|  |  |
| --- | --- |
| **Universidad de Burgos**  Escuela Politécnica Superior | |
| Grado en Ingeniería Informática  Caso de estudio refactorización (LANSIMULATION) | |
| Alumnos | Elvis dos Santos Orfão  Marcos Orive Izarra |
| Tutor | Carlos López Nozal  Departamento de Ingeniería Civil  Área de Lenguajes y Sistemas Informáticos |
| Burgos, 27 de mayo de 2018 | |

Este documento está licenciado bajo [Creative Commons Attribution 3.0 License](http://creativecommons.org/licenses/by/3.0/)

Índice

Tabla de contenido

[1. Preguntas de reflexión 3](#_Toc515211474)

[2. Preguntas de la guía 3](#_Toc515211475)

[Ojear la documentación 3](#_Toc515211476)

1. URL al repositorio

<https://github.com/moi0006/reafactoring_lab_session>

1. Preguntas de reflexión

* ¿Se puede automatizar completamente el proceso de refactorización a través de herramientas?

No, no todas las refactorizaciones pueden ser automatizadas. Hemos tenido que hacer algunas a mano. Como por ejemplo, mover y cambiar los métodos printOn, printHTMLOn y printXMLOn a las clases Printer y Workstation.

* ¿Qué relación encuentras entre el proceso de refactorización y la utilización de sistemas de control de tareas y versiones?

Gracias a los sistemas de control de tareas podemos abrir tareas/issues que detallen las refactorizaciones que queremos realizar.   
El control de versiones no ayuda a ver código duplicado, bad smells, etc y así poder corregirlo.

* **¿Existen las subclases Workstation y Printer?**

Sí, las hemos añadido con el propósito de usar herencia y polimorfismo.

* **¿Workstation y Printer tienen implementados correctamente tres métodos: printOn, printHtmlOn, printXmlOn ?**

Sí, funcionan correctamente y pasa los test.

* **¿Se elimina el código de tipo de nodo?**

Sí, ahora se distingue usando las clases Printer y Workstation.

1. Preguntas de la guía

## Ojear la documentación

¿Cuál es tu primera impresión sobre el sistema?

La primera impresión es que la clase Network tiene muchos más métodos que Packet y Node. Parece que concentra mucha más responsabilidad.

¿Dónde centrarías tus esfuerzos de refactorización?

Seguramente habrá que refactorizar muchas cosas de la clase network, el resto parece que tienen demasiado poco código.

## Lee todo el código en 5 minutos.

Cuál es la segunda impresión sobre el sistema? ¿Estas de acuerdo con la impresión inicial con este nuevo conocimiento sobre el código? ¿dónde centrarías tus esfuerzos de refactorización? Discutir con los miembros del equipo*.*

Efectivamente podríamos considerar la clase Network como una God class. Además Node tiene problema de Herencia simulada, que habrá que arreglar.

Sí estamos de acuerdo, la clase network sigue siendo el mayor problema y es donde centraríamos nuestros esfuerzo de refactorización.

## Hacer una instalación de prueba

¿Crees que el código base esta ya refactorizado? ¿Qué puedes decir de la calidad de los tests: puedes empezar a refactorizar de manera segura? Discutir con los miembros del equipo*.*

Dudamos mucho que el código ya estuviera refactorizado. Los tests no son tampoco de gran calidad, pero en teoría no deberíamos tocar los tests a la hora de refactorizar pero ha sido necesario realizar unas pequeñas modificaciones para refactorizar con seguridad.

## Habla con los de mantenimiento

Desarrolla un plan de proyecto listando a)los riesgos, b) las oportunidades de refactorización (detección de defectos), c) las actividades (plan de refactorizaciones).

1. Riesgos:

No debería haber demasiados riesgos si refactorizamos con cabeza. No se trata de cambiar el código radicalmente si no de mejorarlo para que sea más legible y las responsabilidades se cumplan mejor

1. Oportunidades de refactorización:

Hay bastante código duplicado que podemos extraer en métodos, hay que corregir la herencia simulada de la clase Node y mover algunos métodos de Network a Node y/o packet para corregir errores de envidia de características.

1. Actividades:

Extraer, renombrar y mover métodos. Mover y renombrar variables.

## Extract Method

¿Estas seguro que estas refactorizaciones no rompen el código?¿Crees que estas refactorizaciones merecen la pena? ¿La herramienta de refactorización hace un buen trabajo? Discutir con los miembros del equipo.

## Mover el comportamiento cerca de los datos

Estas seguro que estas refactorizaciones no rompen el código?¿Crees que estas refactorizaciones merecen la pena? ¿La herramienta de refactorización hace un buen trabajo? Discutir con los miembros del equipo.

## Eliminar código de navegación

¿Estas seguro que estas refactorizaciones no rompen el código?¿Crees que estas refactorizaciones merecen la pena? ¿La herramienta de refactorizción hace un buen trabajo? Discutir con los miembros del equipo.

## Transformar códigos de tipo

¿Estas seguro que estas refactorizaciones no rompen el código?¿Crees que estas refactorizaciones merecen la pena? ¿La herramienta de refactorización hace un buen trabajo? Discutir con los miembros del equipo.

## Conclusion

» ¿Hay asuntos que no has considerado ? ¿Hay refactorizaciones que parecen innecesarias ? Discutir con los miembros del equipo.