



UNIVERSIDADE DA CORUÑA

Monitorización de pruebas VVS

Historial de revisiones

Fecha	Versión	Descripción	Autores
16/12/2015	1.0	Ejercicio de refactorización	Xoán Andreu Barro Torres F. Javier Moure López Emma Oitavén Carracedo

Índice

1. Contexto	2
2. Estado actual	2
3. Registro de pruebas	2
3.1. JUnit	2
3.2. Quickcheck	2
3.3. Cobertura	2
3.4. PIT (mutation testing)	3
3.5. Pruebas estáticas/estructurales: FindBugs	3
3.6. Pruebas estáticas/estructurales: CheckStyle	4
4. Registro de errores	5
5. Estadísticas	5
6. Otros aspectos de interés	5

1. Contexto

Este documento hace referencia a las pruebas realizadas sobre el proyecto de la asignatura VVS llamado Spoticopy¹ encontrado en el repositorio de GitHub.

2. Estado actual

Listaxe de funcionalidades actuais, as súas especificacións, as persoas responsables do seu desenvolvemento, e as persoas responsables do proceso de proba. Para cada funcionalidade: número de probas obxectivo, número de probas preparadas, porcentaxe executada e porcentaxe superada. Se esta información é profusa e se almacena noutra fonte, referencia á fonte. Se é cambiante, referencia a unha *shapshot* ou resumo do mais destacado.

3. Registro de pruebas

3.1. JUnit

JUnit se utiliza para realizar pruebas unitarias sobre nuestra aplicación, nos sirven para encontrar errores y solventar los problemas en la programación de forma manual. Se realizan pruebas de todas las funciones implementadas en la aplicación.

3.2. Quickcheck

QuickCheck es una implementación de la herramienta de prueba basada especificación QuickCheck . El objetivo de QuickCheck es sustituir los valores recogidos manualmente con los valores generados . Una prueba basada en QuickCheck trata de cubrir las leyes de un dominio , mientras que la prueba clásica sólo puede probar la validez de los valores distintos. Es una mejora de Junit y podemos encontrar problemas de rendimiento con las iteraciones.

Se crean generadores de:

- GeneradorContenido
- GeneradorCancion
- GeneradorServidor
- GeneradorServidorVacio

Con estos generadores, se comprueban las funciones de Servidor normal y que todo está funcionando correctamente.

3.3. Cobertura

Realizada la prueba de cobertura de pruebas, se observa que faltan muchos test por implementar, por lo que se procede a implementar los test que faltan, según la información que nos ofrece el plugin de ecobertura.

¹<https://github.com/andreu-barro/VVS>

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
▼ spoticopy	61,7 %	512	318	830
▼ src/main/java	50,2 %	317	315	632
▼ es.udc.fic.vvs.spoticopy.token	25,2 %	31	92	123
> Token.java	25,2 %	31	92	123
▼ es.udc.fic.vvs.spoticopy.contenido	41,7 %	98	137	235
> Emisora.java	27,8 %	37	96	133
> Anuncio.java	51,7 %	31	29	60
> Cancion.java	71,4 %	30	12	42
▼ es.udc.fic.vvs.spoticopy.servidor	68,6 %	188	86	274
> ServidorGenerico.java	41,4 %	24	34	58
> ServidorConPadres.java	59,6 %	68	46	114
> ServidorNormal.java	94,1 %	96	6	102
> src/test/java	98,5 %	195	3	198

Después de:

- Aumentar cobertura de Anuncio (casi al 100)
- Aumentar cobertura de ServidorNormal (al 100)
- Aumentar cobertura de ServidorGenerico (casi al 100)
- Aumentar cobertura de Token
- Aumentar cobertura de ServidorConPadres

Quedó:

Total Coverage: 93,68 %				
Filename	Coverage	Total	Not Executed	
es.udc.fic.vvs.spoticopy.contenido.Contenido	0,00 %	0	0	0 ^
es.udc.fic.vvs.spoticopy.servidor.Servidor	0,00 %	0	0	0
es.udc.fic.vvs.spoticopy.token.Token	83,33 %	36	6	6
es.udc.fic.vvs.spoticopy.servidor.ServidorGenerico	89,47 %	19	2	2
es.udc.fic.vvs.spoticopy.servidor.ServidorConPadres	92,86 %	28	2	2
es.udc.fic.vvs.spoticopy.contenido.Anuncio	94,74 %	19	1	1
es.udc.fic.vvs.spoticopy.contenido.Cancion	100,00 %	15	0	0
es.udc.fic.vvs.spoticopy.contenido.Emisora	100,00 %	32	0	0
es.udc.fic.vvs.spoticopy.servidor.ServidorNormal	100,00 %	25	0	0
Total	93,68 %	174	11	11 v

3.4. PIT (mutation testing)

Realizado el mutation testing, resultados:
Informe PIT

3.5. Pruebas estáticas/estructurales: FindBugs

FindBugs es un programa que utiliza el análisis estático para buscar errores en el código de Java.

En la versión inicial podemos comprobar que tenemos los siguientes errores: Informe Find bugs

En el informe podemos comprobar que tenemos 9 bugs, y distinguimos dos categorías

donde tenemos los problemas: Malas prácticas y problemas de estilo. También distinguimos la prioridad: Alta, baja y media. Los primeros errores a solventar son los de criticidad alta, seguidamente de los medios y por último los de prioridad baja.

Errores encontrados:

- **ST WRITE TO STATIC FROM INSTANCE METHOD:** En la clase token existía un método que escribía en una variable estática, esto es una mala práctica cuando está siendo manipulado por varias instancias.
- **RI REDUNDANT INTERFACES:** ServidorNormal y ServidorConPadres implementa la misma interfaz que la superclase.
- **BC EQUALS METHOD SHOULD WORK FOR ALL OBJECTS:** El método equals (Object o) no debe hacer ninguna suposición sobre el tipo de o. Simplemente debe devolver false si o no es del mismo tipo que esta. La clase Anuncio asume el argumento es de tipo anuncio.
- **HE EQUALS USE HASHCODE:** La clase anuncio Esta clase anula Equals (Object) , pero no anula hashCode () , y hereda la implementación de hashCode () de java.lang.Object (que devuelve el código hash de identidad, un valor arbitrario asignado al objeto por el VM) . Por lo tanto , es muy probable que violaría el invariante que los objetos iguales deben tener iguales hashcodes la clase.
- **NP EQUALS SHOULD HANDLE NULL ARGUMENT:** En la clase anuncio, el método no funciona para cuando el objeto es nulo.

Después de identificar los errores, resueltos los problemas mencionados y comprobamos el resultado:

Spoticopy
Last Published: 2015-12-15 | Version: 0.0.1-SNAPSHOT

FindBugs Bug Detector Report

The following document contains the results of FindBugs

FindBugs Version is 3.0.1

Threshold is low

Effort is max

Classes	Bugs	Errors	Missing Classes
9	0	0	0

Files

Class	Bugs
-------	------

Copyright © 2015. All Rights Reserved.

3.6. Pruebas estáticas/estructurales: CheckStyle

Checkstyle es una herramienta de desarrollo para ayudar a los programadores escribir código Java que se adhiere a un estándar de codificación. Para comprobar el estilo, pasamos la herramienta CheckStyle a nuestra aplicación y comprobamos el resultado: Informe CheckStyle

Al pasar la herramienta de CheckStyle descubrimos que nuestra aplicación tiene unos 236 errores de estilo, es decir, que no cumple el estandar de programación java.

Resumen de errores encontrados:

- **Faltan comentarios:** En la mayoría de clases faltan comentarios javadoc. Se añaden.

- Mala indexación código y espacios: Se reestructura el código para que solventar dichos errores.

Se revisa el informe con los 236 errores de estilo y se eliminan los 236 errores de estilo.

Spoticopy
Last Published: 2015-12-15 | Version: 0.0.1-SNAPSHOT

Checkstyle Results

The following document contains the results of Checkstyle 6.11.2 with checkstyle_custom.xml ruleset. [View](#)

Summary

Files	Info	Warnings	Errors
9	0	0	0

Files

File	I	W	E
------	---	---	---

Rules

Category	Rule	Violations	Severity
----------	------	------------	----------

Details

4. Registro de errores

Todos los errores localizados, modificaciones necesarias, etc. pueden encontrarse referenciados en el documento de CHANGELOG.txt, en el cual se encuentran las correcciones hasta el momento.

5. Estadísticas

- Errores diarios encontrados:
- Errores semanales encontrados:
- Progreso en las pruebas:
- Análisis del perfil de detección de errores (lugares, componentes, tipología).
- Informe de errores abiertos y cerrados por nivel de criticidad.
- Evaluación global del estado de calidad y estabilidad actuales.

6. Otros aspectos de interés

Nada de momento. Se conserva el apartado para el futuro.