TEC | Tecnológico de Costa Rica

Seguridad del Software - IC8071

Proyecto 3 - Análisis estático mediante anotaciones en el lenguaje Java (JML)

Profesor:

Herson Esquivel Vargas

Integrantes:

Leonardo Céspedes Tenorio - 2022080602

Kevin Chang Chang - 2022039050

Fecha de Entrega: 1 de Noviembre

II Semestre 2024

Documentación de Proyecto con OpenJML

1. **¿Considera que usando OpenJML descubrió todos los posibles problemas? ¿Está segura(o) de que el código es correcto después de aplicar los cambios?**

   OpenJML es eficaz para identificar inconsistencias en el código y asegura que las especificaciones JML coincidan con la implementación, lo que reduce significativamente los errores. Sin embargo, no garantiza la detección de todos los posibles problemas. Las herramientas de verificación automática tienen limitaciones inherentes, como la dependencia de los contratos y condiciones especificados. Personalmente sentimos que todos los errores fueron resueltos en ambos Amount.java y Bag.java dado a que su código fuente era relativamente corto por lo que pudimos identificar y analizar cada línea de código individualmente. Para un programa más grande pensamos que es posible que hayan errores que no se encuentren con OpenJML y hay más posibilidad de que errores pasen desapercibido. Concluímos que el código es correcto en su totalidad y OpenJML nos ayudó a identificar todos los errores en los códigos fuente.

2. **¿Qué cambios sugeriría para mejorar tanto la herramienta (OpenJML) como el lenguaje (JML)?**

   Nos hubiera gustado que la herramienta indicara los errores de una forma más clara y específica. Saber exactamente qué se debería cambiar hubiera sido de gran ayuda. Los errores que se mostraban al final eran útiles, pero también eran limitados en la cantidad de información que nos daba. Para OpenJML, también sería útil optimizar el análisis de flujo de datos y los modelos de prueba generados, lo que permitiría una verificación más exhaustiva en entornos complejos y con alta concurrencia (como con uso de hilos de ejecución). En cuanto al lenguaje JML, la incorporación de expresiones condicionales simplificaría la especificación de comportamientos más complejos por lo que se podría ser más específico al anotar un código en Java con JML. En general sugerimos mejorar la automatización y facilitar y simplificar aún más su uso.

3. **En lugar de OpenJML ¿qué otra forma (manual o automática, formal o no) de encontrar los problemas identificados por OpenJML propondría usted? La alternativa propuesta**

   Una alternativa es el uso de pruebas basadas en propiedades con herramientas como QuickCheck, que generan casos de prueba aleatorios basados en propiedades especificadas del código. Esto permitiría una cobertura de prueba diversa y encontraría posiblemente una cantidad similar o incluso mayor de problemas, especialmente en casos límite.

   - **¿Encontraría más, menos o igual cantidad de problemas?**

     Es posible que QuickCheck encuentre igual o más problemas, esto se debe a que las pruebas basadas en propiedades generan automáticamente muchos casos de prueba aleatorios y variados, cubriendo situaciones que pueden no estar contempladas en los contratos de JML. Una consideración que tomar en cuenta es que las pruebas basadas en propiedades pueden pasar por alto ciertos errores formales que OpenJML detectaría, ya que no se enfocan en contratos explícitos. Es posible que QuickCheck encuentre más problemas pero OpenJML puede ser más específico o enfocado en los problemas que encuentra.

   - **¿Encontraría los problemas antes o después que OpenJML?**

     Con la herramienta QuickCheck, las pruebas basadas en propiedades suelen identificar problemas a medida que se ejecuta el código en diversos contextos, lo cual podría implicar que ciertos problemas aparezcan en una fase más avanzada del desarrollo. Sin embargo, al inicio, OpenJML podría ser más rápido en detectar errores relacionados con contratos y condiciones predefinidas. Sabiendo esto, probablemente encontraría los problemas después de OpenJML.

   - **¿Requeriría más o menos trabajo?**

     Implementar pruebas basadas en propiedades como lo hace QuickCheck, requiere una inversión inicial para definir las propiedades adecuadas, y puede ser más demandante que configurar OpenJML en un entorno bien definido. OpenJML, al ser una herramienta de verificación formal, demanda menos

esfuerzo en la ejecución, aunque los contratos JML deben especificarse meticulosamente, lo cual también conlleva un esfuerzo considerable. La cantidad de trabajo es similar pero tiene un enfoque diferente en cada herramienta.

- **¿Proveería más o menos confianza sobre la seguridad del código resultante?**

  Trabajando en este proyecto, identificamos que OpenJML proporciona una alta confianza en el cumplimiento de los contratos específicos establecidos para el código, lo cual es valioso en términos de seguridad formal. No obstante, las pruebas basadas en propiedades, al ampliar la cobertura, pueden generar una confianza adicional al demostrar el comportamiento del código en casos no contemplados por los contratos. En conjunto, ambas aproximaciones podrían dar una seguridad robusta en el resultado final, siendo ideal un enfoque complementario entre verificación formal y pruebas basadas en propiedades.

Decidir cuál herramienta usar dependería mucho en el tipo de proyecto que se trabaja y en qué aspectos del código se quiere aplicar mayor esfuerzo.