proceso)

# Wollok Game

- Permite construir pantallas interactivas muy sencillas
- Complementa al testeo unitario y consola interactiva
- Se evita la utilización de E/S
- Motivación en el aprendizaje fomentando la participación



# 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**

# Muchas gracias

## ¡Muchas Gracias!



- Para saber más  
[www.wollok.org/](http://www.wollok.org/)
- Para colaborar con el desarrollo  
[github.com/uqbar-project/wollok](https://github.com/uqbar-project/wollok)
- Para participar en las discusiones  
[groups.google.com/forum/#!forum/wollok-docentes](https://groups.google.com/forum/#!forum/wollok-docentes)