Tercer proyecto: Análisis estático mediante anotaciones en el lenguaje Java
v. 1.0

"If the code and the comments disagree, then both are probably wrong."
- Norm Schryer

Descripción general

El propósito de este proyecto es experimentar con la verificación formal de programas escritos en el lenguaje Java. Para esto, se usará el lenguaje Java Modeling Language (JML) y la herramienta OpenJML que valida especificaciones escritas en JML.

Las especificaciones de JML se escriben como comentarios de Java aunque con un formato especial. De este modo, dichas especificaciones son ignoradas por el compilador de Java y no afectan la generación de bytecode. La sintaxis de las especificaciones JML está disponibles en https://www.openjml.org/tutorial/Syntax.

El programa de verificación OpenJML

OpenJML es un programa de verificación automática de programas escritos en el lenguaje Java. Dichas verificaciones pueden ser estáticas o dinámicas. La verificación estática también es llamada Deductive Verification (DV) o Extended Static Checking (ESC). La verificación dinámica es conocida como Runtime Assertion Checking (RAC). El alcance de este proyecto cubre únicamente la verificación estática.

La verificación estática se basa en técnicas de ejecución simbólica (¿recuerdan angr?). Por este motivo, los resultados del análisis estático son más concluyentes que los resultados de la verificación dinámica que usa unos pocos valores para ejecutar el programa. Durante la verificación estática, cada método se verifica usando las especificaciones (no las implementaciones) de los otros métodos usados.

Para la resolución de este proyecto se recomienda seguir la documentación oficial de OpenJML disponible en https://www.openjml.org/tutorial/, con especial atención a la sección titulada "Simple Method Specifications". Esta sección incluye la especificación de precondiciones, postcondiciones, aritmética, ciclos, entre otros.

Descripción específica

A continuación se detallan las actividades esperadas como parte de la ejecución de este proyecto.

Código por analizar

Se analizará dos archivos fuente:

- `Bag.java`: clase para representar una estructura de datos Bag (“bolsa”) para almacenar enteros.
- `Amount.java`: clase para representar una cierta cantidad de dinero.

Análisis de código fuente

El análisis del código fuente se realizará usando OpenJML. Esta herramienta se puede descargar desde la dirección https://sourceforge.net/projects/jmlspecs/files/latest/download. Después de descargar y descomprimir la herramienta, se debe invocar desde la terminal el archivo ejecutable `openjml`. Para validar que funciona correctamente se puede ejecutar de la siguiente forma:

```
$ ./openjml -version
openjml 0.17.0-alpha-15
```

Una vez que se ha corroborado su funcionamiento, se recomienda invocarlo con los archivos de demostración que la aplicación provee. En el primer caso, hay una clase que está escrita de forma incorrecta por lo que `openjml` imprime la siguiente salida:

```
$ ./openjml -esc ./demos/MaxBad.java
./demos/MaxBad.java:7: verify: The prover cannot establish an assertion
(Postcondition: ./demos/MaxBad.java:3:) in method max
    return t > k ? i : k;
    ^
./demos/MaxBad.java:3: verify: Associated declaration: ./demos/MaxBad.java:7:
  //@ ensures \result >= i && \result >= j && \result >= k;
      ^
2 verification failures
```

En el segundo caso, ya se ha corregido el problema de la clase anterior por lo que `openjml` no imprime ninguna salida:

```
$ ./openjml -esc ./demos/Max.java
```

Actividades

Ejecute la herramienta OpenJML para analizar los archivos asignados (Bag.java y Amount.java). Si la herramienta produce algún mensaje, se debe:

- anotar el código (e.g., invariantes, precondiciones, postcondiciones), o bien
- corregir los bugs en el código

hasta que los mensajes desaparezcan. Use su buen juicio para determinar cuándo aplicar una u otra

opción. Hay algunos bugs insertados de forma deliberada en el código para ser detectados con ayuda de la herramienta. No cambie el código cuando no haya un bug real.

Para el caso de `Amount.java`, se debe especificar formalmente las invariantes que se discuten en el texto del encabezado del archivo en forma de comentarios. La especificación formal de estas invariantes revelará algunos problemas adicionales en el código.

Al corregir los archivos fuente, se espera que OpenJML se ejecute sin generar mensajes de salida. Hay instrucciones más detalladas en los comentarios del encabezado de cada clase. Léalos detenidamente.

Documentación

La documentación del proyecto debe responder a las siguientes preguntas:

1. ¿Considera que usando OpenJML descubrió *todos* los posibles problemas? ¿Está *segura(o)* de que el código es correcto después de aplicar los cambios?
2. ¿Que cambios sugeriría para mejorar tanto la herramienta (OpenJML) como el lenguaje (JML)?
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
    1. ¿encontraría más, menos o igual cantidad de problemas?
    2. ¿encontraría los problemas antes o después que OpenJML?
    3. ¿requeriría más o menos trabajo?
    4. ¿proveería más o menos confianza sobre la seguridad del código resultante?

Aspectos administrativos

Rúbrica

Item	Valor	Comentario
Corrección de `Bag.java`	40%	Al menos 20 anotaciones y/o correcciones al código claramente señaladas por medio de comentarios. Cada anotación y/o corrección al código vale 2%. OpenJML no debe generar errores. Si OpenJML genera errores la calificación será de 0%. Cada uso del keyword `assume` restará 10%.
Corrección de `Amount.java`	40%	Al menos 20 anotaciones y/o correcciones al código claramente señaladas por medio de comentarios. Cada anotación y/o corrección al código vale 2%. OpenJML no debe generar errores. Si OpenJML genera errores la calificación

		será de 0%. Cada uso del keyword `assume` restará 10%.
Documentación	20%	Respuesta a las preguntas de reflexión.

Entregables

- Los dos archivos fuente de Java corregidos y anotados con especificaciones JML.
- La documentación descrita anteriormente en formato PDF. Únicamente se revisará documentación en formato PDF.
- Los 3 archivos mencionados anteriormente (fuentes de java y la documentación en PDF) se deben subir al TecDigital en un único archivo zip.

Fecha de entrega

- Viernes 12 de mayo de 2023 a las 11:59pm.

Aspectos varios

- El proyecto se desarrollará en parejas.
- Todas las dudas que surjan con respecto al proyecto se preguntarán y evacuarán en el foro destinado para tal fin (TecDigital).
- Cualquier intento de fraude se calificará con nota de 0, además de la aplicación del reglamento vigente.